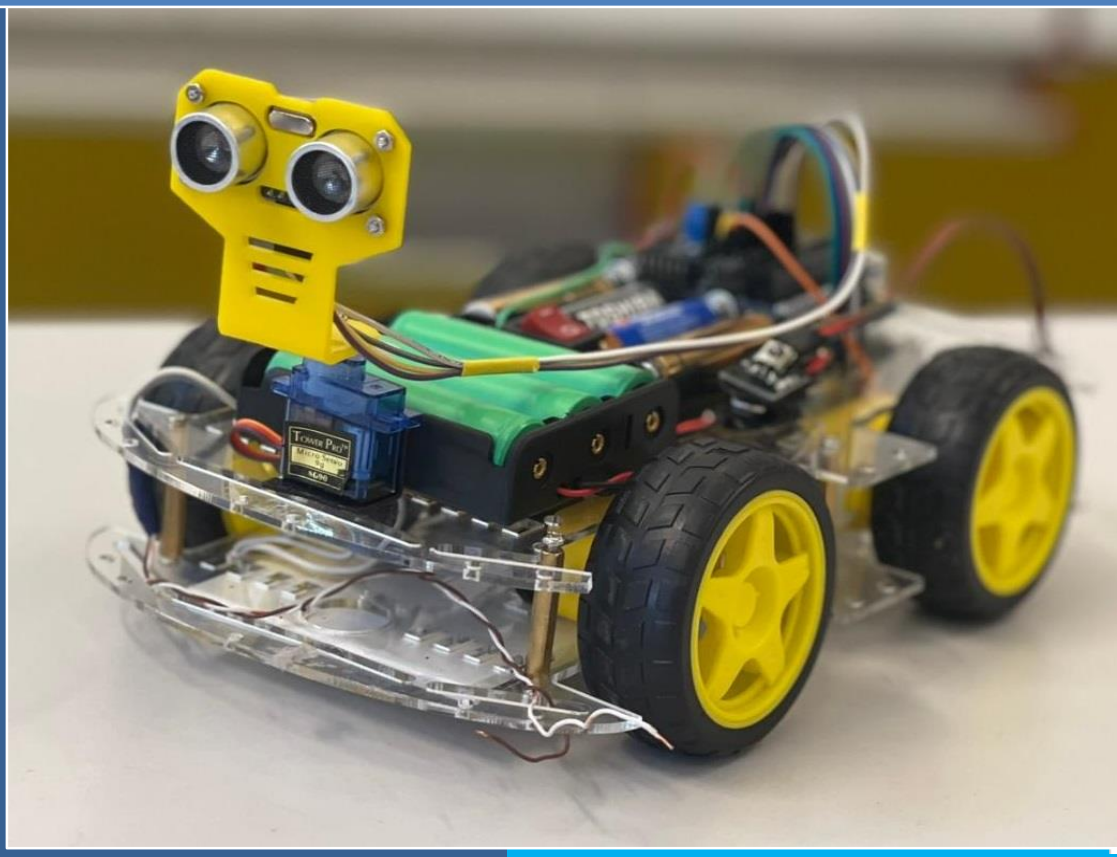
Љубица Маркудова

Микрокомпјутери и нивно програмирање



Електротехничка струка

Електротехничар за компјутерска техника и автоматика

Втора година

Љубица Маркудова

Микрокомпјутери и нивно програмирање

II година

2023

Рецензенти:

- **Вон. проф. д-р Данијела Ефнушева, дипл. елек. инж., УКИМ - Факултет за електротехника и информациски технологии - Скопје ФЕИТ – претседател**
- **Жанета Сервини, дипл. елек. инж., СОТУ ”Ѓорѓи Наумов”, Битола - член**
- **Марјан Малинов, дипл. елек. инж., ССОУ ”Коле Неделковски”, Велес – член**

Наслов на учебникот: Микрокомпјутери и нивно програмирање за II година

Сектор/ струка: ЕЛЕКТРОТЕХНИКА / ЕЛЕКТРОТЕХНИЧКА

Квалификации/образовни профили:
електротехничар за компјутерска техника и автоматика

Автор: Љубица Маркудова, дипл. елек. инж.

Лектор: Елена Саздовска

Компјутерска изработка: Љубица Маркудова

Илустрација за насловна страна: Кристина Башоска

Издавач: МИНИСТЕРСТВО ЗА ОБРАЗОВАНИЕ И НАУКА НА РЕПУБЛИКА СЕВЕРНА МАКЕДОНИЈА Со одлука на Националната комисија за учебници за основно и средно образование при Министерството за образование и наука на Република Северна Македонија, со дел. бр. 26-1916/1 од 19.07.2023 година, се одобрува употребата на овој учебник

Предговор

Овој учебник е наменет за учениците од втора година во средните стручни училишта, електротехничка струка, за образовниот профил електротехничар за компјутерска техника и автоматика. Учебникот е работен според модуларно дизајнираната наставна програма и е во согласност со наставните содржини по предметот Микрокомпјутери и нивно програмирање во втора година. Цел на новата наставна програма е осовременување на наставата и наставните програми во стручното образование и целосно задоволување на барањата на пазарот на труд.

Наставниот предмет Микрокомпјутери и нивно програмирање во втора година е застапен со фонд од 2 часа неделно, или 72 часа годишно. Половина, или 36 часа од вкупниот фонд на часови треба да се реализираат како практична настава или вежби. Часовите за практична настава или вежби се реализираат во групи, согласно Упатството на Министерството за образование и наука. Ваквиот концепт нуди можност стекнатите теоретски знаења да се применат практично, што учењето го прави поинтересно и забавно. Атрактивноста и интересот што го предизвикува развојот на микрокомпјутерите и нивното програмирање многу придонесоа за желбата на авторот да го напише овој учебник. Неговите содржини се разликуваат од содржините на класичните учебници по информатика бидејќи изработените апликации се имплементираат во програмабилни вградливи системи, како што се на Arduino микроконтролерската платформа и Raspberry Pi микрокомпјутерот. Тие, заедно со персоналните компјутери, ќе бидат неопходни за реализација на вежбите.

Учебникот опфаќа четири модуларни единици.

Првата модуларна единица, Употреба на микрокомпјутери, дава голем број примери за примената на компјутерите. Во зависност од поставената цел, се избира соодветен компјутерски систем и софтвер за изработка на апликации.

Во втората модуларна единица, Составни делови на микрокомпјутерите, учениците ги утврдуваат и ги прошируваат своите знаења за карактеристиките и функцијата на основните хардверски компоненти. Покрај инсталацијата и на монтажата на персонален компјутер ставен е акцент ѝ на електронските компоненти за Arduino-базираната микроконтролерска платформа и нивното поврзување.

Со инсталацијата на оперативните системи Windows 10, Ubuntu 17 Linux и Raspbian ќе се запознаеме во третата модуларна единица, Инсталација на оперативен систем. Дадено е ѝ упатство за инсталација и работа со развојната средина на Arduino-базираната микроконтролерска платформа.

Четвртата модуларна единица, Програмирање на микрокомпјутери е најобемна, бидејќи не само што ги проучува програмските јазици C++ и Python, туку и нив ги користи за програмирање на Arduino Uno развојната платформа и Raspberry Pi микрокомпјутерот.

Во учебникот, секоја тема содржи практични вежби за примена на теоретските знаења и прашања за да се направи проверка на постигнувањата на учениците.

Авторот на овој учебник се обиде наставниот материјал да го презентира на колку што е можно поедноставен и разбирлив начин, со прецизни објаснувања на постапките и процесите и изработка на јасни цртежи и слики. За изработка на цртежите и илустрациите беа користени две програми, Microsoft Visio и Fritzing. Fritzing е апликација со отворен тип, која нуди одлична поддршка за изработка на електрични шеми со Arduino Uno, Raspberry Pi и протоплочки.

Овој учебник го посветувам на нашите ученици и искрено се надевам дека ќе им помогне во остварувањето на нивните професионални цели.

Им се заблагодарувам на моето семејство и на колегите, за несебичната и сестрана поддршка во изработката на овој учебник. Исто така, се заблагодарувам на рецензентската комисија, која со своите корисни сугестии придонесе за подобра реализација на овој учебник.

Од авторот

Содржина

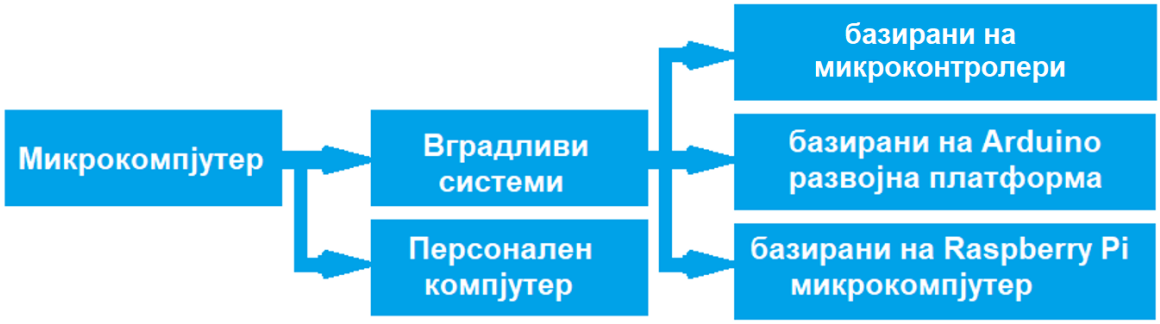
1. Употреба на микрокомпјутери

1.1. Поим и видови на микрокомпјутери

Микрокомпјутерите се користат насекаде и без нив не може да се замисли модерниот живот. Речиси не постои уред што во својот состав не содржи микрокомпјутер. Такви се: апаратите за домаќинство, мултимедијалните уреди и уредите за забава, автоматите, уредите за комуникација, медицинските апарати, индустриските погони, сообраќајните средства, канцелариските уреди итн.

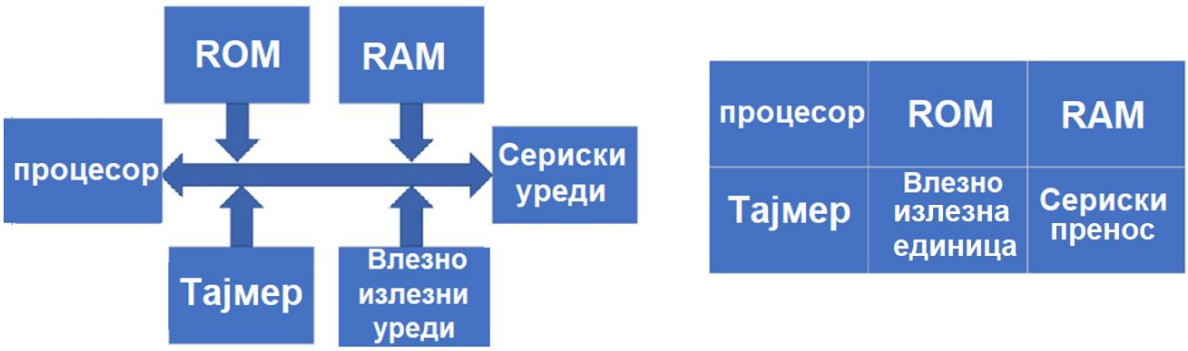
Под **микрокомпјутер** подразбираме систем составен од централна процесорска единица, работна RAM-меморија и трајна меморија за чување на програмите и добиените резултати од процесирањето. Овие компоненти се неопходни и без нив не може да се врши обработка на податоците. Потоа се додаваат влезно-излезни уреди кои овозможуваат човекот да му дава информации на микрокомпјутерот, но ѝ микрокомпјутерот да му дава информации на човекот.

Брзиот напредок на технологијата го смени или поточно го прошири значењето на поимот микрокомпјутер. Порано овој поим тесно се поврзувал со поимот персонален компјутер. Но, денес ова е значително променето, што се должи пред сѐ на појавата и развојот на вградливите микрокомпјутерски системи. На слика 1.1. е прикажана современата поделба на микрокомпјутерите. Микроконтролерите и развојните хардверски платформи се основа на вградливите системи (анг. Embedded System). [1].



Слика 1.1. Поделба на микрокомпјутерите

Процесорот ги извршува програмите кои се наоѓаат во трајната меморија, а во RAM меморијата се чуваат податоците кои моментално се обработуваат. Сите тие, процесорот, трајната ROM и привремената RAM меморијата претставуваат интегрирани кола. Се поставило прашањето, зошто тие да бидат посебни интегрирани кола, кога можат да бидат едно единствено? На ваков начин е добиен **микроконтролер** кој всушност претставува микрокомпјутер во облик на интегрирано коло. Тој во својот состав содржи микропроцесор, мемории и проводници за пренос на податоци. Слика 1.2. претставува споредба меѓу персонален компјутер и микроконтролер. Во персоналните компјутери секоја од наведените хардверски компоненти е посебен чип на матичната плоча и затоа секоја е претставена со посебен блок, а стрелките кои ги поврзуваат блоковите ги претставуваат водовите на матичната плоча кои служат за пренос на податоци. Микроконтролерот е дизајниран да биде сè во едно затоа сите хардверски компоненти се прикажани со еден заеднички блок. Освен процесор, трајна и работна меморија микроконтролерот во својот состав може да содржи тајмери за мерење на време, аналогно-дигитален претворувач и единица за сериска комуникација.



Слика 1.2. Споредба меѓу персонален компјутер и микроконтролер

Поради своите мали димензии и ниската цена на чинење микроконтролерот може да се вгради во многу електронски уреди. За разлика од персоналните компјутери, микроконтролерите имаат многу помала работна фреквенција, помал мемориски капацитет и мрежно потешко се поврзуваат. Како што не можеме да ги користиме персонални компјутери без тастатура, глушец и монитор, така и микроконтролерите не можат да функционираат сами за себе, без да приклучиме сензори и извршни уреди. Микроконтролерите примаат информации за средината што ги опкружува преку сензорите, приклучени на нивните влезни пинови. Потоа овие податоци се процесираат во микропроцесорот и по извршувањето на програмата се активираат извршните единици приклучени на излезните пинови од микроконтролерот (електромотори, пумпи, дисплеи итн.). Пред да биде вграден во некој електронски уред микроконтролерот треба да се испрограмира. За пишување на корисничка програма потребна е развојна програма, или уште позната како **интегрирана развојна средина** (анг. IDE- Integrated Development Environment). За да корисничката програма се имплементира во меморијата на микроконтролерот

потребен е специјален уред таканаречен програматор, кој се поврзува со компјутер најчесто преку USB кабел. Микроконтролерот се поставува на неговото подножје во програматорот, корисничката програма се впишува во трајната меморија и потоа микроконтролерот се префрла на неговото подножје во корисничкиот уред.

Покрај микроконтролерските базирани вградливи системи се почесто се користат Arduino и Raspberry Pi базираните системи. Arduino претставува микроконтролерска базирана развојна платформа со големина на кредитна картичка. Микроконтролерските **развојни платформи** претставуваат електронски плочки коишто освен микроконтролер содржат сериски програматор, кристален осцилатор и аналогни и дигитални влезови-излези**.** Програмерот не треба да се грижи за изборот на програматор, а вградените влезно-излезни пинови дозволуваат едноставно поврзување со сензорите и извршните уреди. Хардверските развојни платформи се одликуваат со едноставно програмирање. На пример, за нивно програмирање се користат бит инструкции. Со бит тест инструкциите се проверуваат влезните пинови на кои се приклучени сензори, а со бит сет и бит ресет инструкциите се вклучуваат и се исклучуваат излезните уреди. За изработка на електронскиот уред, кој треба да го управуваме, наместо печатена плочка може да се користи експериментална протоплочка (анг. Protoboard), која ни дозволува да експериментираме со дизајнот на уредот.

Иако по големина се речиси исти со Arduino, Raspberry Pi има многу помоќен процесор и поголем капацитет на RAM меморијата. Наместо микроконтролер тој содржи специјален чип со процесор и вградена графичка картичка и RAM меморијата е посебен чип на електронската плочка. За разлика од Arduino тој има свој оперативен систем и затоа Raspberry Pi претставува микрокомпјутерски систем наречен компјутер на плочка (анг. SBC-Single Board Computer).

Ние ќе ги проучиме Arduino платформата и Raspberry Pi и ќе се запознаеме со нивните составни делови, функционалноста, поврзувањето со влезно-излезните уреди, начинот на нивно програмирање.

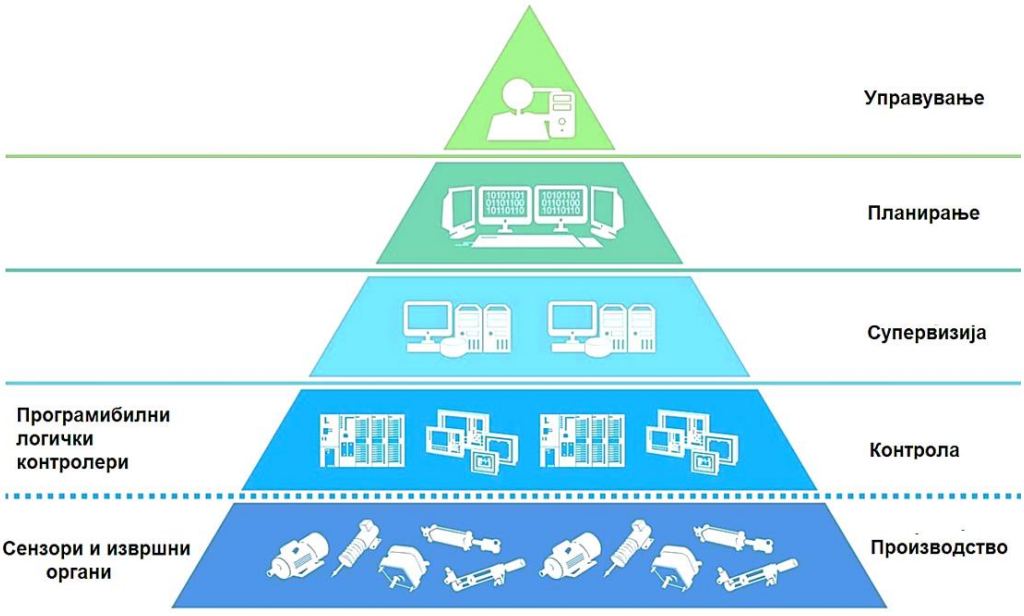
1.2. Примена на микрокомпјутерите

Откако се запознавме со поимот микрокомпјутерски систем, сега подетално да се запознаеме со нивната примена и апликации. Постојат бесконечно многу примери за примена на микрокопјутерите. Ќе наведеме неколку од нив.

- **Медицина**. На почетокот, компјутерите биле користени за складирање и обработка на медицински податоци поради подобра прегледност и достапност. Во оваа категорија спаѓа и телемедицината, која дозволува грижа за пациентите од далечина. Денеска не може да се замисли

дијагностика и навремено откривање на болести без примена на компјутерска томографија, магнетна резонанца и 3D рендгенски зраци. Хирурзите сè повеќе користат виртуелни модели за симулација на хируршки процедури. Вградувањето чипови во човечкото тело е веќе едноставна постапка. На пример, пејсмејкерот претставува чип што генерира електрични импулси со точно дефинирана амплитуда и фреквенција и ја стимулира работата на срцето. Овој чип има сопствено напојување и истовремено може да се користи и за следење на работата на срцето.

- **Автомобилска индустрија**. Во еден современ автомобил има над 70 управувачки единици и 100 сензори. Од иновациите, 80% се засноваат на примена на микроконтролери. Најважен микроконтролер е електронската управувачка единица (ECU – Electronic Control Unit). Најважни функции се контрола на количеството на гориво, тајмингот на палење и контрола на издувните гасови. Главни влезни величини се моќноста на моторот и бројот на вртежи. Постојат и други влезни величини, како што се температурата и протокот на вшмуканиот воздух, температурата на течноста за разладување, положбата на вратилата. Излезни величини се количеството на гориво што се вбризгува во еден циклус и моментот на палење. Овие податоци се запишани во таканаречена матрица, врз основа на што се добиваат излезните коефициенти. Матрицата е, всушност, EEPROM-меморија и таа е различна за различни типови на автомобили. По пресметката, микроконтролерот ги пренесува добиените вредности до актуаторот – вбризгувачите. Доколку температурата е повисока од предвидената, се продолжува времето на отвореност на вбризгувачот (поголем воздушен проток).
- **Автоматизација на индустријата.** Автоматското управување е сложен процес и може да се подели на неколку нивоа.



Слика 1.3. Примена на микрокомпјутери во автоматиката

Првото ниво се сензорите и извршните единици што ги следат и ги менуваат производствените параметри. Второто ниво се програмабилните контролери кои согласно програмата делуваат моментално, без да анализираат. Третото ниво на супервизија ги обработува податоците, изработува математички модели, ги проучува машините и врши оптимизација, изготвува извештаи. Потребен е брз компјутер што брзо ќе ги обработува податоците, голема трајна меморија за чување на минатите вредности и user friendly графика за полесно следење на параметрите.

- **Банкарство и финансии**. Многу финансиски институции вложуваат капитал во дигиталните технологии. Банкарските трансакции се извршуваат преку мобилни и веб-платформи, без оглед во која земја престојувате. Компјутерите се користат за изработка на статистички извештаи, следење на светските трендови и повторна модернизација на електронското банкарство. Такви трендови се вршење трансакции со мобилен телефон наместо со платежна картичка и користење биометриска авторизација заради поголема сигурност и заштита од злоупотреба (користење на отпечаток од прст или компјутерско препознавање на лик).

Секоја област си има свои карактеристики и потребна е специјализација за да се стане експерт за компјутери во таа област. Ние ќе се запознаеме со основите на хардверот и софтверот на микрокомпјутерите, што е многу важно за понатамошниот професионален развој, според желбите и можностите на поединецот.

1.3. Arduino-микроконтролерска базирана развојна платформа

Arduino е првата голема **микроконтролерска платформа** со програмски код од отворен тип (open source). Ова значи слободен пристап до многу готови кориснички програми и датотеки кои можат бесплатно да се преземаат од интернет и менуваат со цел да се подобрат перформансите на електронскиот уред. Arduino беше лансирана во 2005 година, со цел да се поедностави процесот на електронски прототипни иновации. Тоа им овозможи на обичните луѓе со малку или без технички предзнаења да изградат интерактивни производи.

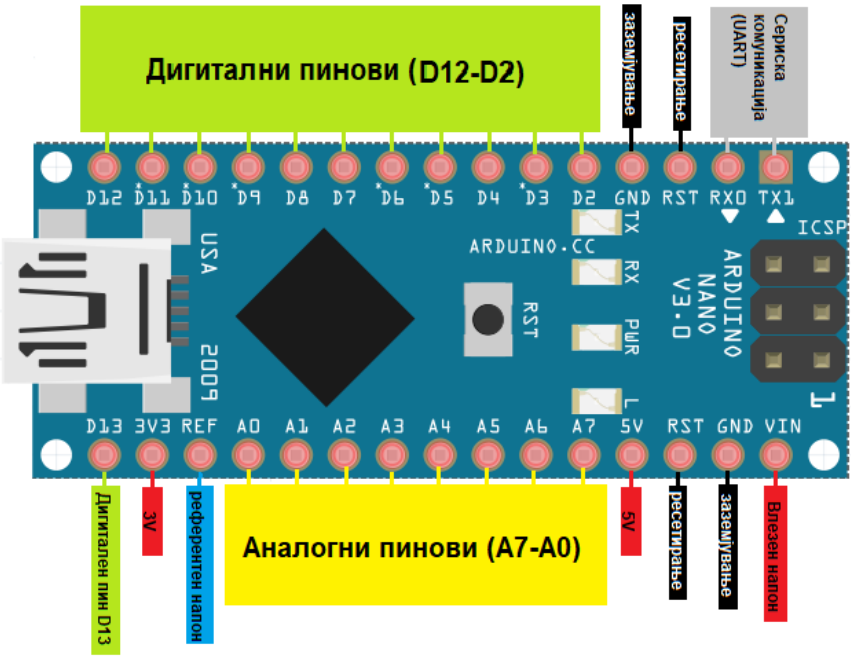
Подоцна детално ќе се запознаеме со хардверските компоненти на Arduino Uno R3 платформата. Ќе ја проучиме нејзината архитектура, влезно-излезните единици, начинот на поврзување, алатките во развојната средина и неколку готови програми што ќе ги имплементираме во програмската меморија.

На влез од Arduino платформата можат да се поврзат: тастери, прекинувачи, готови тастатури, сензори (за температура, притисок, проток, движење и др). Излезите можат да се поврзат со најразлични излезни уреди како што се: лед-диоди, светилки, зујалици, мотори, дисплеи. Пример, дали лед-диодата ќе светне или моторот ќе почне да работи зависи од тоа дали тастерот бил притиснат или дали сензорот детектирал промена на некоја физичка величина. Дополнителните штитови (анг. Shields) се електронски плочки што се монтираат на основната Ардуино, со што се прошируваат нејзините основни можности и можат да се извршуваат дополнителни функции како контрола на мотор, поврзување со сензор, безжична комуникација.

За програмирање на Arduino се користи програмскиот јазик C/C++, модификација на C и C++. Постои разлика меѓу инструкциските множества за Arduino и за персонален компјутер, бидејќи Arduino манипулира со влезните-излезните уреди. Исто така, самото програмирање бара добро познавање на хардверот посебно на начинот на поврзување со сензорите и извршните единици.

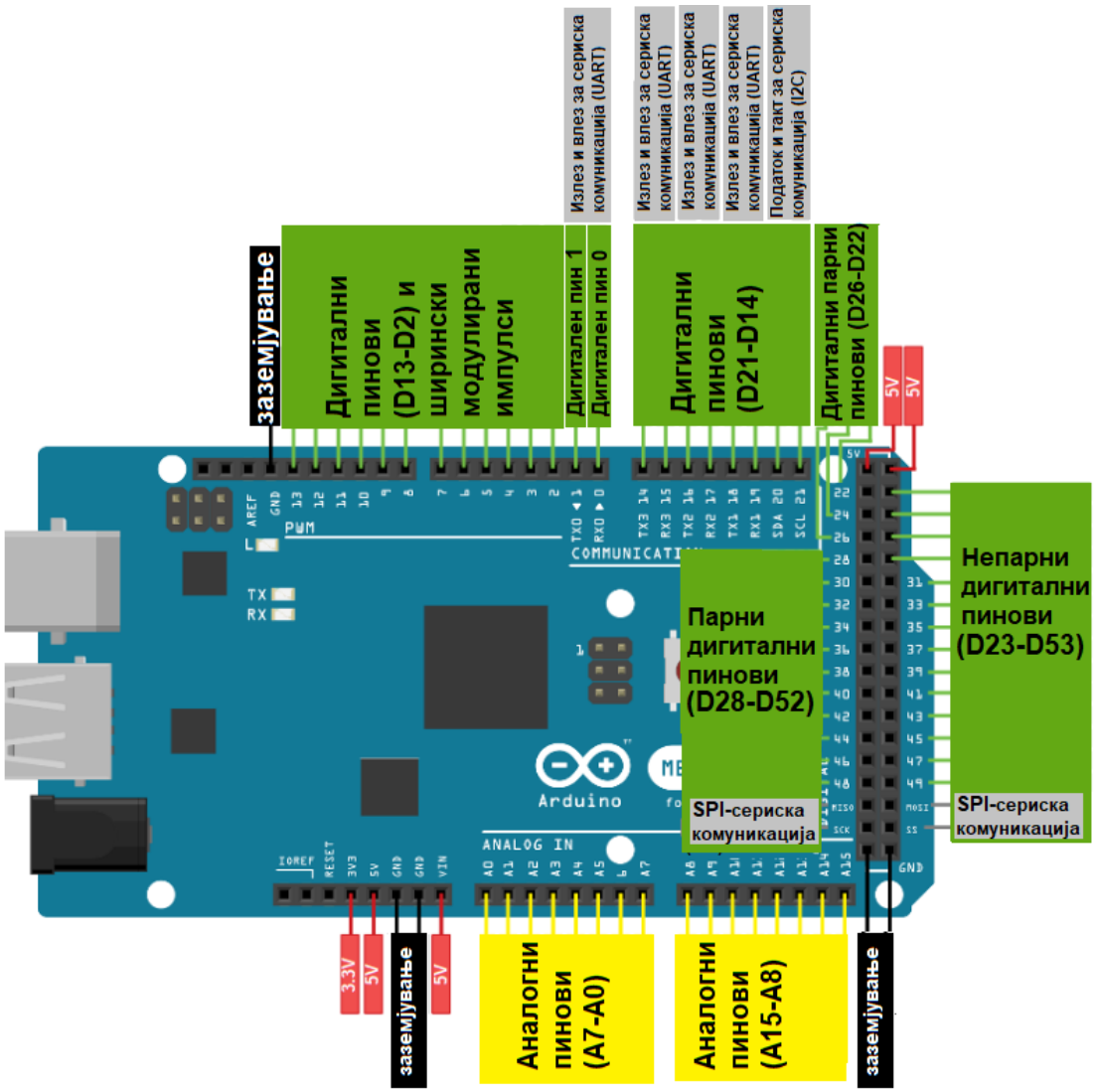
Постојат повеќе од **15 различни модели** на Arduino платформи, а моментално најпопуларни се Arduino Uno R3, Leonardo, Nano, Pro Mini, Mega 2560 R3, Due. Накратко ќе ги објасниме разликите меѓу нив. **[2]**

- Третата верзија на **Arduino Uno** (Rev3 или R3) е најдобра платформа за почетници и истата ќе ја објасниме и користиме при реализацијата на практичните вежби. Оваа платформа има осумбитен ATmega 328 микроконтролер, 14 дигитални влезно-излезни пинови (од кои шест можат да се користат како излези на ширински модулирани импулси), 8 аналогни влезови, USB-конектор, влез за напојување. Платформата може да се напојува со батерија или преку AC/DC-конвертор. Сите останати модели ќе ги споредуваме со платформата Arduino Uno R3.



Слика 1.4. Пин дијаграм на Arduino Nano платформа

- **Arduino Nano**. Arduino Uno R3 и Arduino Nano имаат исти процесори, мемориски капацитети и пинови. Но постојат три основни разлики. Arduino Nano е со двапати помали димензии, што е предност при монтажа. Второ Arduino Nano има женски пинови од двете страни и лесно може да се постави на протоплочка. Бидејќи Arduino Nano има мини USB, тој не може да се поврзе со многу периферни уреди, за разлика од Arduino Uno R3, кој има USB приклучок тип B.
- **Arduino Pro Mini** е една шестина од големината на Arduino Uno R3. Напонот на напојување е помал и изнесува 3,3 V, за разлика од 5 V на Arduino Uno R3. Има двапати помала работна фреквенција и нема можност за сериска комуникација. Но, голема предност е неговата мала потрошувачка на енергија и тоа го прави погоден за поставување на тешко достапни места.
- **Arduino Mega 2560** е како постар брат на Arduino Uno R3. Пин дијаграмот на Arduino Mega 2560 е прикажан на слика 1.5.



Слика 1.5. Пин дијаграм на Arduino Mega 2560

Arduino Mega 2560 има 54 дигитални влезно-излезни пинови, од кои 15 можат да се користат како излези на ширински модулирани импулси. Располага со 14 аналогни влезови Овозможува сериска комуникација со четири уреди истовремено поради четирите вградени универзални асинхрони порти за прием и испраќање на податоци (анг. UART-Universal Asynchronous Receiver-Transmitter**).** Располага со 256 KB програмска меморија, 8 KB RAM меморија и 4 KB податочна меморија. Гледаме дека работната меморија на Arduino Mega е осумпати поголема од онаа на Arduino Uno R3.

- **Arduino Leonardo**. Основната разлика е во процесорите и во незначително различниот број на пинови. Процесорот на Arduino Leoardo не може да се замени во случај на дефект, но тој има USB-поддршка за поврзување со други уреди, а на Arduino Uno R3 му треба посебен контролер. Но, од друга страна, Arduino Leonardo е некомпатибилен со многу додатоци на Arduino Uno R3.
- **Arduino Due** има многу појак процесор од Arduino Uno R3, но и неговата цена е речиси двапати поголема. На пример, работната фреквенција на Arduino Due изнесува 84 MHz, наспроти 16 MHz на Arduino Uno R3. Arduino Uno R3 е осум битен, а Arduino Due е 32-битен. Може да извршува 104 милиони инструкции во една секунда. Поради тоа, Arduino Due најчесто се користи за вршење пресметки со децимални броеви кои уште се познати под името броеви со подвижна запирка (анг. floating point number).

1.4. Микрокомпјутерски систем Raspberry Pi

Raspberry Pi се најголемите конкуренти на Arduino. Првата генерација на Raspberry Pi излезе на пазарот во 2012 година, во Англија. Доказ за неговата популарност е тоа што во првите шест месеци беа продадени околу половина милион примероци. Слично како Adruino, Raspberry Pi беше наменет за студентите по компјутерски науки на Универзитетот во Кембриџ, со цел програмирањето да биде полесно. Денес Raspberry Pi има големо научно, образовно и практично значење.

Иако се конкуренти, Raspberry Pi многу се разликува од Arduino. По своите карактеристики Raspberry Pi е многу сличен со персонален компјутер и може да се поврзе со монитор, тастатура и глувче. Raspberry Pi располага со функционален **оперативен систем** (најчесто Linux), поради што може да извршува комплексни програми. Недостаток е тоа што софтверот за Raspberry Pi не е од отворен тип, како што е случај со Arduino. Развојната средина на Raspberry Pi не содржи бесплатни готови програми и библиотеки со драјвери, туку за тоа се потребни посебни лиценци. Arduino многу лесно се поврзува со сензори и со извршни уреди, а пак за Raspberry Pi ефикасно да се поврзе со некој сензор, потребен е дополнителен софтвер. Од друга страна, сите модели на

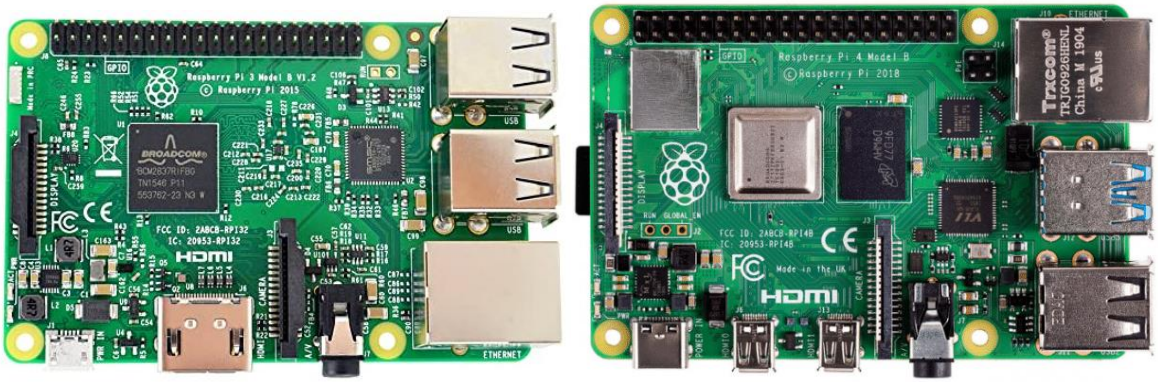
Raspberry Pi поседуваат **графичка картичка** којашто го прави многу лесен за поврзување со видеоуреди. Најважна компонента во сите модели е Broadcom SoC (анг. System on a Chip) чипот со ARM процесор и интегрирана графичка картичка. Интересно е тоа што Raspberry Pi нема интерна трајна меморија. Најчесто се користи екстерна SD-картичка за чување на оперативниот систем и како програмска меморија. Преку 40-пинскиот конектор (анг. GPIO-General Purpose Input Output) можат да се поврзат до 27 различни уреди. Raspberry Pi поседува **мрежни конектори** и можност за поврзување преку Wi-Fi, што дополнително го олеснува пристапот до интернет-мрежата. Некои модели содржат и интегриран Bluetooth, како уште еден начин за безжична комуникација. Raspberry Pi може да се програмира во речиси сите програмски јазици. Најчесто користен е програмскиот јазик Pyton, но може да се користи и кој било друг програмски јазик како што е C/C++, Java Script, Scratch. **[3]**

	Raspberry Pi 4 Модел B	**Raspberry Pi 3 Модел B**	**Raspberry Pi 2 Модел B**	**Raspberry Pi Модел B**
Процесор	QUAD Core со 1.5 GHz	QUAD Core со 1.2 GHz	QUAD Core со 900MHz	Едно јадро со 700MHz
Графичка картичка	Videocore IV	Videocore IV	Videocore IV	Videocore IV
Трајна меморија	MicroSD	MicroSD	MicroSD	MicroSD
RAM	До 4GB SDRAM со 400 MHz	1GB SDRAM @ 400 MHz	1GB SDRAM со 400 MHz	512MB SDRAM со 400 MHz
Работна фреквенција	1,5GHz	1,4GHz	900MHz	700MHz
Chipset	Broadcom BCM2838 64Bit	Broadcom BCM2837 64Bit	Broadcom BCM2836 32Bit	Broadcom BCM2837 32Bit
USB 2.0	4 x USB порти	4 x USB порти	4 x USB порти	4 x USB порти
Мрежен приклучок	RJ45	RJ45	RJ45	RJ45
Wi-Fi	Вграден	Вграден	Нема	Нема
Bluetooth	Вграден	Вграден	Нема	Нема
Влезови и излези со општа намена	40-пински конектор	40-пински конектор	40-пински конектор	40-пински конектор
Напојување	5 V	5 V	5 V	5 V

Табела 1.1. Споредба на карактеристики на различни модели на Raspberry Pi

Досега се произведени **четири генерации** на Raspberry Pi и во секоја генерација има по неколку модели. Првата генерација од 2012 година е со два модела, A и B. Моделот B има подобра конфигурација, а моделот A има помала меморија и помал број на приклучоци, но и помала потрошувачка на енергија. Raspberry Pi 2, модел B излезе на пазарот во 2015 година, а следната година се појави и третата генерација. Во 2019 излезе последна генерација, Raspberry Pi 4. Во табелата 1.1, дадени се спецификациите за секоја генерација. Од табелата

може да се заклучи дека Raspberry Pi според своите можности е многу поблиску до персонален компјутер отколку Ардуиното.



Raspberry Pi 3 Модел B Raspberry Pi 4 Модел B

Слика 1.6 Надворешен изглед на два модели на Raspberry Pi

	Raspberry Pi	Arduino
предности	64- битен процесор со работна фреквенција од 1.5GHz и можност за истовремено извршување на повеќе задачи	Лесно поврзување со аналогни сензори и прецизна контрола на работа на мотори
	Вграден мрежен приклучок, можност за безжично поврзување преку Wi-Fi или Bluetooth	Голем избор на додатоци за и зголемување на функционалноста
	Функционален оперативен систем	Програмите ги извршува веднаш по вклучување на напојувањето без дополнителни нагодувања
	Содржи аудио излез, приклучок за камера, HDMI излез и неколку USB	Ниска цена на чинење
	Одличен избор за проекти како што се роботи или видеоигри	Извршува едноставни апликации во реално време
	Може да се програмира во повеќе програмски јазици	Поедноставен хардвер и софтвер
недостатоци	За поврзување на електронски компоненти потребен е дополнителен софтвер	Работна фреквенција од 16MHz, мала брзина на работа и извршува само една програма
	Висока цена на чинење	Без Ethernet додаток не може да се поврзе со Интернет мрежата
	Поголемо загревање при употреба	За програмирање се користи само програмскиот јазик C/C++

Табела 1.2. Споредба меѓу Arduino и Raspberry Pi

Ние ќе се запознаеме со начинот на инсталирање на оперативниот систем Raspbian, со основните наредби и функции на програмскиот јазик Python и ќе реализираме неколку проекти што ги користат извонредните интерфејс перформанси на Raspberry Pi.

Многу е важно паметно да се избере развојната платформа за реализација на дадена практична вежба или проектна задача. Во тоа може да ни помогне табела 1.2. во која споредбено се дадени предностите и недостатоците на Arduino и Raspberry Pi. Raspberry Pi е софтверски ориентиран односно задачите ги извршува пред сè со примена на програми, а Arduino е хардверски ориентиран. На пример, доколку сакаме да направиме неколку фотографии, истите да ги обработиме и објавиме на некоја веб страна тогаш Raspberry Pi нуди поедноставни решенија. Но, доколку сакаме да контролираме брзина на вртење на мотор на еднонасочна струја тогаш платформата Arduino Uno е посоодветна поради поголемата брзина на реакција и поголемата прецизност.

Arduino и Raspberry Pi се дел од платформата Интернет на нештата (анг. IoT- Internet of Things). Оваа платформа е нов тренд во технологијата и се занимава со поврзување на сензори и уреди, нивно вмрежување со примена на различни видови комуникациони протоколи и анализа на податоци. Оваа платформа не е предмет на наша анализа, но нејзината поврзаност со Arduino и Raspberry Pi е неминовност.

Заклучоци

Под компјутерски систем подразбираме систем составен од централна процесорска единица, работна RAM меморија и трајна меморија за чување на програмите и добиените резултати од процесирањето.

Според современата поделба, покрај персоналните компјутери, во микрокомпјутери спаѓаат и вградливите микрокомпјутерски системи. Тие можат да се базираат на микроконтролери, хардверски развојни платформи или Raspberry Pi.

Вградливите микрокомпјутерски системи се многу популарни поради нивната мала цена на чинење, флексибилност, едноставно програмирање.

Микроконтролерската развојна платформа претставува електронска плочка во која се вградени: микроконтролер, сериски програматор, кристален осцилатор, порти, аналогни и дигитални влезови и излези. Таа преку USB кабел може да се поврзе со компјутер и да се внесе готова програма во нејзината програмска меморија.

На влез од микроконтролерската развојна платформа можат да се поврзат: тастери, прекинувачи, готови тастатури, сензори (за температура, притисок, проток, движење и др). Излезите можат да се поврзат со најразлични излезни уреди како што се: лед диоди, сијалици, бипери, мотори, дисплеи.

Постојат повеќе од 15 различни модели на Ардуино платформи, а моментално најпопуларни се Arduino Uno R3, Leonardo, Nano, Pro Mini, Mega 2560 R3, Due.

Raspberry Pi располага со функционален оперативен систем (најчесто Linux) поради што може да извршува комплексни програми. Сите Raspberry Pi модели поседуваат графичка картичка која го прави многу лесен за поврзување со видео уреди. Raspberry Pi поседува мрежни конектори и можност за поврзување преку Wi-Fi што дополнително го олеснува пристапот до Интернет мрежата. Некои модели содржат и интегриран Bluetooth како уште начин за безжична комуникација

Основни предности на Aruino платформата се: лесно поврзување со сензори, голем број на додатоци, извршување едноставни апликации во реално време.

Прашања за повторување

1. Што подразбираме под поимот микрокомпјутерски систем?
2. Која е суштинската разлика меѓу микрокомпјутер и микроконтролер?
3. Објасниј ја функцијата и значењето на микрокомпјутерот ECU (анг. Electronic Control Unit) вграден во поновите автомобилите?
4. Кои хардверски компоненти ги содржи во себе платформата Arduino Uno?
5. Набroj неколку различни модели од серијата Arduino развојни платформи?
6. Кои се трите основни разлики меѓу Arduino Uno и Arduino Nano развојните платформи?
7. Опиши ги можностите за поврзување со други уреди на микрокомпјутерот Raspberry Pi, споредено во однос на Arduino Uno развојната платформа!
8. Кои се предностите и недостатоците на Arduino платформата во однос на Raspberry Pi?

2. Составни делови на микрокомпјутери

2.1. Составни делови на микрокомпјутерски систем

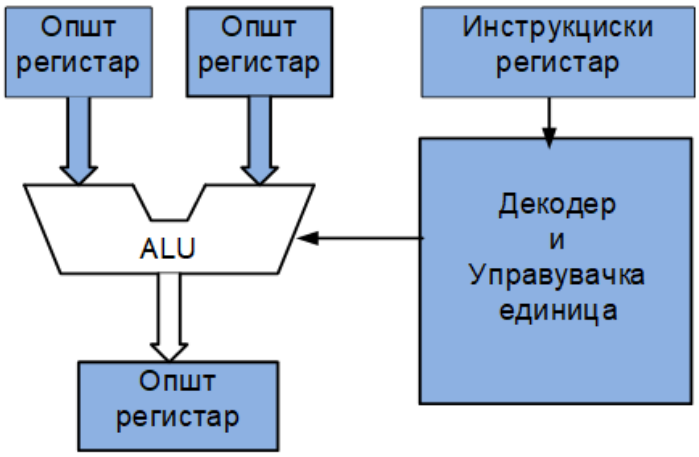
За да се испрограмира еден микроконтролер или развојна платформа потребно е да се познаваат карактеристиките на нивните хардверски компоненти. Под хардвер се подразбираат механичките делови од кои е направен микрокомпјутерот. Составни хардверски компоненти се: микропроцесор, мемории, магистрали и периферни уреди. **Микропроцесорот** ги обработува податоците, **мемориите** ги чуваат, а **магистралите** ги пренесуваат. **Периферните уреди** се познати и под името влезно-излезни единици. Периферните уреди ги претвораат корисничките информации во дигитални сигнали, нули и единици, но и обратно. Кориснички информации се оние информации што ги користи човекот, како што се: звук, слика, текст, видео.

Софтверот го контролира хардверот. **Софтверот** претставува збир од програми што овозможуваат користење на хардверот. Функционално, програмите се поделени во три групи: оперативен систем, развојни средини и апликативен софтвер. **Оперативниот систем** е посредник меѓу хардверот и софтверот. Оперативниот систем е софтвер којшто управува со хардверските компоненти: процесорот, мемориите, влезно-излезните уреди, трансферот на податоци по водовите на матичната плоча, податочните датотеки и др. Без оперативен систем, компјутерите нема да бидат употребливи за обичниот човек. Во третата тема ќе се запознаеме со различните видови на оперативни системи, нивната инсталација, приспособување и оптимизација. **Развојна средина** е софтвер за изработка на нов софтвер. Новите програми се уредуваат (анг. editing), преведуваат на машински јазик (анг. compile), се чистат од грешки (анг. debug) и се поврзуваат со други програми. Со алатките на развојните програми и изработката на апликативен софтвер ќе се запознаеме во третата и четвртата тема. [4]

2.1.1. Микропроцесор

Кај персоналните компјутери микропроцесорот е посебна хардверска компонента, чип со свое подножје на матичната плоча. Наместо микропроцесор Arduino платформите користат вграден микроконтролер (ATMega328P за Arduino Uno R3) во чиј состав е микропроцесорот заедно со програмската и податочната меморија. Микрокомпјутерскиот систем Raspberry Pi 4 користи процесор ARM Cortex-A53 кој заедно со графичката картичка е вграден во заеднички чип Broadcom SoC. Принципот на работа на микропроцесорот е ист без оглед дали е посебен чип или дел од микроконтролер.

Микропроцесорот е „**мозокот**" на компјутерот. Тој има две основни функции: извршува програми и управува со другите уреди на матичната плоча. За микропроцесорот програмите се множество од инструкции наредени во последователни мемориски локации. Микропроцесорот ги **извршува програмите** инструкција по инструкција. Откако ќе ја обработи тековната инструкција, ја презема следната инструкција од RAM-меморијата, ја обработува, и оваа постапка се повторува сè додека процесорот не дојде до крајот на програмата. **Управувачката функција** го прави микропроцесорот господар на матичната плоча (анг. master). Тој прибира информации од сите уреди во компјутерот, ги обработува, ги процесира и потоа поттикнува одредени активности, со цел да се обезбеди сигурна работа на целиот компјутер. На слика 2.1. прикажана е основната организација на микропроцесорот. **Составни делови** на еден микропроцесор се: регистри, аритметичко-логичката единица и управувачката единица со декодерот. Регистрите се брзи мемориски локации во внатрешноста на микропроцесорот кои ги содржат податоците и операцискиот код на тековната инструкција. Општите регистри ги содржат податоците за обработка и обработените податоци, а кодовите потребни за извршување на тековната инструкција се сместуваат во специфичен регистар – инструкциски регистар.



Слика 2.1. Блок-шема на основен модел на микропроцесор

Во аритметичко-логичката единица се извршуваат сите аритметички и логички инструкции. Управувачката единица прима информации од сите уреди и потоа одлучува, менува содржина на регистри, менува логичка состојба на пинови, повикува програми, активира периферни уреди итн. Нулите и единиците што излегуваат од управувачката единица ги вклучуваат или ги исклучуваат хардверските компоненти. Бројот и функцијата на контролните битови треба да се предвиди уште при дизајнирањето на микропроцесорот.

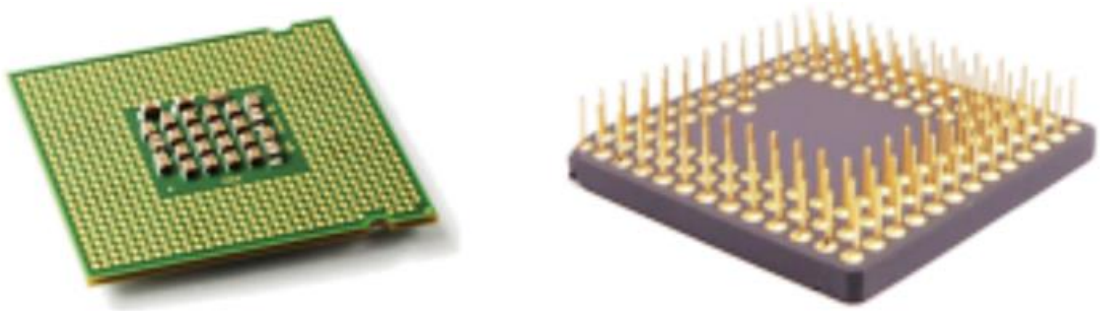
Најважни карактеристики на микропроцесорот се: работната фреквенција, големината на податоците, проточниот концепт, комплексноста на инструкциското множество, бројот на јадра, распоредот на пиновите и др.

Работна фреквенција е фреквенцијата на такт сигналот. Тактовите се важни бидејќи времетраењето на секоја операција се мери во цел број на тактови. На пример, за да се прочита еден бајт од RAM-меморијата потребни се три такта. Во првиот такт, микропроцесорот испраќа адреса, во вториот се пребарува меморијата и, во третиот такт, микропроцесорот го добива саканиот податок. Веќе истакнавме дека Raspberry Pi користи такт сигнал со фреквенција од редна величина на GHz, а кај Arduino Uno R3 такт сигналот го генерира кристалниот осцилатор поставен на самата електронска плочка и тој е со фреквенција од 16MHz. Кај Arduino платформата, програмерот може многу лесно да пристапи до бројачите чија функција е броење на тактови од кристалниот осцилатор. Ова е многу важно во апликации кои бараат прецизно мерење на време, како на пример тајмер за кујнски апарат или контрола на семафори. Да споменеме дека Arduino Uno R3 располага со три бројачи, од кој еден е 8 битен (брои од 0 до 255) и два се 16-битни (бројат од 0 до 65535).

Според големината на обработуваните податоци, микропроцесорите се делат на 8, 16, 32, 64 и 128-битни микропроцесори. Микропроцеорот ARM Cortex-A53 на Raspberry Pi е 64-битен, а микроконтролерот ATMega328P на Arduino Uno R3 е 8-битен. Микропроцесорите што обработуваат поголеми податоци се помоќни, но за тоа се потребни поголеми хардверски ресурси.

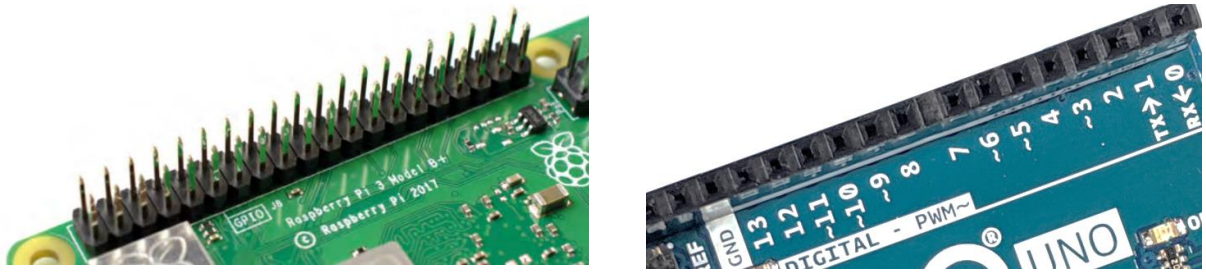
Процесорите можат да имаат комплексно или редуцирано инструкциско множество. Денес речиси сите процесори се со редуцирано инструкциско множество (анг. RISC-Redused Instruction Set Computer) и овие процесори хардверски извршуваат едноставни инструкции и се многу брзи. Микроконтролерот ATMega328P користи 10-битни операциски кодови и може да извршува 131 инструкции, што е неспоредливо малку во однос на процесорот ARM Cortex-A53 .

Пиновите се метални продолженија на внатрешните водови и служат за поврзување на микропроцесорот со другите уреди и компоненти во микрокомпјутерскиот систем. Поновите генерации процесори за персонални компјутери имаат околу 2000 пинови со пречник од 0,3 mm. Кај постарите процесори, пиновите се на процесорот, а отворите за нив се на подножјето на матичната плоча. Овој концепт е познат под кратенката PGA (Pin grid array). Спротивен концепт е LGA (Land grid array), каде што отворите се на процесорот, а пиновите на подножјето.



Слика 2.2. Изглед на пинови на процесори според LGA и PGA концептот

За разлика од персоналните компјутери, корисникот односно програмерот може да пристапува до пиновите на Arduino и Raspberry Pi и да манипулира со истите. Кога велиме пинови на Arduino и Raspberry Pi не мислиме на пиновите на самиот микроконтролер или Broadcom SoC чипот туку се мисли на пиновите на самата електронска плоча, на кои може да се поврзуваат сензори, извршни единици и најразлични додатоци кои ја олеснуваат работа со системот.



Слика 2.3. Надворешен изглед на пиновите на Raspberry Pi и Arduino

На пример, Arduino Uno R3 содржи три вида регистри за работа со пинови и преку нив одредуваме дали пинот ќе биде влезен или излезен, читаме состојба на влезни пинови, запишуваме вредности на излезни пинови.

2.1.2. Мемориска организација

Сите податоци и програми што ги извршува процесорот мора да бидат сместени во меморииte. Најважни **карактеристики на меморииte** се нивниот капацитет и брзината. Капацитетот на меморииte се мери во бајти (B) или поголеми единици за капацитет се: килобајти (KB), мегабајти (MB), гигабајти (GB), терабајти (TB). Брзината на меморијата се мери преку времето на пристап или нејзината работна фреквенција. Времето на пристап е времето потребно меморијата да се пребара и да го пронајде саканиот податок. Работната фреквенција на меморијата ни покажува колку битови може да прими или да испрати меморијата во една секунда.

Компјутерите содржат различни видови мемории. Мемориската организација подразбира поделба на функциите за секој вид меморија и начинот на пристап до саканиот податок. На сликата 2.4. е прикажана пирамидата од мемории. Како

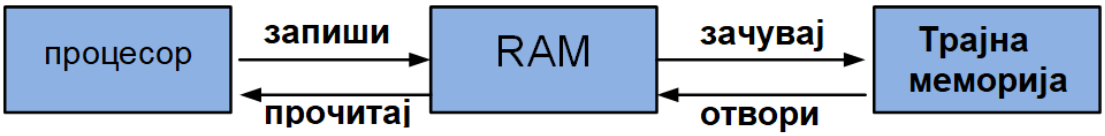
се симнуваме надолу, така се зголемува капацитетот на мемориите, а се намалува нивната брзина на работењето. Поголемата брзина на работата повлекува зголемување на цената по еден бајт. Колку мемориите се поблиску до врвот, толку поблиску се до микропроцесорот. Регистрите служат за привремено чување на податоците, пред тие да бидат обработени, но и за сместување на резултатите веднаш по нивното добивање. Регистрите се најбрзи мемориски локации во компјутерот, но нивниот капацитет изнесува само неколку бајти.

Со цел да се намали времето на чекање и да се зголеми брзината на процесорот, се користи кеш-меморија. Кеш-меморијата е многу брза RAM-меморија. Потребните податоци од побавните меморији однапред се донесуваат и се сместуваат во кеш-меморијата и таа претставува еден вид „чекалница" за микропроцесорот. Во кеш-меморијата може да се чуваат податоци што микропроцесорот ги користи почесто, а не може да ги смести во своите регистри. Кеш-меморијата може да биде во самиот микропроцесор или надвор од него.

Слика 2.4. Пирамида на меморији

RAM е кратенка од англиските зборови Random Access Memory, што во превод значи **меморија со случаен пристап**. Имено, процесорот има ист пристап до сите локации од меморијата, или ниедна локација не е со помал или со поголем приоритет. RAM е меморија од која може истовремено да се читаат, но и да се пишуваат нови податоци. Тоа е меморија со променлива содржина и претставува **работна меморија**, еден вид работен лист на кој микропроцесорот привремено ги запишува сите податоци што ќе му бидат потребни за успешно реализирање на дадената програма. Во RAM-меморијата се чуваат програмите што моментално се извршуваат во процесорот. RAM-меморијата е позната и под името примарна меморија, поради нејзината важност. RAM е **привремена меморија**. Доколку напојувањето се исклучи, податоците од RAM не можат да се обноват по повторното вклучување. Поради тоа, пред да го исклучиме компјутерот, податоците од RAM меморијата треба да се сместат во некоја трајна меморија (пр. хард-дискот), преку притискање на иконата зачувај (анг. save). Ова е прикажано на слика 2.5.

RAM-меморијата го снабдува процесорот со информации, а пак таа добива информации од некоја трајна меморија, како на пример хард дискот. Програмите се чуваат во трајната меморија и тие се пренесуваат во RAM кога треба да бидат обработени. Доколку RAM-меморијата е поголема, тогаш одеднаш ќе земе поголем дел од корисничката програма и таа ќе се обработи во микропроцесорот. Со ова се намалува бројот на преноси од RAM до трајната меморија, а тоа значи и заштеда на време. Микропроцесорот ги извршува програмите **инструкција по инструкција**. Откако ќе заврши со тековната инструкција, се повикува RAM-меморијата да ја пронајде следната. Ако се работи за линиска програма тогаш инструкциите се наредени во последователни мемориски локации (редици) од RAM-меморијата, што не е случај доколку се работи за програма со гранење.



Слика 2.5. Функција на RAM-меморијата како работна меморија на микропроцесорот

RAM-от, кеш-меморијата и регистрите се привремени мемории. Во персоналниот компјутер, за трајно чување на податоци се користат **хард дискот, SSD уредите** и надворешните мемории (компакт-дискови, USB-мемории, SD-картички и др). Хард дискот е мемориски уред со најголем капацитет. Во него трајно се чуваат сите програми, вклучувајќи го и оперативниот систем. Тоа е магнетна меморија и составена е од неколку магнетни плочи коишто ротираат со брзина од 7 200 вртежи во една секунда. Помеѓу магнетните плочи, поставен е чешел со глави за читање.

Хард диск	SSD диск
Мемориски уред со тврд диск (анг. HDD=Hard Disk Drive)	Уред со трајна полупроводничка меморија (анг. SSD=Solid State Drive)
Магнетна меморија	Електронска меморија
Максимален капацитет 15TB	Максимален капацитет 1TB
Појава на греење	Нема греење
Евтина меморија	Скапа меморија
Мала брзина	Голема брзина
Пократок век на траење	Подолг век на траење

Табела 2.1. Споредба на HDD и SDD мемориски уреди

Најголем недостаток на хард дискот е малата брзина и тој недостаток најмногу доаѓа до израз при активирањето на оперативниот систем. Поради тоа, хард дисковите почнаа да се заменуваат со нови SSD-мемории (анг. Solid State Drive), кои претставуваат електронски чипови со огромни капацитети. Овие мемории се речиси петпати побрзи од магнетната меморија, но сепак се со

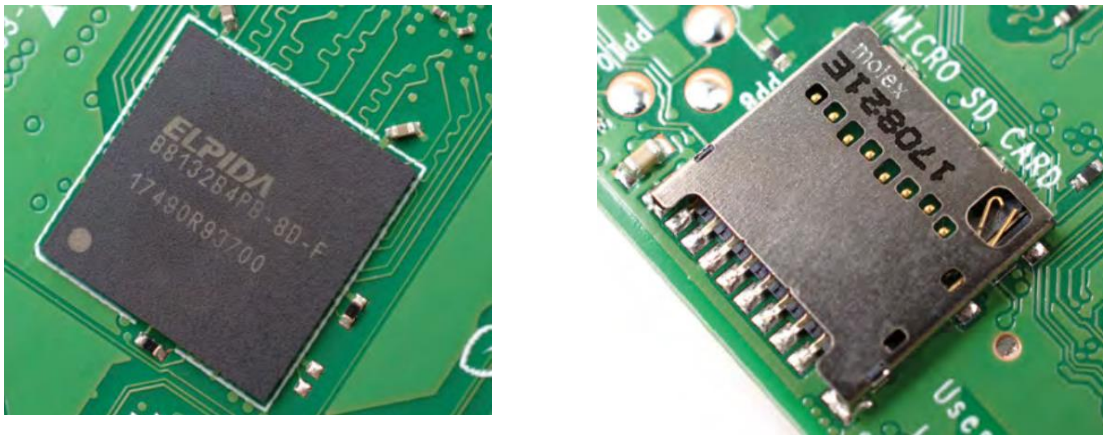
помали капацитети од неа. При изборот на трајна меморија треба да се води сметка за тоа дали ни е поважна брзината или капацитетот за чување на податоците.

По видот и капацитетот, мемориската организација на Arduino и Raspberry Pi се разликува од онаа на персоналниот компјутер, но и меѓу самите нив. Arduino Uno R3 содржи три видови на мемории и сите се сместени во микроконтролерот ATMega328P, заедно со микропроцесорот. Трите мемории се: RAM-меморија, програмска и податочна меморија. Програмската меморија е флеш (анг.flesh) меморија и во неа се впишува корисничката програма при програмирањето на Arduino Uno R3. Репрограмирањето е брзо и може да се повторува речиси неограничен број на пати. Податочната меморија е EEPROM (Electrical Ereasable Programmable Read Only Memory). Запишувањето и читањето се врши бајт по бајт, а бришењето е преку запишување на нови податоци врз старите. Капацитетите на трите мемории за неколку Arduino платформи се дадени во табела 2.2.. [2]

Arduino платформа	микроконтролер	флеш меморија	RAM-меморија	EEPROM меморија
UNO Rev3	ATMega328P	32KB	2KB	1KB
Mega 2560 Rev	ATMega2560	256KB	8KB	4KB
Leonardo	ATMega32u4	32KB	2,5KB	1KB
Nano	ATMega328P	32K	2KB	1KB

Табела 2.2. Мемории во Arduino-базирани хардверски платформи

RAM-меморијата во Raspberry Pi е посебен чип, вграден во самата електронска плочка, со капацитет од 1GB до 8GB. Како трајна меморија користи мемориска картичка Micro SD, за која постои посебен слот на електронската плочка. [3]

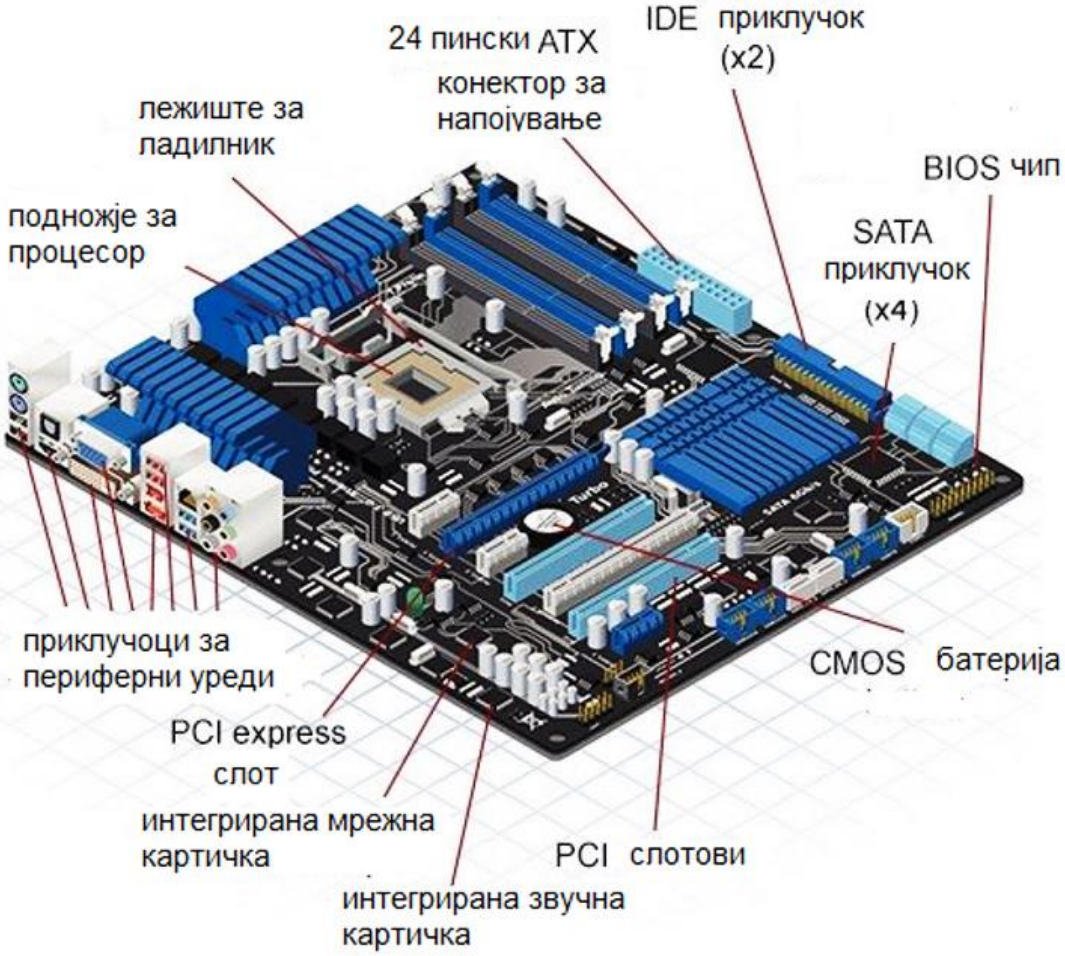


Слика 2.6. Надворешен изглед на RAM меморијата и слотот за мемориската картичка за Raspberry Pi

Во мемориската картичка се чува буквално сè: оперативниот систем, програмите и податоците за обработка. Raspberry Pi поддржува мемориски картички со капацитет од 8GB до 32GB. [3]

2.2. Матична плоча со опис и приклучоци

Матичната плоча ги **поврзува** сите хардверски компоненти во една функционална целина. Компонентите меѓу себе се физички поврзани со бакарни водови наречени магистрали. Денес матичните плочи се изработуваат во осум до десет слоеви од бакарни водови. ATX (анг. Advanced Technology eXtended) е напредна технологија, со проширување, во дизајнирањето на матичните плочи. Постојат три основни големини на матични плочи: ATX (305 mm × 244 mm), Micro ATX (244 mm × 244 mm) и Mini ITX (170 mm × 170 mm). Матичните плочи со поголеми димензии имаат поголем број на слотови и поголем капацитет на RAM-меморијата. Освен бакарни водови, матичната плоча содржи голем број различни конектори и чипови што го контролираат преносот на податоци од еден до друг уред. Накратко ќе се запознаеме со секој од нив.

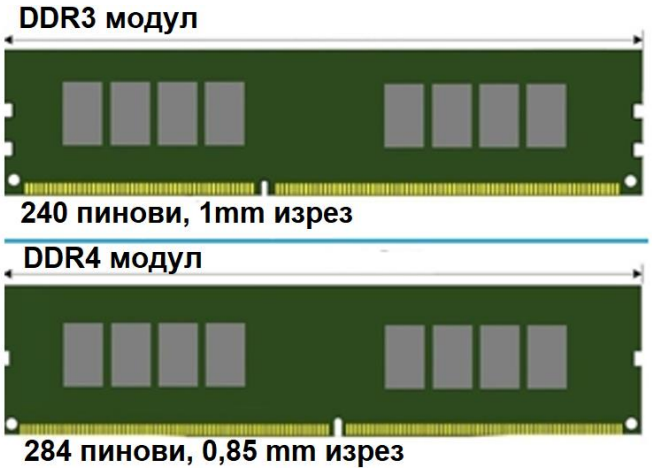


Слика 2.7. Составни делови на матичната плоча

Процесор ⟶ Изборот на матична плоча зависи од изборот на процесор, бидејќи секој тип на процесор има уникатен распоред на пиновите, што бара и соодветно подножје.

Чипсет ⟶ Чипсетот е **множество од специјални интегрирани кола коишто го контролираат протокот на податоци**. Дури и самата матична плоча се именува според видот на чипсетот вграден во неа. Секој чипсет е составен од **две основни компоненти**: северен и јужен мост. Северниот мост е графички и мемориски контролор, тој е многу брз и е директно поврзан со микропроцесорот. Јужниот мост е влезно-излезен контролор и тој комуницира со микропроцесорот, со посредство на северниот мост.

RAM ⟶ Од изборот на матичната плоча зависи видот и капацитетот на RAM-меморијата. Денес најчесто се користат следниве видови на RAM-меморија: DDR4, DDR3, DDR2 и DDR. На пример, не може DDR4 RAM модул да се постави во DDR3 слот. DDRAM (анг. Double Data Rate RAM) е RAM меморија со двојно поголема количина на податоци кои се пренесуваат за времетраење на еден такт. Тие се разликуваат по бројот и распоредот на пиновите, напојувањето и работната фреквенција, односно брзината. Еден од елементите за препознавање на видот на RAM-мемориите е изрезот поставен помеѓу пиновите.



Слика 2.8. Видови RAM-мемории

Графичка картичка ⟶ **PCI Express×16 и AGP** се најдобри конектори за поврзување на графичките картички. Кратенката PCI Express×16 (анг. peripheral component interconnect express) значи експресна конекција на периферни компоненти со пропусен опсег 16 битови по такт. Кратенката AGP (анг. Accelerated Graphics Port) значи графичка порта со забрзување. PCI-слотовите се

со помала брзина и ограничен пристап до меморијата во однос на AGP. Да споменеме дека некои матични плочи имаат вградени графички картички, но добро е таа да содржи и дополнителни слотови, во случај на надградба на видеосистемот.

PCIx1 → PCIx1-слотот се користи за поврзување на **звучни картички, мрежни картички, диск драјв контролери** и други контролери за најразлични периферни уреди.

BIOS

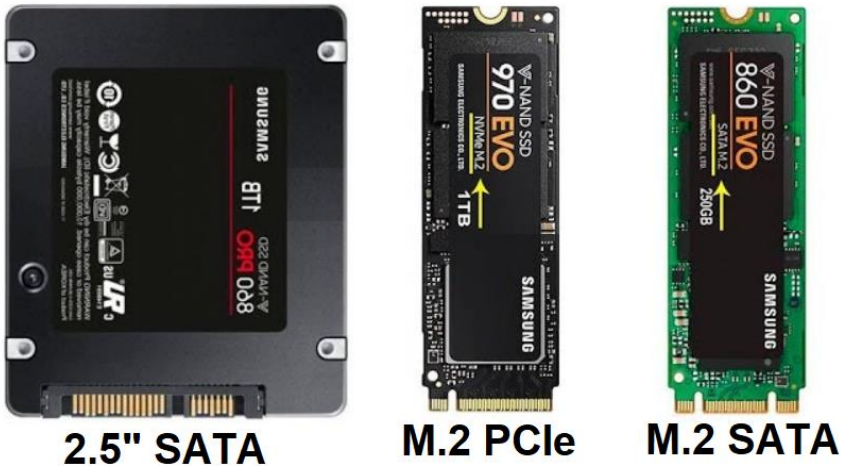
CMOS → BIOS (анг. Basic Input Output System) и CMOS (анг. Complementary Metal Oxide Semiconductor) батеријата се две различни мемории на матичната плоча, кои не можат една без друга, но се разликуваат по своите карактеристики и функција. BIOS-от е ROM-меморија што во себе содржи програма којашто се извршува кога се вклучува компјутерот и од таму доаѓа кратенката што значи основен влез-излез на системот. BIOS-от врши проверка на хардверските компоненти, го лоцира оперативниот систем и го внесува во меморијата, се грижи за сигурноста на системот и врши проверка на лозинките. CMOS-от е RAM-меморија што користи сопствена батерија за напојување. Во CMOS-от се чуваат податоци за времето и датумот и податоци за нагодување на хардверските компоненти при стартување на компјутерот. Кратенката CMOS се користи за опис на полупроводничка технологија за производство на интегрирани кола.

Хард диск

SDD → Поголем број матични плочи содржат два **вида SATA (анг. Serial Advanced Technology Attachment) конектори, SATA2 и SATA3**. Кратенката значи сериска напредна технологија за поврзување. Конекторот SATA2 е со сина боја и се користи за поврзување на хард дискот и во него се приклучува податочниот кабел на хард дискот. Хард дискот поседува уште еден SATA-кабел за напојување и тој се поврзува со единицата за напојување на матичната плоча. Конекторот SATA3 вообичаено е со бела боја и се користи за приклучување на SSD-меморијата.

M.2 SSD модули → Поновите модели на матични плочи содржат специјални конектори со ознака M.2 SSD за поврзување на специфични SSD-модули. Тие се разликуваат од SSD мемориските уреди по физичкиот изглед и конекцијата и ова е прикажано на слика 2.9. Дури и самите SSD-модули можат да бидат со два типа на конекција, PCIe или SATA. Модулот со PCIe конекција има еден изрез поставен меѓу пиновите, а модулот со SATA

конекција има два изрези. Поради тоа при самиот избор на модул треба да се внимава од кој тип е M.2 SSD конекторот на самата матична плоча. M.2 SSD-модулот со PCIe конекција има поголема брзина на пренос.



Слика 2.9. Споредба меѓу SSD SATA мемориски уред и M.2 SSD-модули

Серискатa, паралелната порта ⟶ Серискатa и паралелната порта не се среќаваат кај поновите генерации персонални компјутери, но тие сè уште имаат своја примена. На пример, серискатa порта се користи за поврзување модеми, рутери, RSR-232 уреди, како што се индустриските управувачи. Доколку компјутерот не поседува сериска порта, тогаш потребно е да се користи USB сериски конвертор. Паралелната порта сè уште се користи за поврзување на печатачи.

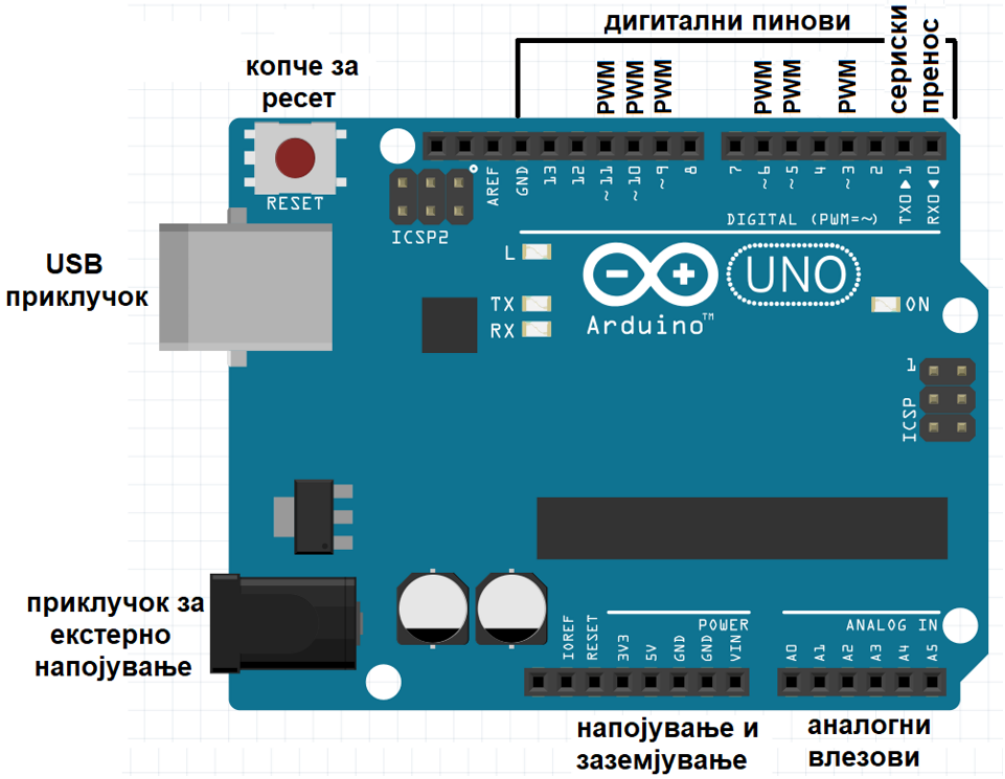
Видео уреди ⟶ VGA (анг. Video Graphics Array) е аналоген видео стандард. Самата кратенка значи видео графичка низа и се однесува на дводимензионалната низа од бои дефинирана со самиот стандард. Се користи за поврзување на монитори, телевизори, проектори и друга видеоопрема. **HDMI (анг. High Definition Multimedia Interface)** е приклучок за видео и звучни уреди и тој со својот квалитет на пренос го потисна VGA-приклучокот. Кратенката значи поврзување на мултимедијални уреди со висока резолуција.

USB-уреди ⟶ USB-приклучоците се најчесто користени компјутерски приклучоци за голем број периферни уреди: глувче, тастатура, плеери, надворешни хард-дискови, камери и фотоапарати, развојни платформи како што се Arduino и Raspberry Pi. Најчесто функционираат на принципот plug and play, што значи софтверот за овие уреди се инсталира автоматски.

2.3. Составни делови на Arduino Uno R3

Во првата тема веќе се запознавме со значењето и примената на Arduino Uno R3 микроконтролерската развојна платформа. Во оваа тема ќе се запознаеме со хардверот: составните делови на оваа платформата, сензорите и атенуаторите. Во следната модуларна единица ќе се запознаеме со софтверот и со развојната средина на платформата Arduino Uno R3.

Основна компонента во Arduino Uno R3 е микроконтролерот ATmega328 од производителот Atmel. Овој микроконтролер е 8-битен чип so 28 пинови, изработен во RISC-технологија. Програмска меморија е со капацитет 32 KB, податочната EEPROM-меморија е со 1 KB и внатрешната RAM-меморија изнесува 2 KB. Максималната фреквенција изнесува 20 MHz. Arduino Uno R3 се поврзува со компјутер преку USB-кабел, при што самата платформа има мини USB-приклучок. Преку овој кабел се пренесуваат програмите од компјутерот во Arduino Uno R3 и истите се впишуваат во програмската меморија на микроконтролерот ATmega 328. Откако ќе се испрограмира, Arduino Uno R3 се исклучува од компјутерот, се вградува во друг електронски уред и се користи за процесно управување.

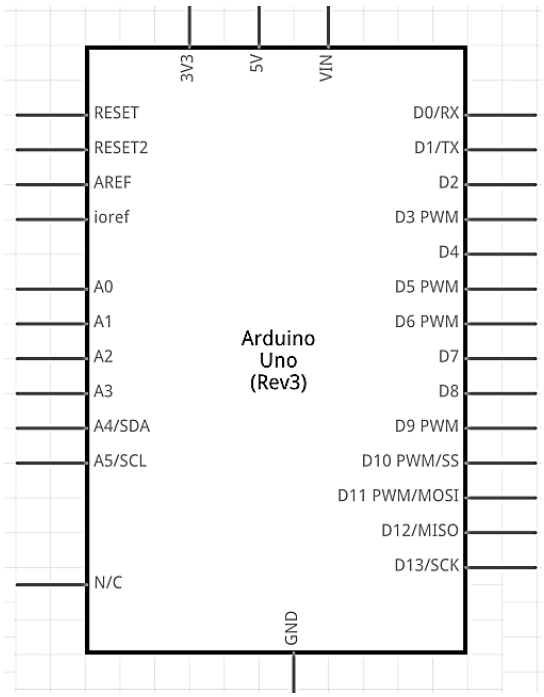


Слика 2.10. Составни делови на Arduino Uno R3 (поглед од горе)

Arduino Uno R3 содржи три сигнални лед-диоди. Лед-диодата со ознака ON е за напојување. Лед-диодите со ознака TX и RX (Transmit-Receive) се покажувачи за сериска комуникација. Овие две лед-диоди поинтензивно трепкаат кога се внесува програма во микроконтролерот. Освен горните лед-

диоди, постои уште една вградена лед-диода,означена со буквата L којашто е поставена на 13-тиот дигитален пин и се користи за проверка на исправноста на хардверската платформа преку извршување на програмата за трепкање (анг. Blink).

Кога е поврзан со компјутер Arduino Uno R3 се напојува преку USB-портата. Во спротивно, може да се напојува преку батерија или надворешен адаптер, со напон од 7 до 12 V.

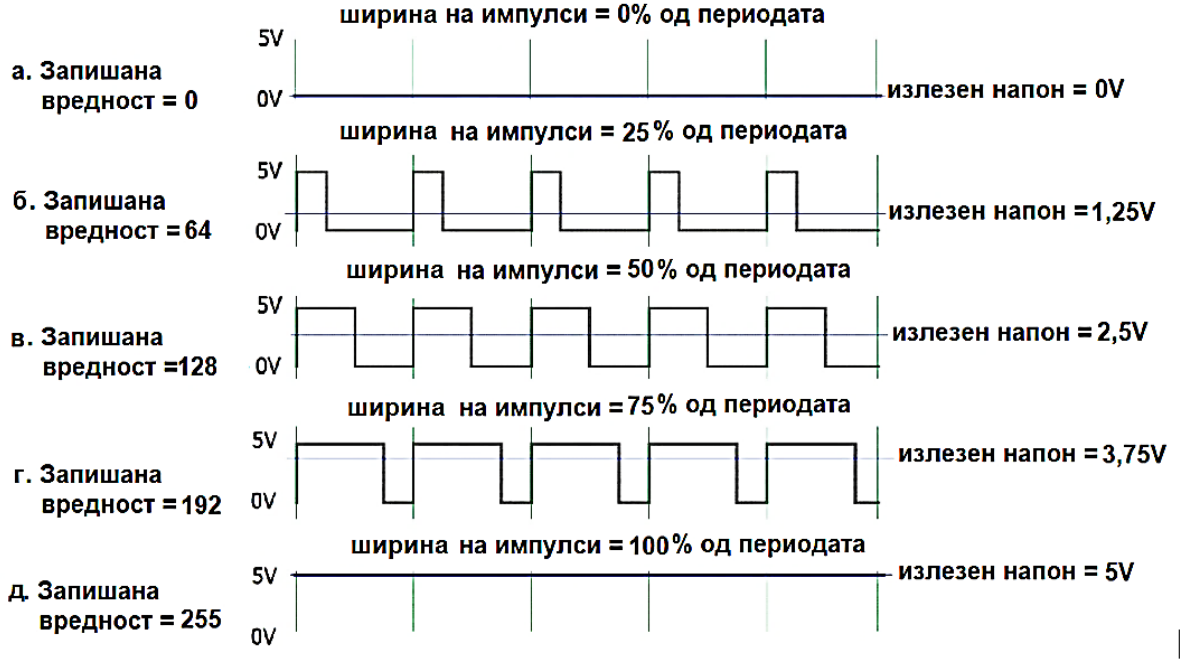


Слика 2.11. Функционална шема на Arduino Uno R3

На слика 2.11. е прикажана функционалната шема на Arduino Uno R3 платформата. Таа располага со **14 дигитални пинови.** Дигиталните пинови можат да бидат влезни или излезни, но не двонасочни. Наједноставен елементи за работа со дигиталните пинови се тастерот како влезен елемент и лед-диодата како излезен елемент. Дигиталните сигнали примаат само две вредности, логичка нула или логичка единица. На пример, кога тастерот ќе биде притиснат, микроконтролерот добива податок логичка единица (се поврзува на напон од 5V), а кога е отпуштен, тоа е состојба на логичка нула (поврзаност со заземјување).

Пиновите со реден број 3,5,6,9,10 и 11 имаат двојна функција. Тие може да се користат како дигитални пинови или аналогни излези. Аналогните излези се обележани со ознаката PWM (анг. Pulse Width Modulation) што во превод значи импулсна-ширинска модулација. Со оваа постапка аналогниот сигнал се претвора во низа од импулси и паузи. При програмирање на Arduino Uno R3 и работа со аналогни излези, програмерот задава (запишува) целобројни вредности во опсегот од 0 до 255. Зависноста на ширината на импулсите од големината на целобројната вредност е прикажана на слика 2.12. . Ширината односно времетрањето на импулсите и големината на излезниот напон зависат правопропорционално од запишаната целобројна вредност. На пример, вредноста 65 претставува 25 % од максималната дозволена вредност 255, затоа ширината на импулсите ќе изнесува 25% од вредноста на периодата и излезниот напон ќе биде 25% од неговата максимална вредност (5V) што е еднакво на 1,25 волти. Импулсно ширинската модулација се користи и кај аналогните влезови, но во обратна насока односно наместо запишување се врши читање. Влезниот напон, кој се движи во опсегот од 0V до 5V, се претвора во поворка од импулси со променлива ширина и ширината на импулсите ја одредува целобројната

вредност што ќе се прочита како влезна големина. Целобројните вредности на влез се движат во опсегот од 0 до 1023 односно во четири помал опсег од оној на излезот. [5]



Слика 2.12. Примена на импулсна ширинска модулација кај аналогни излезни пинови

Исто така, првиот и нултиот пин од Arduino Uno R3 имаат двојна функција. Освен како дигитални пинови истите може да се користат како пинови за сериска комуникација, на пример при поврзување на две Arduino платформи или Arduino платформа со Raspberry Pi.

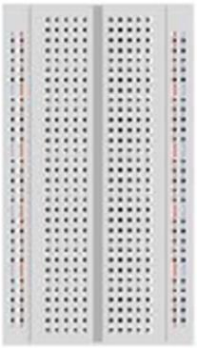
Пинот за заземјување (анг. GND, Ground) и пинот за напон од 5 V се користат за напојување на протоплочка, на која прецизно се поврзани сите електронски компоненти според одредена електрична шема.

2.4. Електронски компоненти за Arduino Uno R3 и нивно поврзување

За Arduino платформата да комуницира со надворешниот свет и да прима информации за средината што го опкружува, потребни се влезни единици: **сензори** (пиезо, тилт-сензор, фотоотпорник, температурен сензор), тастери, потенциометри. За Arduino платформата да делува врз одреден процес, потребни се излезни единици, познати под името актуатори. Во оваа категорија спаѓаат: дисплеи, мотори, зујалици.

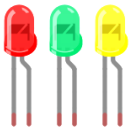
За влезно-излезните единици ефикасно да се поврзат со платформата Arduino Uno R3, потребни се дополнителни електронски компоненти како што се:

отпорници, кондензатори, насочувачки диоди, оптокаплери, насочувачи итн. Накратко ќе ја опишеме нивната функција. [5]



протоплочка

→ Протоплочката е одличен избор за почетници што немаат големи предзнаења од електроника и лемење. Под пластичната плоча, поставени се вертикални и хоризонтални проводници. Во отворите на пластичната плоча се поставуваат изводите на електрични компоненти и жици за нивно поврзување, краткоспојници. Упатството за работа со протоплочка е дадено во практичниот дел на модуларната единица.



Лед-диоди

→ Лед-диодите се наједноставни излезни елементи. За лед-диодата да свети, потребно е подолгиот приклучок (анода) да се поврзе на повисок потенцијал во однос на пократкиот приклучок (катода).

Тастери

→ Тастерите имаат четири приклучоци, по два од две спротивни страни. **Приклучоците од иста страна не се електрично поврзани** и тие треба да се постават во отворите на два различни проводници на протоплочката, за електричното коло да се затвори кога тастерот ќе се притисне.

Оптокаплер

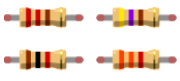
→ **Оптокаплерот** е составен од лед-диода и фотодиода. Низ колото на фотодиодата протекува струја само додека свети лед-диодата. Лед-диодата е поврзана со извор за напојување, а фотодиодата со потрошувач. Со помош на оптокаплерот струјните кола на изворот и потрошувачот се електрично изолирани.

Потенциометар

→ Потенциометарот е променлив отпорник и најчесто е поврзан со еден од аналогните влезови на Arduino Uno R3 со цел да се создаде променлив напон.

Фотоотпорник

→ **Фотоотпорникот** има променлива отпорност, во зависност од светлината што паѓа на неговата површина.



Отпорници

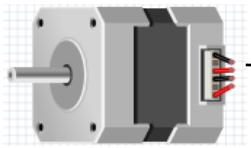
→ Отпорниците се користат за приспособување на јачината на работната струја за да не дојде до оштетување на елементите.

Мотор на еднонасочна струја

Моторот на еднонасочна струја ја претвора електричната енергија во механичка, вршејќи кружни движења. Ако се промени насоката на струјата, ќе се промени и насоката на вртење на моторот.

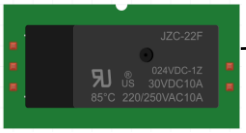
Серво мотор

Серво-моторот не се врти за полн круг, туку само до 180 степени. Во зависност од влезниот напон, тој ќе се помести за определен агол, до одредена позиција, и ќе остане во таа позиција сè додека не се промени влезниот напон. Аголот на вртење зависи од ширината на импулсите по извршената импулсно-ширинска модулација.



Чекорен мотор

Чекорните мотори вршат дискретни поместувања односно со доаѓањето на секој влезен импулс моторот се придвижува за определен агол. Ова поместување се нарекува чекор. Колку чекори се потребни за да се направи полн круг зависи од конструкцијата на моторот (број на полови).



Релеј

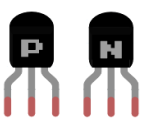
Релеите овозможуваат вклучување или исклучување на потрошувачи со големи моќности преку употреба на нисконапонски или нискострујни сигнали. Постојат два вида на контактни точки, нормално затворен и нормално отворен контакт. Кога е активен релејот, нормално отворените контакти се затвораат и потрошувачот се поврзува со неговото напојување.

Пиезо-компонентата

Пиезо-компонентата може да се користи како сензор за вибрации или генератор на тонови со различна фреквенција.

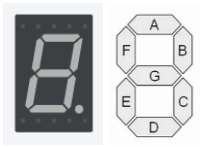
Тилт-сензор

Тилт-сензор е сензор за позиција. Во неговата внатрешност постои метално топче кое се поместува и на таков начин го отвора или затвора контактот односно ги прекинува или ги поврзува приклучоците на сензорот. Во која позиција, хоризонтална или вертикална, ќе биде затворен контактот зависи од типот на сензорот.



Транзистори

Транзисторите се користат како напонски или струјни засилувачи и прекинувачи. Транзисторот 2N2222 е погоден за струи до 500 mA, а транзисторот TIP120 за струи до 5 A.

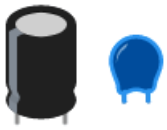


7-сегментен LED екран

7-сегментниот екран се користи за визуелно прикажување на декадни цифри. Седум пинови од вкупно осум се користат за побуда на сегментите, еден пин е за точката и два се заземјување. 7-сегментниот екран може да биде со заедничка анода или заедничка катода и ова е важно за побудување на пиновите.

Насочувачи

H-мостот (анг. H-bridge) е електронско коло кое овозможува промена на насоката низ моторот за еднонасочна струја. Промената на насоката на струјата предизвикува промена на насоката на вртење на моторот. Се изработува од прекинувачки елементи или во облик на интегрирано коло како што е L239D.



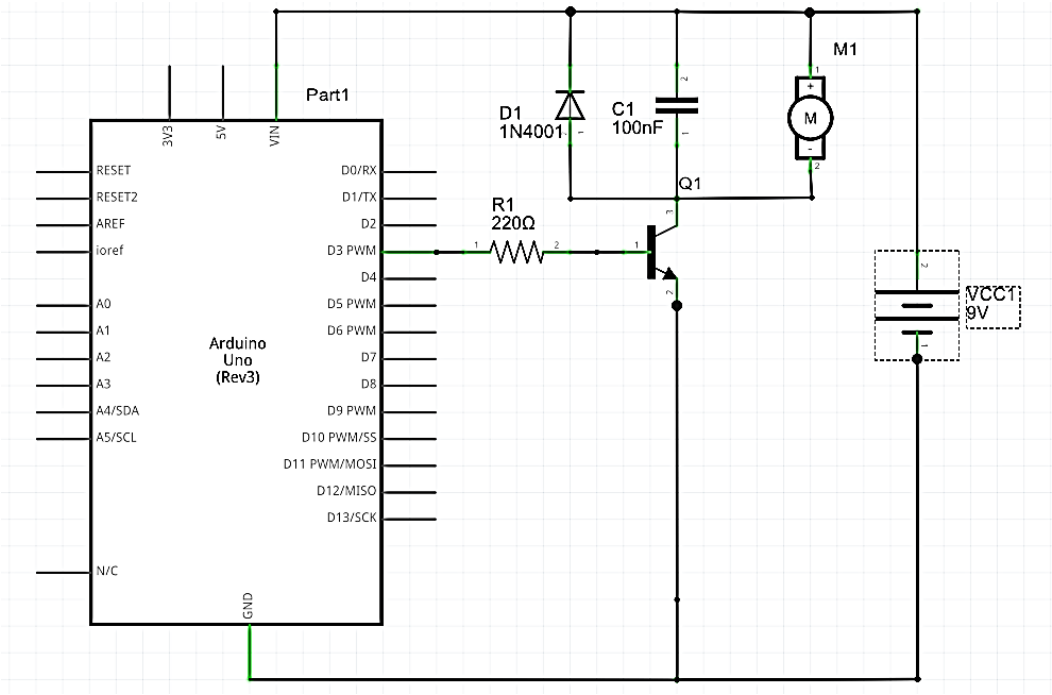
Кондензатори

Ако напонот на кондензаторот е помал од напонот на изворот, тогаш кондензаторот се полни, а во спротивно се празни. Вообичаено кондензаторите се поврзуваат паралелно со сензор или со мотор, со цел да се спречи брза промена на напонот, најчесто предизвикана од други компоненти во колото.

Пред да започнеме со поврзувањето на компонентите на протоплочката потребно е да се изврши анализа на нивните карактеристики и избор на истите. Секој производител на електронски компоненти изготвува техничко-технолошка документација која содржи информации за перформансите, начинот на употреба и карактеристиките, како што е вредноста на минималниот напон кој е потребен за работа на компонентата и максимално дозволениот напон. Некои компоненти како што се отпорникот и обичниот кондензаторот се неполаризирани компоненти и кај нив не е важно кој од изводите ќе биде на повисок, а кој на понизок електричен потенцијал. Диодите, транзисторите, интегрираните кола се поларизирани компоненти и е потребно да се изврши индентификација на изводите.

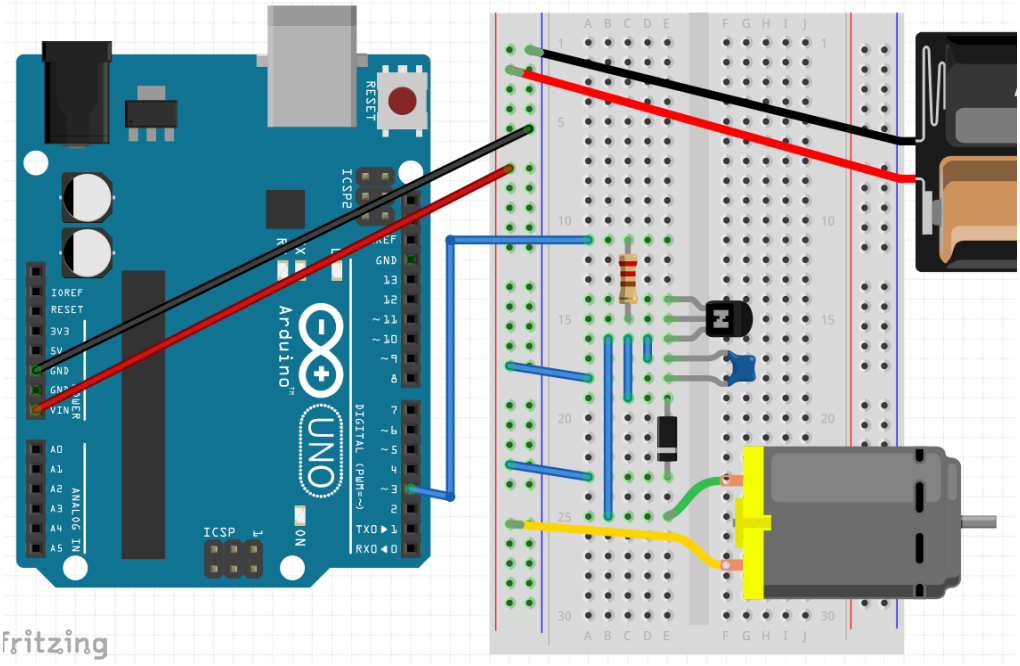
Со Arduino Uno R3 може да се поврзуваат компоненти со работен напон до 5V и максимално дозволена струја од 40mA. Овие вредности не се исти за сите Arduino платформи. На пример, Arduino Uno WiFi Rev2 платформата работи со 20mA максимално дозволена струја, а Arduino Zero со 7mA.

На слика 2.13. и слика 2.14. се прикажани функционалната и монтажната шема за поврзување на мотор на еднонасочна струја со Arduino Uno R3 платформата. Во функционалната шема симболите ги претставуваат електронските компоненти, а линиите начинот на нивно поврзување. Ако електричното коло е едноставно, со мал број на компоненти, можеби монтажната шема е полесна за разбирање. Но, во случај на голем број на компоненти и жици за нивно поврзување тогаш неопходна е функционална шема зошто таа ни нуди поголема прегледност. [6]



Слика 2.13. Функционална шема за поврзување на мотор на еднонасочна струја со Arduino Uno R3

Во шемата прикажана на слика 2.13. транзисторот има функција на прекинувач.



Слика 2.14. Монтажна шема за поврзување на мотор на еднонасочна струја со Arduino Uno R3

Базата на транзисторот е поврзана со третиот пин на Arduino Uno R3 кој се користи како аналоген излез со ширински модулирани импулси и од времетраењето на импулсите зависи големината на напонот за напојување на моторот односно јачината на струјата која тече низ него. Во техничко-

технолошката документација на транзисторот треба да провериме дали максимално дозволениот напон колектор-емитер е поголем од напонот на напојување на електричниот мотор и јачината на колекторската струја треба да биде 25% поголема од струјата што тече низ моторот. Базната струја зависи од колекторската струја и коефициентот на струјно засилување. На пример, ако коефициентот на струјно засилување изнесува 100, а посакуваната колекторска струја изнесува 1A тогаш базната струја ќе изнесува 1A/100=0,01A=10mA. Со помош на Омовиот закон можеме да ја пресметаме вредноста на отпорникот поврзан со третиот дигитален пин. На пример 5 / 0,01 = 500Ω и бидејќи ова не е стандардна вредност избираме 470Ω. Диодата е поврзана паралелно со моторот и ги штити пиновите на Arduino Uno R3 од појава на големи индукциони струи, кои се јавуваат при вклучување и исклучување на моторот. Промената на состојбата на дигиталните пинови може да биде многу брза и во колото се јавуваат осцилации во напонот на напојување и истите се филтрираат со употреба на кондензатор.

Напојувањето на Arduino Uno е уште една карактеристика на која треба да се внимава. Веќе знаеме дека Arduino Uno R3 може да се напојува на два начини, со USB или екстерно напојување. После впишувањето на програмата нема потреба од поврзување на Arduino Uno R3 со компјутер и тогаш може да се користи екстерно напојување, батерија или адаптер кој наизменичниот напон го претвора во еднонасочен со вредност од 7 до 12V. Напони поголеми од 12V може да предизвикаат оштетување на електронската плоча. Напонот на батеријата исто така треба да биде во опсегот од 7 до 12V. Капацитетот на батеријата е даден во mAh. Колку ќе трае батеријата зависи од потрошувачката на електрична енергија на компонентите вградени во електронскиот уред.

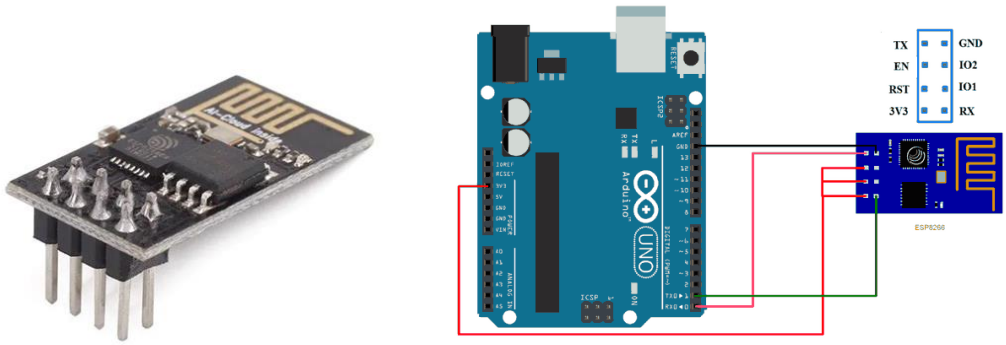
Анализата на шемата за поврзување на Arduino Uno R3 со мотор на еднонасочна струја е само еден пример за тоа како треба да се проучат функцијата и карактеристиките на електронските компоненти пред истите да се поврзат со развојната платформа.

2.5. Додатоци за Arduino платформа

Arduino платформата стана многу популарна меѓу нејзините корисници, за што голем придонес даде и концептот на отворен извор (анг. open source), како на хардвер така и на софтвер. Многу корисници кои експериментираа со дизајнот на своите уреди им дадоа идеја на производителите на електронски компоненти да креираат и пуштат во продажба готови додатоци со кои многу се олесни работата со Arduino и значително се зголеми нивната функционалност. Додатоците се всушност готови електронски плочи со точно определена намена. Поврзувањето е едноставно и најчесто додатоците содржат долги машки пинови кои се поставуваат во женските пинови на Arduino. **Штитовите (анг. Shields)** се додатоци коишто се поставуваат врз основната Arduino-плоча при што двете

плочи се една врз друга, но не се допираат поради долгите машки пинови. Називот штит има симболично име. Бидејќи Arduino софтверот е од отворен тип, корисниците имаат пристап до огромен број на готови програмски кодови за додатоци кои можат да ги менуваат според своите потреби. Исто така поголемиот дел од овие додатоци имаат софтверска поддршка во самата интегрирана развојна средина на Arduino. При користење на додаток за развојна плочка, во програмскиот код мора да се внесе таканаречената библиотека без која додатокот не може да комуницира со Arduino. Библиотеките содржат додатни инструкции со кои многу се олеснува манипулацијата со хардверот. Како што ќе видиме подоцна, големината на додатоците е најразлична, со димензии од 1-2 сантиметри и 4 пинови, до додатоци кои се со иста големина и број на пинови како и самата развојна плоча, какви што се штитовите. Обично додатоците за Arduino платформа не се за почетници и во практичниот дел нема да ги користиме, но добро е да се познаваат можностите кои ги нудат истите. Ќе се запознаеме со еден додаток и пет штитови.

Без интернет пристап или Wi-Fi приклучок Arduino не може да комуницира со надворешниот свет, да испраќа податоци добиени од сензорите или да се управува од далечина. За вмрежување на Arduino потребен е посебен додаток како Wi-Fi додаток или Ethernet штит. Додатокот ESP8266-01 е Wi-fi модул со многу ниска цена на чинење и димензии 25mmx15mm што го прави посебно погоден за вмрежување на Arduino Uno R3 во апликации на платформата Интернет на нештата. **[6]**



Слика 2.15. Надворешниот изглед и монтажната шема за поврзување на ESP8266-01 додаток

Надворешниот изглед и монтажната шема за поврзување на ESP8266-01 додатокот со Arduino Uno R3 е прикажан на слика 2.15.. Тој содржи осум пинови и истите не може да се поставаат во пиновите на Arduino Uno R3 туку се користат жици за поврзување. Додатокот ESP8266-01 нема вграден USB приклучок туку се користи универзална асинхрона сериската комуникација преку двата пина TX и RX. Двата влезно-излезни пинови може да се искористат за поврзување на пример, на сензор. Максималната оддалеченост за Wi-Fi поврзаност изнесува од 100-250 метри. Библиотеката на овој модул не е дел од интегрираната развојна средина на Ардуино туку е потребна дополнителна инсталација.

На слика 2.16. е прикажан штитот Arduino Ethernet 2 кој поддржува осум истовремени конекции. Се поврзува со Arduino Uno R3 преку долги машки пинови и бидејќи пин дијаграмот е ист можно е врз него да се нададе уште еден штит.

Слика 2.16. Arduino Ethernet 2 штит

За поврзување со интернет мрежата се користи RJ-45 приклучок и постои вграден слот за мемориска картичка за чување на документи кои треба да се сервисираат во мрежата.

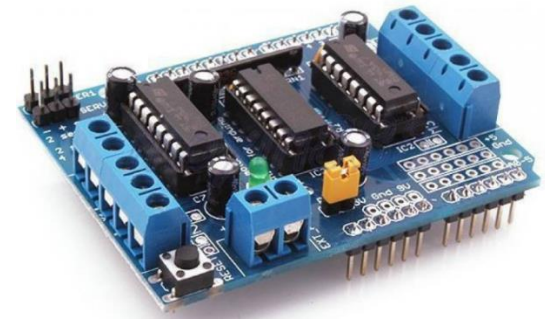
GSM/GPRS Arduino штитот е додаток кој овозможува вмрежување на Arduino во мобилна мрежа.

Слика 2.17. GSM/GPRS Arduino штит

Корисникот може да прима и испраќа SMS пораки, да се јавува и одговара на повици и пристапува до Интернет преку глобалната мобилна мрежа и пакетски ориентираниот сервис за пренесување на податоци (анг. Global System Mobile / General Packet Radio Service).

Штитот располага со вграден слот за SIM картичка, конектори за слушалки, антена, Bluetooth 3.0 со антена и се она што е потребно за да Arduino Uno R3 има функција на мобилен телефон.

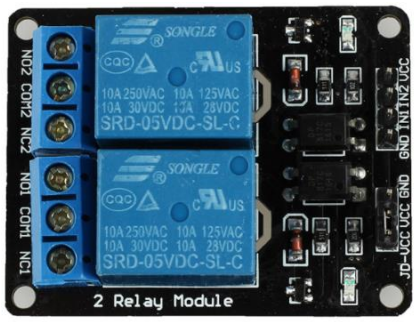
На слика 2.18. е прикажан Arduino штит за мотори, кој може да контролира четири мотори на еднонасочна струја или два серво мотори или 2 чекорни мотори.



Слика 2.18 Arduino штит за мотори

Овие мотори не може да бидат директно контролирани само со Arduino поради јаките струи и големите моќности. Arduino мотор штитот врши струјно одвојување на изворот за напојување на моторот и ја користи логиката на Arduino платформата. Содржи две интегрирани кола L293D и претставува две-канален H-мост за контрола на вртежи на мотор.

Arduino не е дизајниран за контрола на потрошувачи со големи моќности, максималниот напон изнесува 5V, со работна струја до 40mA.



Слика 2.19. Arduino релеј штит

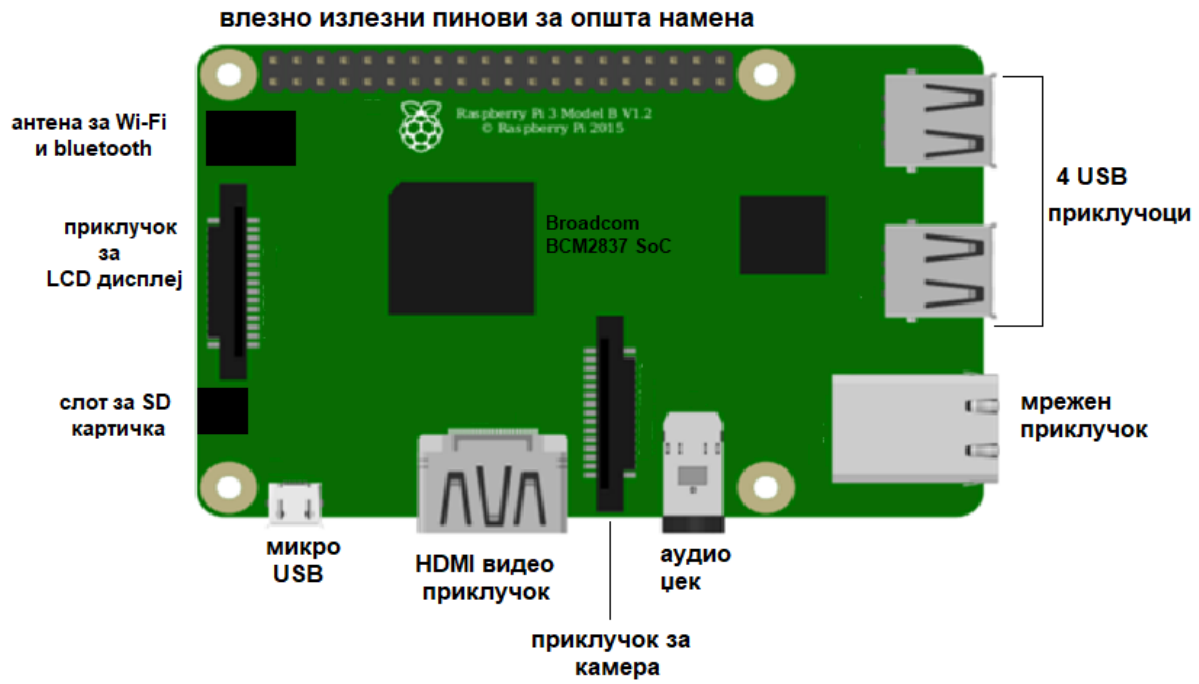
Релеите обезбедуваат галванско одвојување на нисконапонскиот од високонапонскиот дел на системот за управување со моќни потрошувачи. На слика 2.19. е прикажан две-канален Arduino релеј штит со излезен наизменичен напон 250V или еднонасочен напон 30V и максималната струја 10A.

Не е возможно да се набројат сите додатоци кои се користат за проширување на функционалноста на Arduino платформата. Без оглед на изборот секогаш прво треба да се прочита техничко-технолошката документација која ги содржи сите информации за начинот на поврзување и заштита на уредите од евентуална штета.

2.6. Составни делови на Raspberry Pi 3B+

Raspberry Pi е извонреден микрокомпјутер на една плочка со големина на кредитна картичка и многу ниска цена на чинење. Со него можеме да пребаруваме на интернет, играме видеоигри, пишуваме компјутерски програми, креираме електрични кола и др. Ние ќе се запознаеме со моделот Raspberry Pi 3 B+. Во 2019 година излезе новиот модел Raspberry Pi 4 модел B, со многу поголем капацитет на RAM меморијата (1GB, 2GB или 4GB) и со три до четири пати поголема брзина на работа во однос на неговиот претходник. Овој модел е идеален доколку сакаме Raspberry Pi да го користиме како домашен персонален компјутер или за обработка на веб-слики (анг. Computer Vision). За разлика од другите модели Raspberry Pi 4 модел B има проблеми со загревањето, па дури се препорачува и употреба на ладилник. Доколку намената на Raspberry Pi се проекти од областа на електрониката или автоматизација на нашите домови тогаш се препорачува употреба на Raspberry Pi 2 или 3 модел B. [7] Да нагласиме дека сите модели на Raspberry Pi се компатибилни, односно софтверот напишан за еден модел може да го извршува кој било друг модел.

На сликата 2.20. е прикажана третата ревизија на моделот Raspberry Pi 3 B+. Како и секој друг компјутер, така и Raspberry Pi е составен од повеќе компоненти, од кои најважен е системскиот чип Broadcom BCM2837 SoC (System on Chip). Тој во себе содржи: 64-битен ARM Cortex-A53 четиријадрен процесор со работна фреквенција 1,4 GHz, 32 KB L1 кеш-меморија, 512 KB L2 кеш-меморија и графички процесор VideoCore IV. RAM-меморијата не е интегрирана на ист чип со процесорот, како што тоа беше случај со Arduino Uno R3. Таа е посебен чип на развојната плоча, со максимален капацитет од 1 GB. RAM-меморијата не можеме да ја видиме на долната слика бидејќи се наоѓа на задната страна. [3]



Слика 2.20.Составни делови на Raspberry Pi 3B+ (поглед од горе)

Под металното капаче со изгравираното лого на Raspberry Pi се наоѓа радиопредавател. Тој се користи за вмрежување на Raspberry Pi во локални компјутерски мрежи и во интернет-мрежата преку Wi-Fi или за поврзување со други паметни уреди (сензори, мобилни телефони) преку Bluetooth. Мрежниот и USB-контролерот е чип одговорен за пренос преку **мрежната порта и четирите USB-приклучоци**. Во непосредна близина на мини USB-приклучокот се наоѓа еден помал чип којшто се грижи за напојувањето на сите компоненти на развојната плоча.

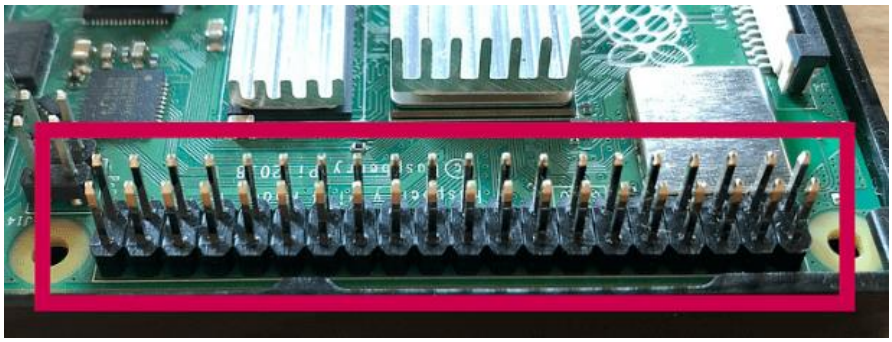
Најголема предност на Raspberry Pi во однос на Arduino е неговата поврзливост, односно големиот број на приклучоци. Да се потсетиме дека Arduino Uno R3 имаше само еден мини USB-приклучок. Raspberry Pi има четири USB 2.0 и едно микро USB-приклучоци. На слика 2.20. се гледаат само два USB 2.0 приклучоци, но всушност се четири, од кои два се поставени еден врз друг. Овие приклучоци се користат за поврзување со голем број надворешни уреди: тастатури, компјутерски глувчиња, дигитални камери, екстерни хард-дискови и други мемории. Микро USB-приклучокот се користи за напојување, исто како што се напојуваат мобилните телефони. Овој приклучок може да се користи и за поврзување со компјутер за пренос на податоци.

Освен преку Wi-Fi, пребарувањето на интернет може да се врши и преку мрежната порта. За таа цел ни е потребен кабел со RJ45-приклучок. Другиот крај на кабелот се поврзува со рутер. Во долниот дел на мрежната порта има две лед-диоди што служат како покажувачи за успешна комуникација.

Како што кажува и самиот назив **HDMI** (анг. High Definition Multimedia Interface), овој приклучок нуди најквалитетен аудио и видео пренос. Преку него, Raspberry Pi се поврзува со монитор или со телевизор. 3,5-милиметарскиот AV

(анг. Audio Video) приклучок пред сè е наменет за поврзување со слушалки или со аудиозасилувач. Тој може да се користи и како видеоприклучок, но тогаш треба да се набави посебен адаптер.На самата развојна плоча постојат два **специјални конектори за** поврзување специјално дизајнирани Raspberry Pi модули. Едниот е познат под кратенката CSI (анг, Camera Serial Interface) и служи за поврзување на камера, а вториот DSI (Display Serial Interface) за поврзување на сензорен екран(анг. touch screen).

Raspberry Pi располага со приклучок со **40 пинови за поврзување со влезно-излезни уреди со општа намена.** Кратенката за овој приклучок е GPIO (анг. General Purpose Input Output). Дваесет и шест пинови од овој конектор се за поврзување со влезно-излезни уреди, а останатите се за заземјување и напојување, од 3,3 V и 5 V. Под влезно-излезни уреди подразбираме: лед-диоди, тастери, сензори, мотори и друго. Со нивната функција и значење се запознавме кога говоревме за составните делови на Arduino Uno R3. Се разбира, протоплочката е наједноставно решение за поврзување на овие уреди.

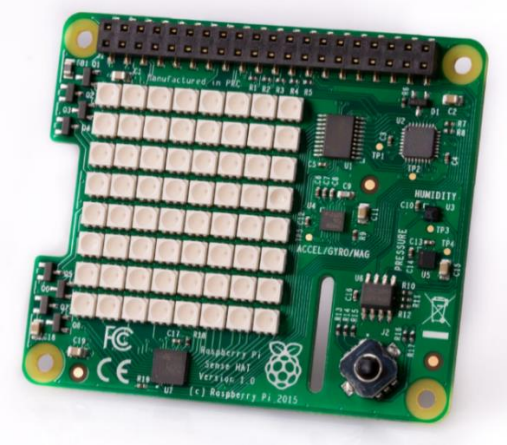


Слика 2.21. GPIO-пинови за поврзување на влезно-излезни уреди

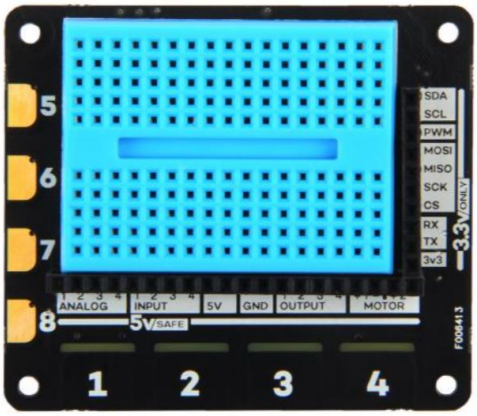
2.7. Додатоци за Raspberry Pi

Додатоците за проширување на функционалноста на Raspberry Pi се познати под името хардвер прикачен одозгора (анг. HAT-Hardware attached on top). Најчесто за поврзување се користат сите 40 пинови за општа намена на Raspberry Pi, но сепак повеќето додатоци дозволуваат пиновите да ги користат и други уреди, па дури е дозволено поврзување на повеќе додатоци еден врз друг. Најголем број додатоци се поврзуваат по принципот поврзи и работи (анг. Plug and play) односно автоматски се конфигурираат. [3]

На слика 2.22. е прикажан еден од најпопуларните додатоци, додатокот со сензори и дисплеј (анг. Sense hat). Тој во својот состав содржи 8x8 LED матричен дисплеј, мал џојстик со пет копчиња и шест интегрирани сензори: жироскоп (сензор за аголна брзина), акселерометар (линеарен сензор за забрзување), магнетометар (преку мерење на земјиното магнетно поле, магнетометарот го одредува правецот на географскиот север), барометар (сензор за притисок), температурен сензор и сензор за релативна влажност.



Слика 2.22. Додатокот за Raspberry Pi со сензори и дисплеј



Слика 2.23. Додатокот за истражување на Raspberry Pi

Додатокот за истражување, прикажан на слика 2.23, се користи во роботиката пред сè поради можноста за контрола на мотори на еднонасочна струја. Овој додаток е компатибилен и со Arduino микроконтролерските платформи. Додатокот за истражување содржи четири влезови и излези со максимална вредност на напонот од 5V, четири капацитивни копчиња на допир, четири копчиња за поврзување на штипки крокодилки, четири аналогни влезови, два H-мостови за контрола на вртење на мотори на еднонасочна струја и мини протоплочка која се поставува одозгора.

Raspberry Pi може да се користи и како мерен инструмент благодарение на додатокот за мерење и собирање на податоци (анг. DAQ MCC 128 - Data Acquisition Measurement Computing).

Слика 2.24. Додаток за мерење и собирање на податоци

Оваа електронска плочка содржи осум аналогни влезови за мерење на напон. Постојат два начини за мерење на влезниот напон. Кај мерењето со еден крај (анг. Single ended) се мери влезниот напон во однос на нултата потенцијална точка, заземјувањето. Диференцијалното мерење ја дава разликата меѓу два влезни напони. Во една секунда се земаат 100К примероци. Аналогниот дигитален претворувач има 16 битна резолуција.

Може да се наредат осум додатоци за мерење на напон еден врз друг и во тој случај се добива 64 податочни канали со максимален пропусен опсег 320К примероци во една секунда.

2.8. Практични вежби за модуларната единица: Составни делови на микрокомпјутери

2.8.1. Мерки за заштита и безбедност при работа

Учениците самостојно ги извршуваат практичните вежби во групи, не повеќе од 2-3 ученици, под надзор на предметен наставник. Заради ефикасно и безбедно изведување на вежбите како и правилно ракување со електронската опрема, учениците треба да се придржуваат кон следните правила.

- Поврзувањето на електронските компоненти се извршува исклучиво во безнапонска состојба.
- Електричната опрема не се допира со влажни или мокри раце.
- Не е дозволено истовремено со едната рака да се држи проводник, инструмент или друг елемент под напон, а со друга рака друг проводник.
- Алатите за работа треба да имаат изолирани дршки.
- Пожар предизвикан од електрична струја не се гаси со вода.
- По исклучувањето од струјното коло електролитските кондензатори треба да се испразнат со кусо спојување на нивните изводи
- При работа под напон да не се допираат спроводници, отпорници, батерии поради нивно загревање.
- Електронскиот отпад да се собира на посебно предвидено место.
- Пред почетокот на секоја практична вежба ученикот треба внимателно да ги прочита барањата за изведба на вежбата и да повтори за начинот на работа на електронските компоненти кои се користат во истата.
- Електронските компоненти да се поврзуваат внимателно, пополека и без употреба на сила.
- Жиците за поврзување на електронските компоненти да не се испреплетуваат и премногу да се затегнуваат
- По поврзувањето на електронските компоненти во електрично коло да се повика наставникот за да изврши проверка, по чие одобрение се вклучуваат потребните напони.
- За сите дефекти, оштетувања или недостатоци на компонентите или приборот веднаш да се извести предметниот наставник.

2.8.2. Практична вежба за инсталација на составни делови на персонален компјутер

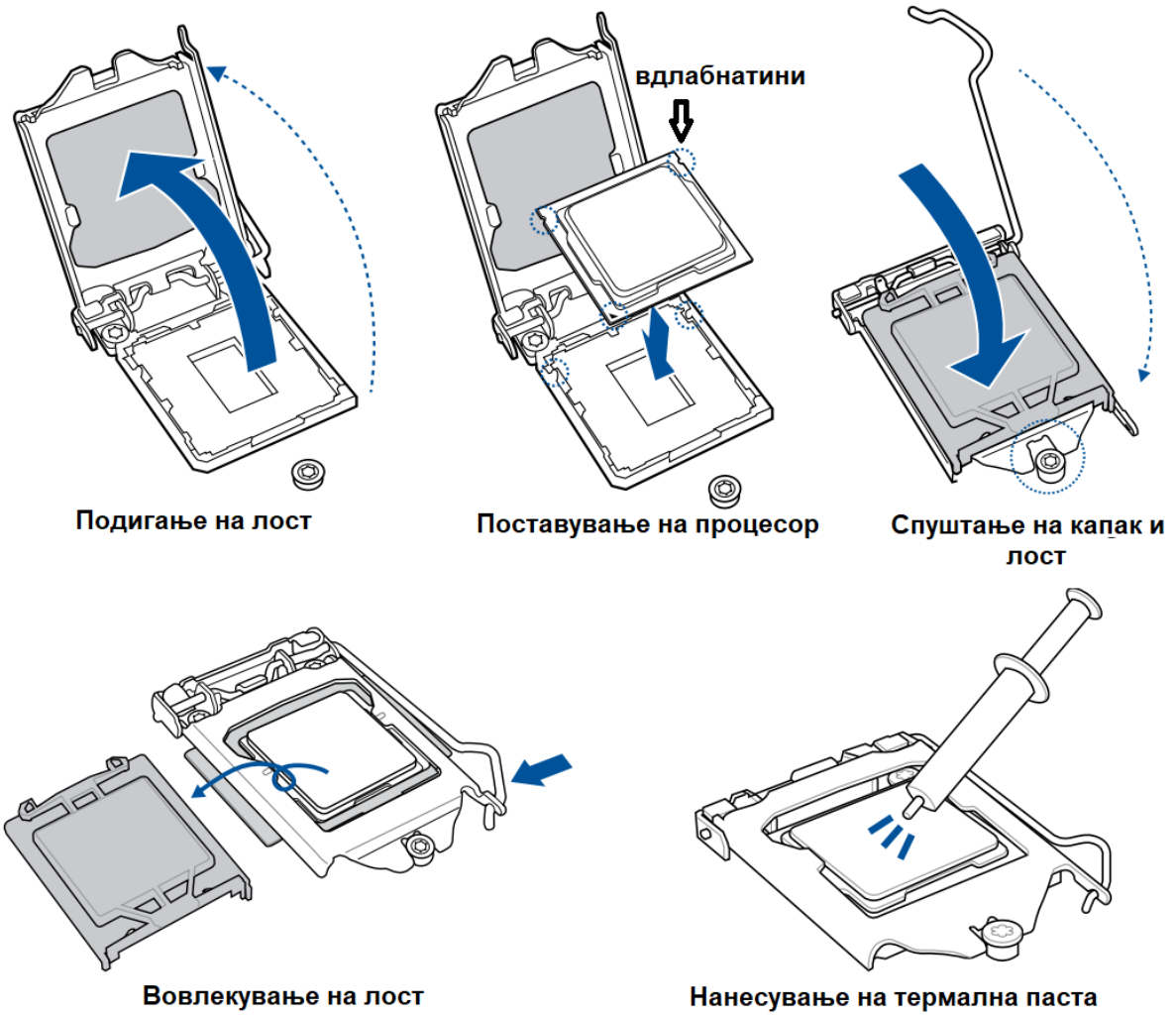
2.8.2.1. Мерки за заштита и безбедност при работа со хардверски компоненти на персонален компјутер

- Со цел да се спречи појава на струен удар пред да започне инсталацијата да се исклучи напојувањето на компјутерот.
- Да се провери исправноста на сите кабли и конектори пред да се вклучи напојувањето.
- Да се користи уред за напојување со соодветна сертификација од самиот производител.
- Задолжително да се прочита техничко-технолошката документација за конкретниот модел на матична плоча бидејќи може да постојат извесни разлики во параметрите и обележувањето.
- Бидејќи електронските чипови се осетливи на статички електрицитет потребно е да се допре некаква метална површина, на пример самото куќиште. На таков начин доаѓа до празнење на човековото тело.
- За да се избегне кратко спојување да не се допираат конекторите и подножјата со метални предмети како завртки или шрафцигери.
- Инсталацијата на хардверските компоненти да се изведе на неподвижна и стабилна работна површина.
- Компјутерот треба да се заштити од влага и нечистотија бидејќи таа штетно влијае врз контактите на матичната плоча и го намалува ладењето и вентилацијата, па може да дојде до презагревање.
- Корисникот треба правилно да го исклучува компјутерот за да се спречи евентуално оштетување на неговите делови.

2.8.2.2. Инсталација на процесор

Доколку се работи за нова матична плоча, подножјето на процесорот треба да биде покриено со жолта заштитна фолија. Од десната страна на подножјето се наоѓа **метален лост**, кој треба да се притисне надолу, да се повлече надесно и да се крене нагоре, при што ќе се подигне и капакот на подножјето. Жолтата заштитна фолија се вади и таа треба да се сочува во случај да се извади процесорот. Процесорот треба да се држи отстрана и **да не се допираат златните пинови.** Процесорот го поставуваме на подножјето, со

пиновите свртени надолу, при што внимаваме двете странични вдлабнатини на процесорот да се совпаднат со двете испакнатини на подножјето. [8]



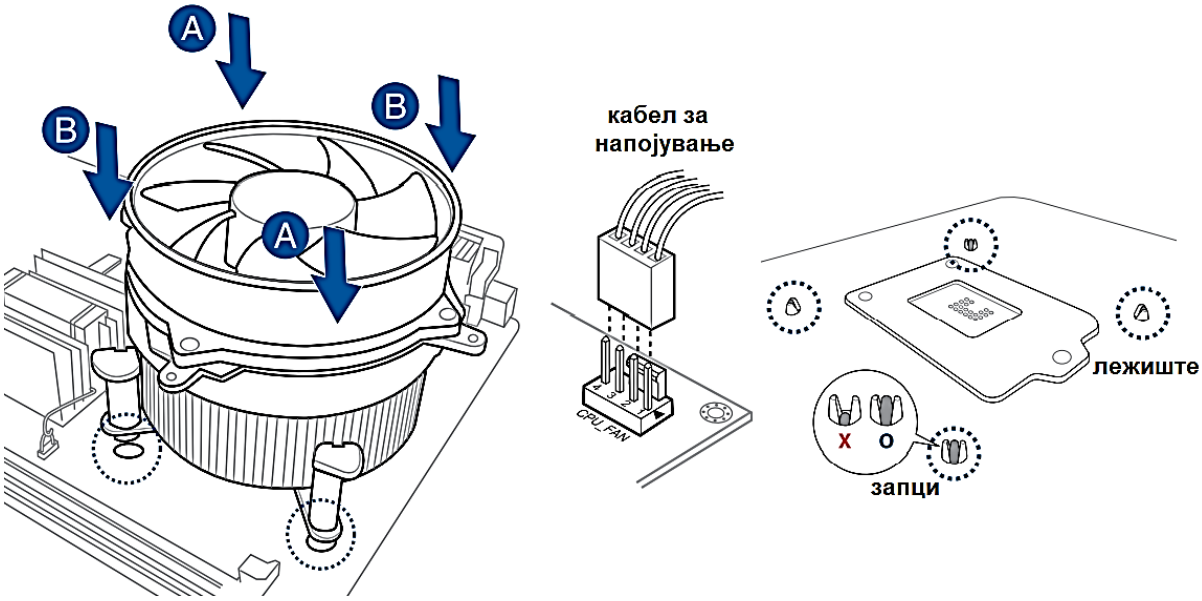
Слика 2.25. Чекори при инсталација на процесор

Процесорот не смее да се притиска. Се спушта капакот до навртката, се спушта металниот лост и тој се вовлекува во лево, за да се намести во своето место.

2.8.2.3. Инсталација на ладилник

Пред инсталацијата на ладилникот треба да провери дали има потреба од нанесување на **термална паста** врз горниот дел на процесорот. Термалната паста ја намалува температурата од 20˚ до 30˚. Пред да се нанесе нова паста, треба да се отстрани старата и за таа цел, најдобро е да се користи 90% алкохол и микрофибер крпа. Процесорот треба претходно да се извади од подножјето. На крпата се нанесува мало количество алкохол, колку големина на поштенска марка, и потоа нежно се поминува горниот дел на процесорот. Пастата се нанесува на средината, колку половина од зрно грашок. Пастата ќе се размачка

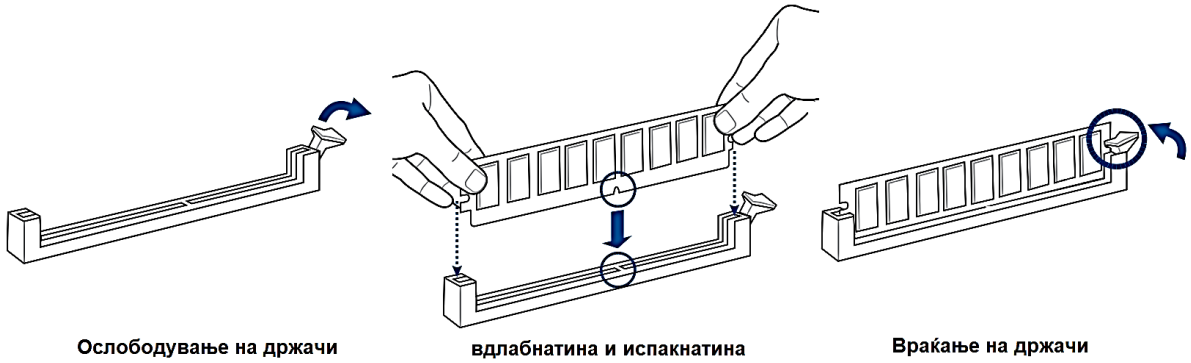
под дејство на топлината на процесорот и тежината на ладилникот. Пред да го поставиме ладилникот врз процесорот, проверуваме дали жлебовите на главите **од пластичните запци** се насочени кон центарот на ладилникот. Избираме два запци што се дијагонално поставени и ги притискаме. Откако ќе ги притиснеме сите четири запци, од задната страна на матичната плоча проверуваме дали запците се израмнети со своите лежишта.



Слика 2.26. Чекори при инсталација на ладилник за процесор

2.8.2.4. Инсталација на RAM-меморија

Слично како кај процесорот, и кај RAM-меморијата треба да внимаваме на статичкиот електрицитет. Затоа, најдобро е пред инсталацијата да го допреме барем металното куќиште. Исто така, RAM-модулите треба да се држат странично и никако да не се допираат златните пинови.

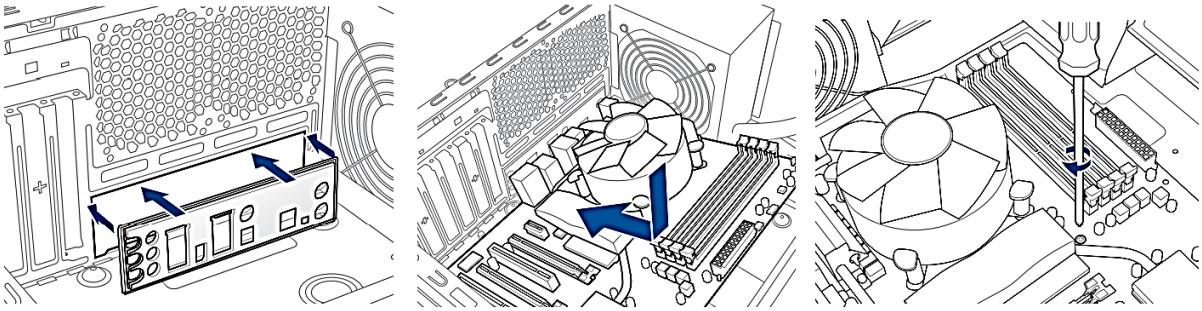


Слика 2.27. Чекори при инсталација на RAM-меморија

Помеѓу пиновите се наоѓа **мала вдлабнатина** и таа вдлабнатина треба да се совпадне со испакнатината на DIMM-слотовите (RAM-подножјето). Пред да го вметнеме меморискиот модул, потребно е да се ослободат **страничните држачи** на слободниот слот.

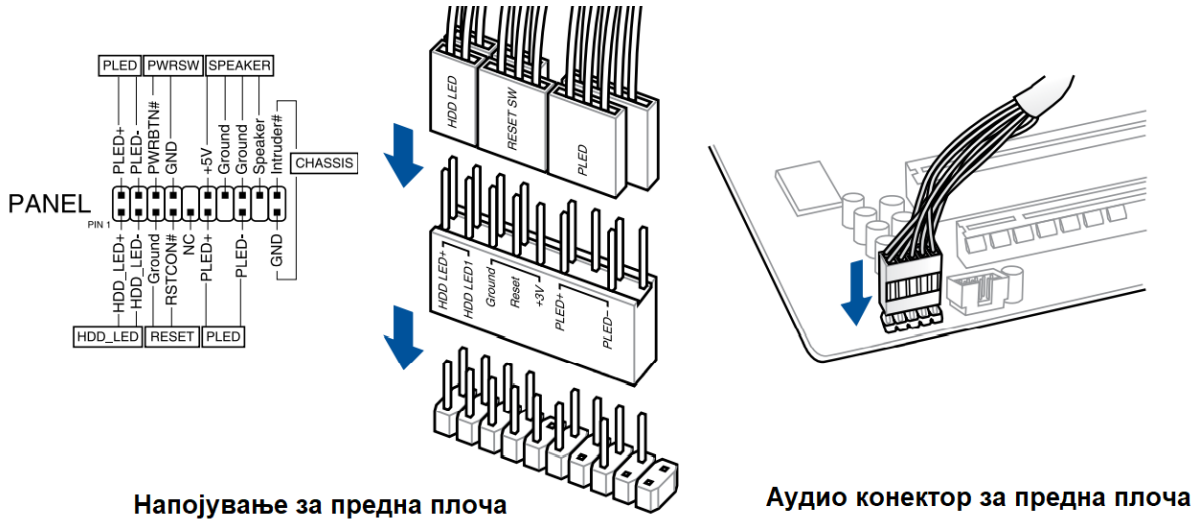
2.8.2.5. Монтажа на матична плоча во куќиште

Најдобро е процесорот, ладилникот и RAM-меморијата да се инсталираат пред матичната плоча да се монтира во куќиштето. Матичните плочи доаѓаат во комплет со **метална плоча** на којашто има отвори за приклучоците на задната страна од куќиштето. При поставувањето на металната плоча треба да се внимава нејзините отвори да одговараат на приклучоците на задниот дел од матичната плоча. Потоа треба да се лоцираат **навртките** во куќиштето и матичната плоча се поставува така што отворите за завртките на матичната плоча да се совпаднат со завртките на куќиштето.



Слика 2.28. Чекори при монтажа на матична плоча

Откако ќе ги зашрафиме навртките, потребно е да се поврзат **приклучоците за напојување** на предната плоча и приклучокот за напојување на вентилаторот на самото куќиште.



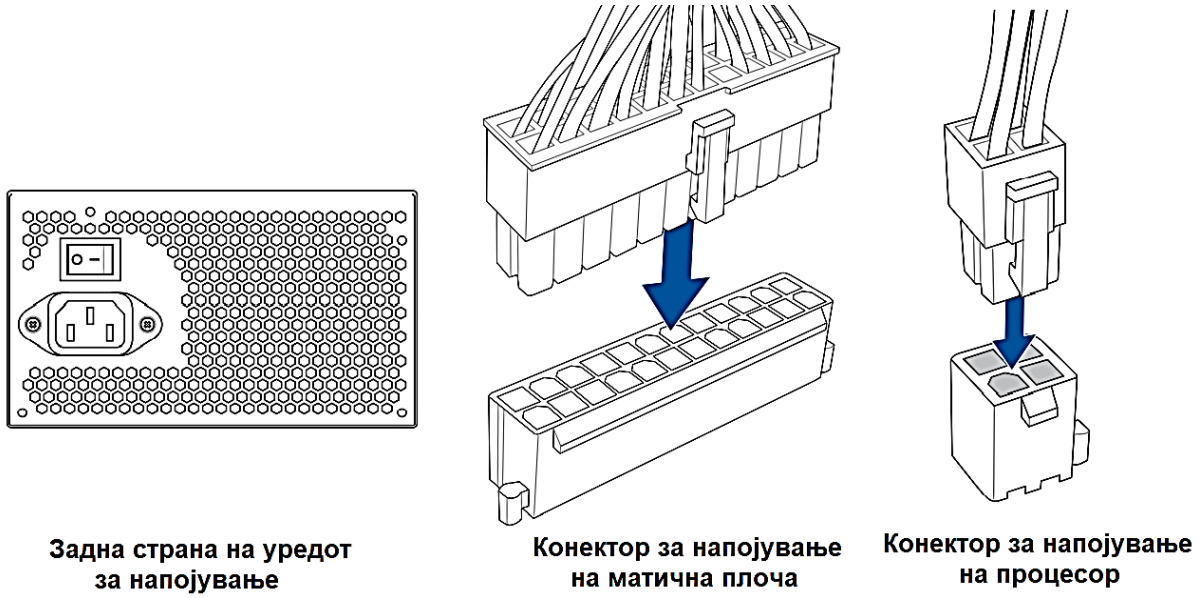
Слика 2.29. Поврзување на матичната плоча со напојувањето

На предната плоча од куќиштето има копче за напојување, копче за ресетирање, предни USB-порти, предни аудио конектори и LED-диоди како индикатори за напојувањето и за хард дискот. Нивните приклучоци за

напојување треба да се поврзат со подножјето на матичната плоча, кое е обележано со буквата **F (Front)**. USB и аудио конекторите се приклучуваат наједноставно поради уникатниот распоред на нивните пинови. Кај приклучоците на копчињата за напојување и за ресетирање не е важен поларитетот. Кај LED-диодите треба да се внимава на знаците. Вообичаено, црната и белата жица се плус, а жиците во другите бои се минус односно заземјување. На крајот, на матичната плоча треба да се лоцира подножјето со ознака **SYS_FAN и PWR_FAN** и на него да се постави приклучокот за напојување на вентилаторот на куќиштето.

2.8.2.6. Инсталација на уред за напојување

Уредот за напојување може да се постави во горниот или во долниот дел, на задната страна на куќиштето. Го внесуваме уредот во внатрешната преграда, со вентилаторот свртен кон матичната плоча, го туркаме до задниот дел и го прицврстуваме со навртки.

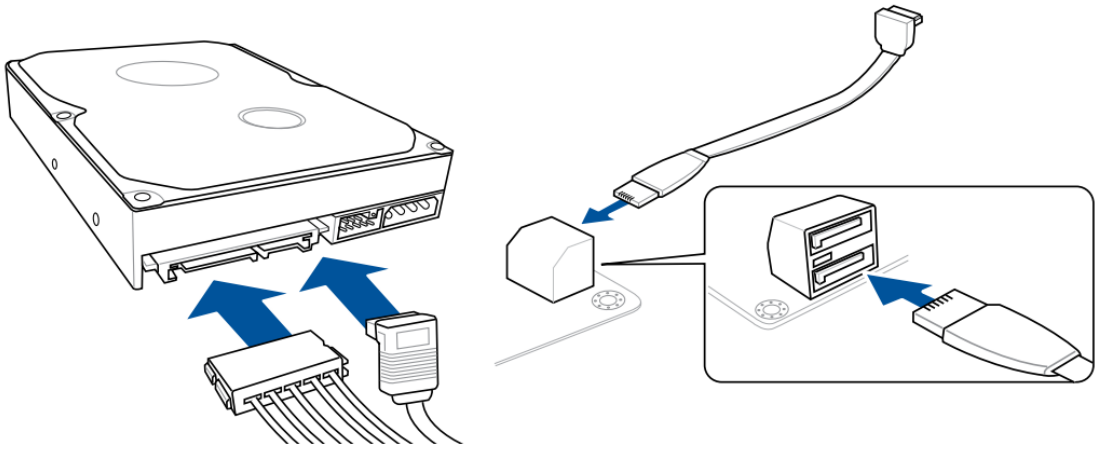


Слика 2.30. Инсталација на уред за напојување

Уредот за напојување има повеќе кабли што треба да се поврзат со матичната плоча или со одредена компонента. Добро е што сите кабли за напојување имаат уникатен дизајн, па не може да се згреши при нивното поврзување. На матичната плоча се поврзуваат два конектори: **ATX-кабелот со 20+4 пина**, кој ја напојува матичната плоча, **и ATX 12 V со 4 пина**, кој го напојува процесорот. За напојување на хард дискот и DVD-уредот се користат два вида кабли: **SATA и IDE со 4 пинови, и нив ќе ги објасниме наскоро.** Шест-пинскиот кабел за напојување на графичката картичка е обележан со ознаката VGA1 и тој е познат под името PCI Express кабел.

2.8.2.7. Инсталација на хард-диск и SSD уред

Во внатрешноста на куќиштето постои посебна преграда со висина од 9 cm. Во неа се вметнува хард дискот, при што напред треба да се гледаат неговите приклучоци во форма на буквата L. Во поголемиот приклучок се прицврстува **кабелот за напојување** од уредот за напојување, а во помалиот приклучок се поставува **SATA податочниот кабел**, кој доаѓа во комплет со хард-дискот. Другиот крај на SATA податочниот кабел се приклучува на синиот конектор на матичната плоча обележан со SATA2 во форма на буквата L.



SATA кабел за напојување и податочен кабел

Поврзување на каблите со приклучокот на матичната плоча

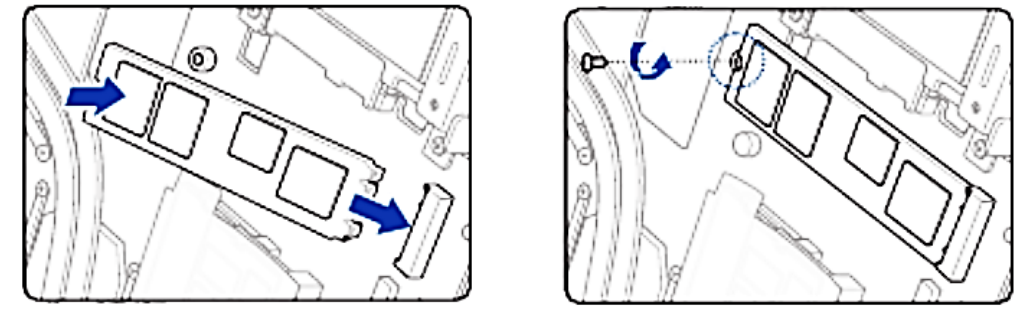
Слика 2.31. Поврзување на хард-диск со SATA-кабли

Бидејќи SATA SSD мемориските уреди се со иста големина и распоред на пинови како и хард-дисковите, за нивно поврзување се користат истите SATA конектори за напојување или пренос на податоци. Се разбира претходно треба да се провери дали во куќиштето постојат две прегради за SSD и хард-диск. Најчесто оперативниот систем се инсталира во SSD-меморијата, а хард-дискот служи за чување кориснички документи и апликации. На ваков начин се скратува времето потребно да се отвори оперативниот систем и се подобрува брзината на работа на персоналниот компјутер.

2.8.2.8. Инсталација на M.2 SSD модул

При инсталацијата на M.2 SDD-модулот треба да го провериме типот на конекција на истиот, PCIe или SATA. Доколку модулот и конекторот на матичната плоча немаат иста конекција истите не можат физички да се поврзат (слика 2.9). Под мал агол, пиновите на M.2 SSD-модулот внимателно се втиснуваат во соодветниот конектор. Откако модулот ќе се постави на матичната плоча истиот се прицврстува со помош на навртка. Да напомнеме дека постојат три лежишта за навртки во близина на M.2 конекторот бидејќи M.2 SSD модулите можат да

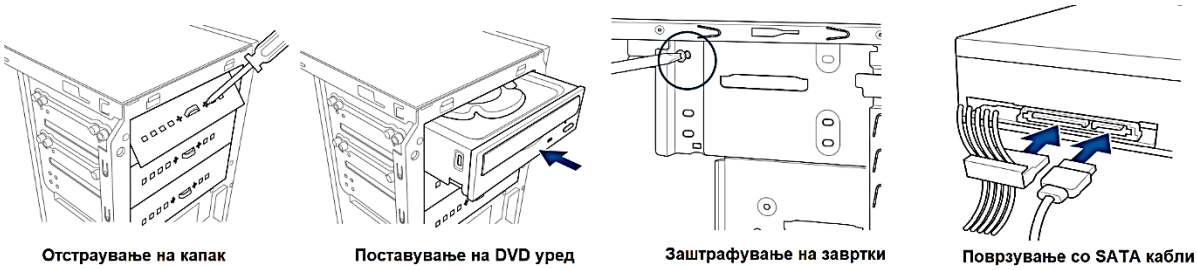
бидат со три различни должини(22, 60 или 80 mm) додека ширината е секогаш иста.



Слика 2.32. Инсталација M.2 SDD-модул

2.8.2.9. Инсталација на DVD-уред

DVD-уредот се монтира на предната страна од куќиштето. За таа цел, потребно е да се извади предната **маска** на куќиштето, која е прицврстена со два запци вметнати во отворите на матичната плоча. Потоа **треба да се отстрани капакот** на 14 cm високата преграда. DVD-уредот се поставува во преградата и од страна се прицврстува со навртки.

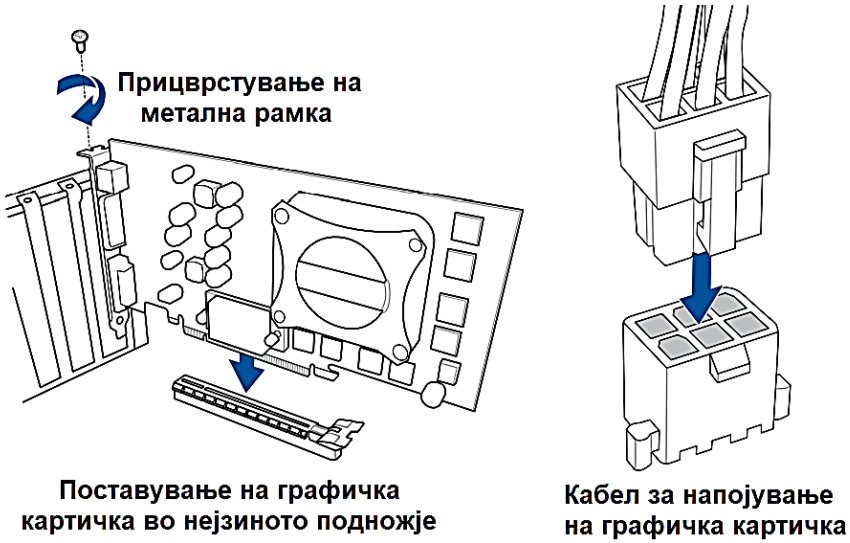


Слика 2.33. Инсталација на DVD-уред

Исто како и хард дискот, и DVD-уредот има два SATA-приклучока во форма на буквата L, еден за напојување и вториот податочен.

2.8.2.10. Инсталација на графичка картичка

Графичката картичка се поставува на **PCI Expressx16** слотот на матичната плоча. Пред почетокот на монтажата, потребно е да се отстранат металните рамки на задната страна на куќиштето, со одвртување на навртките. Откако ќе се вметне графичката картичка во PCI Expressx16 слотот, потребно е да се прицврсти **металната рамка** на самата графичка картичка, на задниот дел од куќиштето, со помош на навртки. На крајот го прицврстуваме шестпинскиот кабел за напојување, обележан со ознаката VGA1.



Слика 2.34. Поставување на графичка картичка во PCI Expressx16 слотот

2.8.3. Практични вежби за поврзување на електронски компоненти со Arduino Uno R3

2.8.3.1. Мерки за заштита и безбедност при работа со Arduino-базирана микроконтролерска платформа

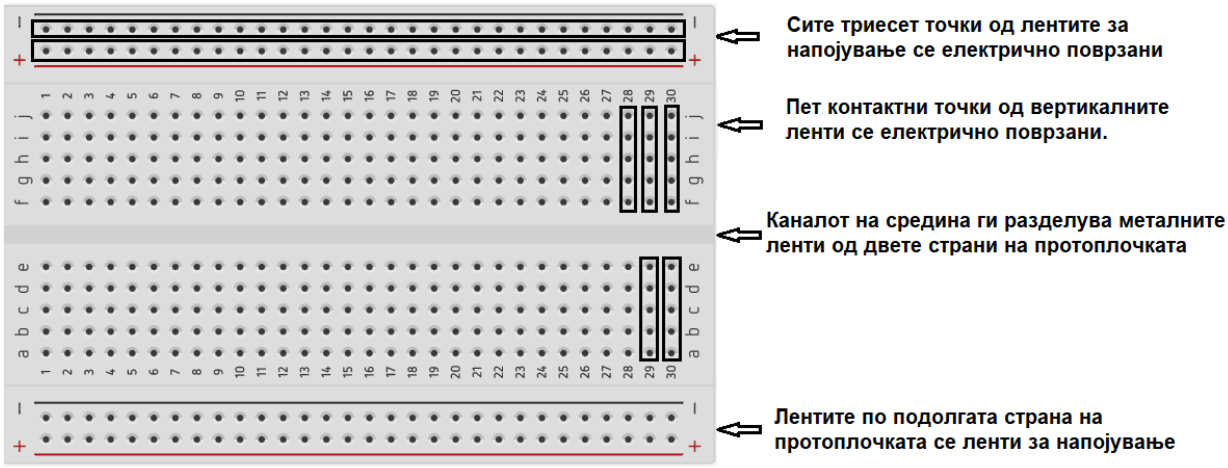
Практичните вежби се изведуваат со примена на Arduino Uno R3 платформата која работи со нисконапонски и нискострујни сигнали. Со цел истите да се заштитат од оштетување треба да се преземат одредени мерки на претпазливост.

- Пред почетокот на секоја монтажа треба да се исклучи Arduino Uno R3 платформата од нејзиниот извор за напојување.
- Батеријата, како извор за напојување, се поврзува со Arduino Uno R3 преку пиновите V_{in} и GND. Максимално дозволената вредност на напонот изнесува 20 V, но се препорачува употреба на батерии со напон од 9 до 13 V. При поврзувањето да се внимава на поларитетот на напонот.
- Максималниот напон за влезно-излезните пинови на Arduino Uno R3 изнесува 5,5V.
- Максималната влезна струја за еден пин на Arduino Uno R3 изнесува 40mA. Збирот на влезните струи за сите пинови не смее да биде поголем од 200mA.
- Не е дозволено кратко спојување на пиновите.
- Изводите на поларизираните компоненти да се идентификуваат и соодветно да се поврзат.
- Пред да се започне со поврзување да се пресмета јачината на струјата низ електричните компоненти и да се провери дали истата е еднаква или помала од максимално дозволената струја според техничко-технолошката документација за таа компонента.
- За поврзување на мотори или други индуктивни потрошувачи да се користат соодветни контролери (анг. motor driver)
- За ограничување на струјата низ лед диодите да се користат отпорници

2.8.3.2. Упатство за користење на протоплочка

Лемењето е постапка за изработка на електронски уреди на печатена плочка со постојан дизајн, стабилна работа и сигурни електрични контакти. Но,

често пати се случува дизајнот на електричното коло да се менува со цел да се подобри функционалноста и квалитетот на уредот. Ова значи промена на начинот на поврзување на компонентите, па дури и замена на истите. Протоплочките се идеални за брзо и едноставно менување на дизајнот на електричните кола. [6]

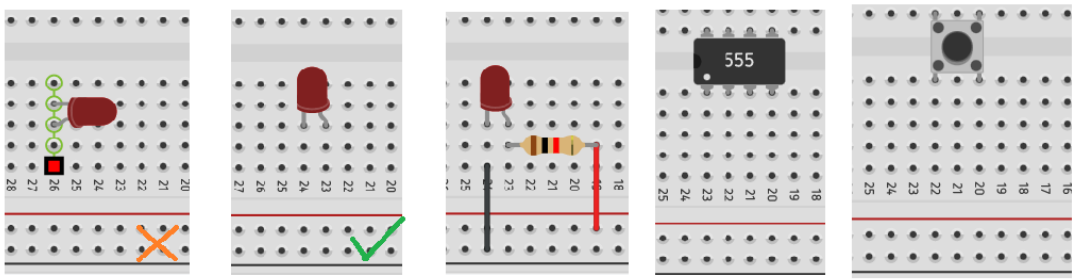


Слика 2.35. Надворешен изглед на протоплочка

Протоплочката претставува плочка изработена од висококвалитетна пластика со отвори на истата кои се всушност контактни точки или точки за поврзување. Во контактните точки се поставуваат изводите на електронските елементи или жиците за поврзување на истите. На слика 2.35. е прикажан надворешниот изглед на протоплочката. Под пластичната маска се наоѓаат проводници, поставени паралелно во неколку групи, кои вршат електрично поврзување на електронските елементи. На самите проводници се наоѓаат контакти во облик на штипки и електронските елементи механички се прицврстуваат, закачуваат на протоплочката со благо притискање на изводите.

Електронските елементи се поставуваат на вертикалните ленти, во средината на плочката. По подолгите страни на плочката, од двете страни се наоѓаат два пара ленти (хоризонталните ленти на слика 2.35.). Лентата обележана со + или црвена боја се поврзува со напојувањето, а лентата обележана со - или црна боја со заземјувањето.

За правилно поврзување на елементите многу е важно да се познава распоредот на металните ленти. **Сите контактни точки што лежат на иста лента се електронски поврзани меѓу себе и треба да се внимава при поставување на елементите да не дојде до кусо поврзување на изводите,** бидејќи тогаш низ нив нема да тече струја.



Слика 2.36.. Поврзување на електрични компоненти на протоплочка

На пример, на слика 2.36., првата лед диодата е неправилно поставена бидејќи нејзините електроди, анодата и катодата, се поставени во контактни точки кои припаѓаат на иста лента. Втората диода на слика 2.36. претставува правилно поврзување на лед диодата со протоплочката. Доколку сакаме да поврземе два елементи, на пример лед диода и отпорник, тогаш нивните изводи треба да бидат поставени во контактни точки кои припаѓаат на иста лента.

На средината на протоплочката има вдлабнатина, канал кој електрично ги разделува вертикалните проводници на два дела. Ова дозволува поврзување на интегрирани кола со два реда на изводи (анг. DIP-Dual in Line). Интегрираното коло се поставува точно врз каналот така што неговите пинови се поставени на различни страни од каналот и не се електрично поврзани. Контактните точки на протоплочката се на растојание 2,45mm колку што изнесува растојанието меѓу пиновите на интегрираните кола. На средишниот канал се поставуваат и тастери. Тастерите имаат четири приклучоци, по два од две спротивни страни. **Приклучоците од иста страна не се електрично поврзани** и тие треба да се постават во контактните точки кои припаѓаат на два вертикални проводници, за да електричното коло се затвори кога тастерот ќе се притисне.

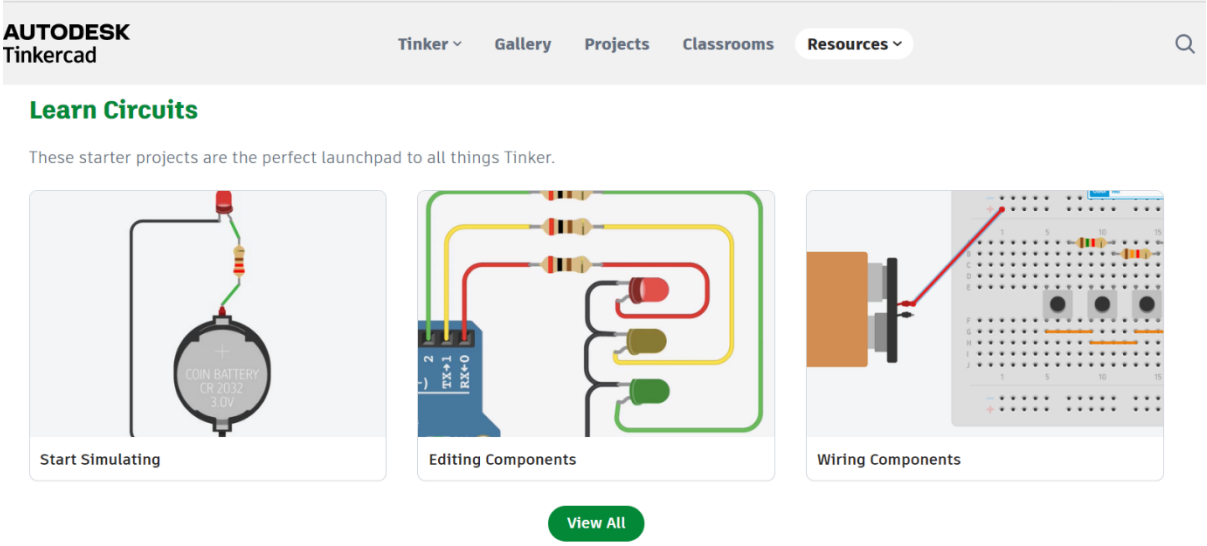
2.8.3.3. Компјутерска симулација за Arduino Uno R3

Компјутерските симулатори станаа често користена алатка при учење и работа во областа на електрониката и компјутерската техника. Тие ги поттикнуваат љубопитноста, инвентивноста, интуитивното размислување и се побезбедни за работа.

Предвидените практични вежби треба да се изведуваат во кабинет за практична настава и за тоа е потребна соодветна опрема, хардвер и електронски елементи. Дел од вежбите можат да бидат целосно реализирани и со примена на компјутерска симулација. Tinkercad е бесплатна веб-платформа која нуди одлична компјутерска симулација за Arduino Uno R3.

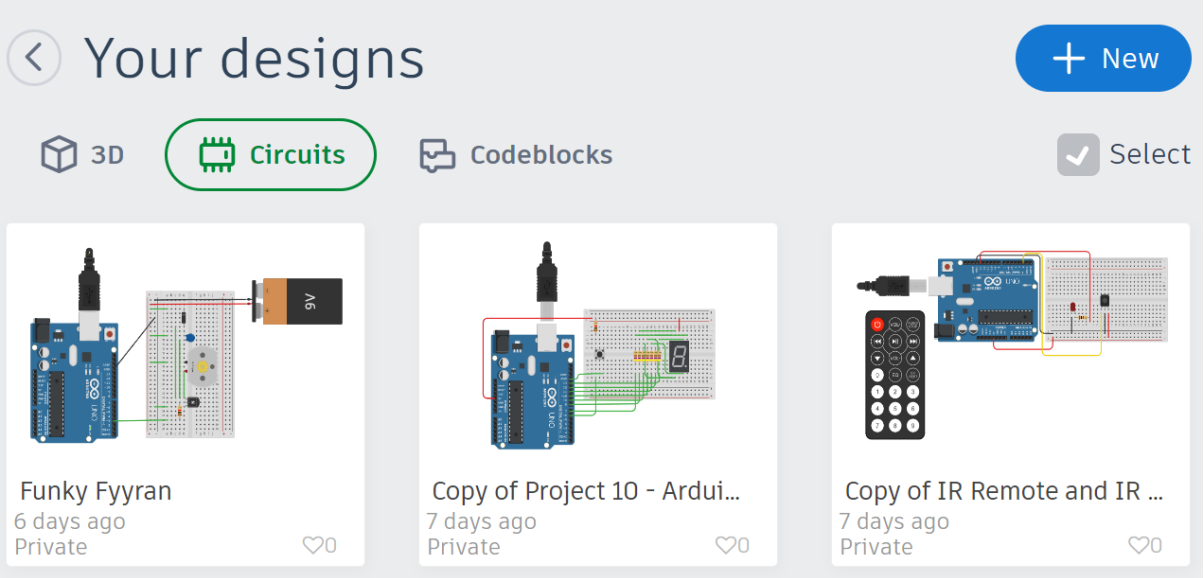
Корисникот треба да поседува Google или Apple корисничка сметка и со неа да се најави на официјалната веб-страница на платформата www.Tinkercad.com. После успешната **најава** се отвора почетната веб-страница со главното мени. Можностите се многу големи. Да напомниме дека Tinkercad платформата нуди работа со 3D дизајн, креирање на програмски кодови во програмскиот јазик Codeblocks и дизајн на електрични кола. Нам ни е потребна симулација на електрични кола со употреба со на Arduino Uno R3 и програмскиот јазик C/C++.

Со кликање на опциите Resources→Learning Center→Learn Circuit (мак. Ресурси→Центар за учење→Учење на електрични кола) се отвораат упатства за избор на компоненти, нивно поврзување, пишување на програми и симулирање на истите. Упатствата за работа се напишани во чекори и се интерактивни односно може веднаш да се применат бидејќи со отворање на упатството се отвора и работна површина и мени за избор на компоненти.



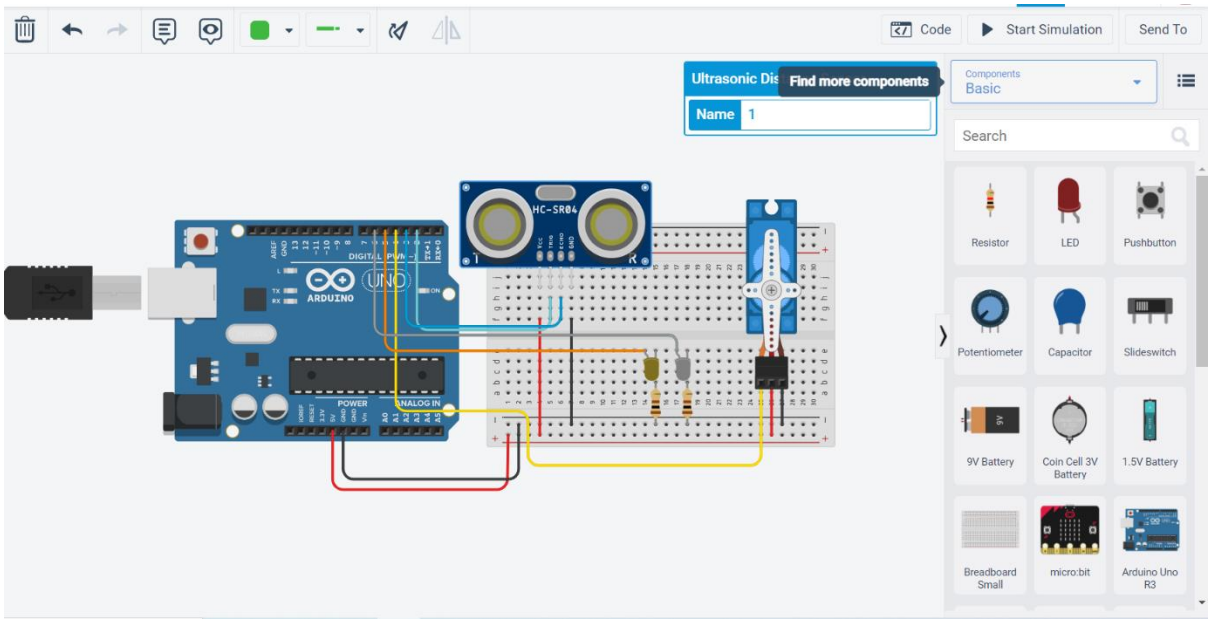
Слика 2.37. Отворање на упатства за креирање на елктрични кола со употреба на Arduino Uno R3

Со притискање (клик) на профилната слика и избор на опцијата Мој дизајн (анг. My Designs) се отвораат сите проекти на кои сме работеле претходно. Тие се сочувани од самата веб-платформа. Доколку сакаме да креираме ново електрично коло тогаш ја избираме опцијата Ново (анг. New).



Слика 2.38.. Отворање на нови и стари дизајни на електрични кола

На слика 2.39. е прикажан изгледот на работната површина за составување на електрични кола и програмирање на Arduino Uno R3. Електричното коло се составува многу лесно, со повлекување и пуштање на електронските компоненти, а потоа истите се поврзуваат. Програмата за работа може да ја видиме или напишеме доколку притиснеме на копчето **Code** и ја одбереме опцијата **Text.** Откако ќе го составиме електричното коло и ја напишеме програмата, го притискаме копчето **Start Simulation** и набљудуваме дали колото работи како што е предвидено. Доколку има грешки во кодот, истите ќе бидат обележани со црвена боја.



Слика 2.39. Изглед на работната површина за работа со електрични кола во веб-платформата Tinkercad

Tinkercad веб-платформата е од отворен тип и овозможува споделување на проекти. Со кликање на опцијата Испрати на (анг. Send To) во самиот проект се отвора нов прозорец, избираме Покани луѓе (анг. Invite people.), го копираме (анг. Copy) линкот за споделување и потоа со вметни (анг. Paste) го испраќаме проектот до саканата личност преку е-пошта или некоја социјална мрежа. Всушност со пребарување на интернет може да се најдат многу готови проекти изработени во Tinkercad и со нивно отворање и избор на опцијата Размисли (анг. Tinker this) веб-пребарувачот директно го отвора избраниот проект во Tinkercad платформата. Ова може многу да помогне во подобрување на нашите проекти.

2.8.3.4. Практична вежба: Реализација на И логичко коло со употреба на прекинувачи или Arduino Uno R3

1. **Цел на вежбата е поврзување на електронски компоненти и добивање на електрично коло за** реализација на И логичка функција на два начина
    - со употреба на сериски поврзани тастери
    - со употреба на Arduino Uno R3

    На ваков начин полесно ќе се разбере логичката функција и ќе се направи споредба меѓу двете изведби. Бидејќи со софтверот ќе се запознаеме подоцна сега ќе го составиме и објасниме само електричното коло, а во четвртата модуларна единица ќе ја испрограмираме Arduino платформата. [9]

2. **Потребни компоненти** за реализација на оваа вежба се:
   - Arduino Uno R3
   - протоплочка
   - два тастери за печатена плочка
   - лед диода
   - три отпорници (R3=220Ω, R1=R2=1KΩ)
   - жици за поврзување (краткоспојници)

3. **Подготовка за вежбата**

Пред да се започне со изведба на вежбата, ученикот треба да го прочита упатството за употреба на протоплочка, да се потсети на пин дијаграмот на Arduino Uno R3 и карактеристиките на тастерот и лед диодата како електронски елементи.

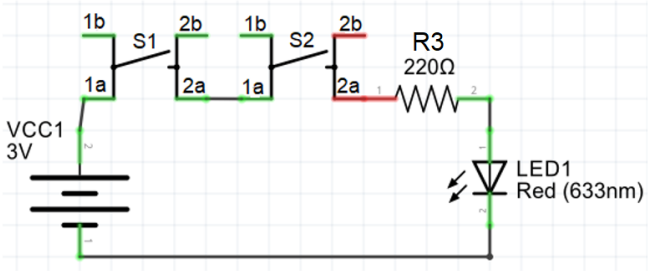
Логички симбол		
		
Табела на вистинитост		
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Слика 2.40. И логичко коло

Логичките кола се користат за обработка на дигитални сигнали. На слика 2.40. е даден симболот и таблицата на вистинитост на И логичко коло. За да на излез од И колото добиеме високо ниво односно логичка единица треба сите влезови да бидат на високо ниво. За да на излез добиеме ниско ниво односно логичка нула доволно е барем еден од влезовите да биде на ниско ниво.

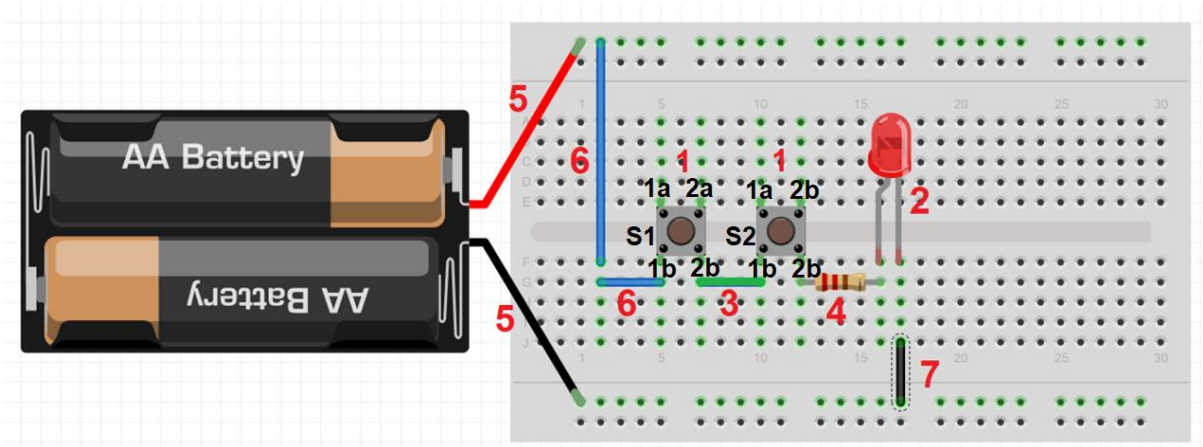
4. **Реализација на И логичка функција со употреба на тастери**

На слика 2.41. е прикажана функционалната шема за реализација на И логичка функција со употреба на два сериски поврзани тастери и една лед диода.



Слика 2.41. функционалната шема за два сериски поврзани тастери

Тастерите претставуваат влезови, а лед-диодата е излезот на И логичкото коло. На слика 2.42. е претставена монтажната шема и чекорите за реализација на вежбата.



Слика 2.42. Монтажна шема на И коло со употреба на тастери

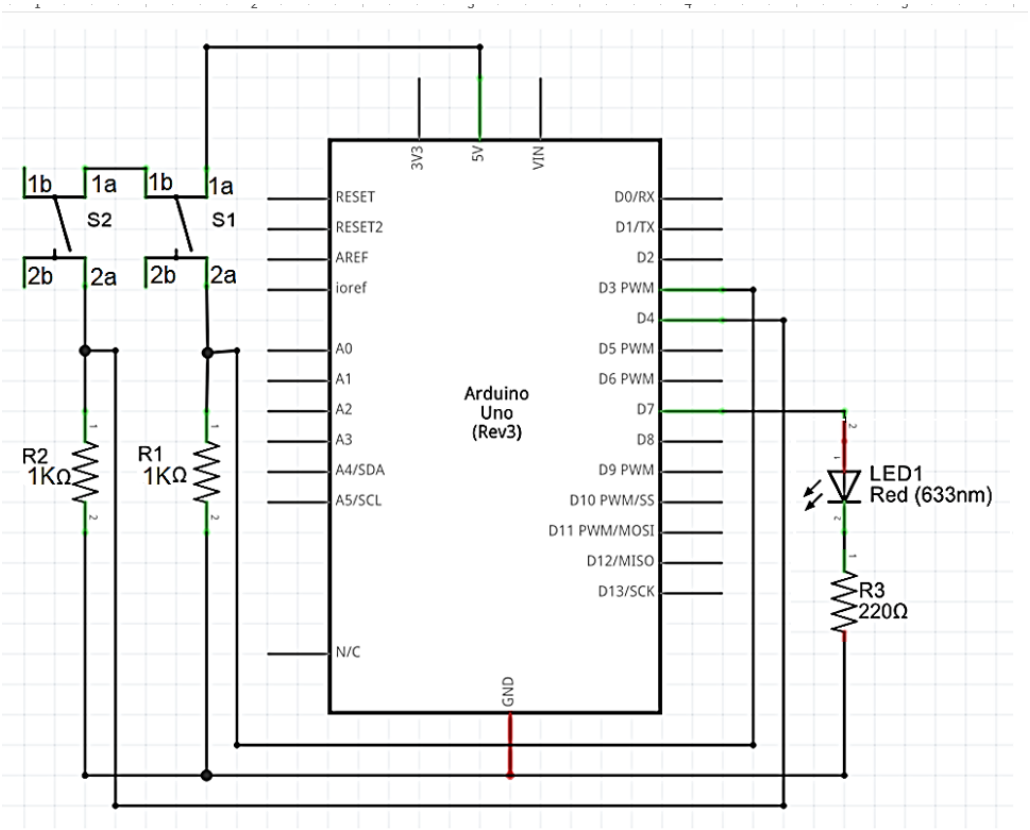
(1) Најнапред ги поставуваме двата тастери на средината од протоплочката, согласно слика 2.36. од упатството за работа со протоплочка.
(2) Ја поставуваме лед-диодата при што треба да се внимаваме анодата да биде поврзана со позитивниот пол на напојувањето.
(3) Со зелената краткоспојница ги поврзуваме соседните приклучоци на тастерите, приклучокот 2b на тастерот S1 со приклучокот 1b на тастерот S2.
(4) Со отпорникот R3 ја нагодуваме работната струја на лед-диодата. Истовремено преку отпорникот R3, лед-диодата ја поврзуваме со тастерот S2.
(5) Ја поврзуваме батеријата со лентите за напојување на протоплочката. Со црвена краткоспојница го поврзуваме позитивниот пол на батеријата со горната лента за напојување, а со црната жица го поврзуваме негативниот пол со долната лента.
(6) Со сините краткоспојници приклучокот 1a-1b на тастерот S1 го поврзуваме со позитивниот пол на напојувањето. Да се потсетиме приклучокот 1a е кратко споен со 1b.
(7) Катодата на лед-диодата ја поврзуваме со негативниот пол на напојувањето.

Само кога двата тастери ќе бидат притиснати низ колото ќе тече струја и лед-диодата ќе свети.

5. Реализација на И логичка функција со употреба на Arduino Uno R3

Од функционалната (слика 2.43.) и монтажната шема (слика 2.44.) може да се види дека тастерите S1 и S2 не се сериски поврзани. Тастерот S1 е поврзан со напојувањето од една страна (приклучок 1a-1b) и дигиталниот пин D3 од другата страна (приклучок 2a-2b). Истото важи и за тастерот S2 со таа разлика што наместо D3 тој е поврзан со дигиталниот пин D4. Да напомниме дека пиновите D3 и D4 треба да бидат влезни пинови. Меѓу секој тастер и заземјувањето е поврзан отпорник (R1 и R2).

Без нив не може да се знае логичкото ниво за дигиталните пинови бидејќи со притискање на кој било тастер напојувањето од 5V и заземјувањето кусо ќе се поврзат што не е дозволено.



Слика 2.43. Функционална шема за И логичко коло реализирано со Arduino Uno R3

Кога тастерите се отворени тогаш дигиталните пинови D3 и D4 се поврзани со заземјувањето преку отпорниците R1 и R2, низ кој не тече струја бидејќи нема напојување. Кога тастерите ќе се затворат пиновите D3 и D4 ќе се поврзат со напојувањето и ќе се постават на високо логичко ниво. Анодата на лед-диодата е поврзана со дигиталниот излезен пин D7, а катодата е поврзана со заземјувањето преку отпорникот R3. Кога дигиталниот пин D7 ќе биде на високо ново лед диодата ќе светне. За да ова се случи потребно е двата тастери да бидат притиснати. Состојбата на тастерите се проверува софтверски.

Чекорите за реализација на оваа вежба се следни:

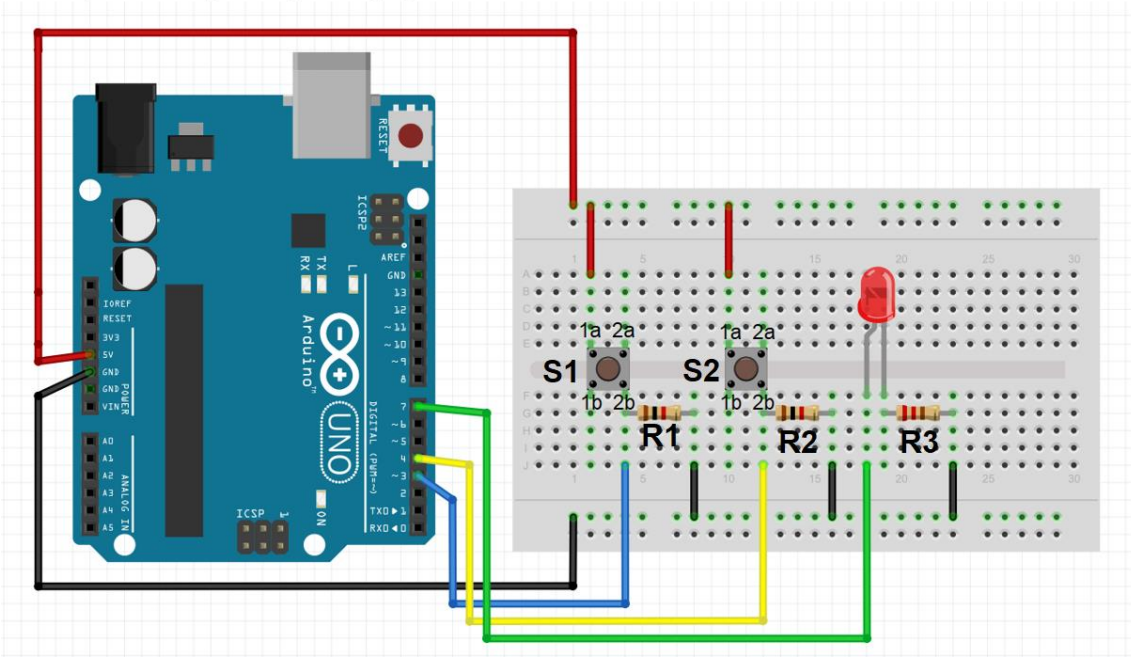
(1) Го поврзуваме напојувањето за протоплочката. Со црвена краткоспојница го поврзуваме пинот 5V со горната лента за напојување на протоплочката, а со црната краткоспојница го поврзуваме заземјувањето (анг. GND-Ground) со долната странична лента. Едната од двете краткоспојници за напојување треба да се поврзе на крајот од вежбата, после поврзувањето на сите елементи на протоплочката.

(2) Ги поставуваме тастерите на средината од протоплочката. Приклучоците 1a-1b на двата тастери соодветно ги поврзуваме со напојувањето со

црвена краткоспојница, а приклучоците 2a-2b, преку отпорникот R1 или R2, со заземјувањето.

(3) Со сината краткоспојница ја поврзуваме точката меѓу тастерот S1 и отпорникот R1 со дигиталниот пин D3. Истото го повторуваме со жолтата краткоспојница за точката меѓу тастерот S2 и отпорникот R2 и дигиталниот пин D4.

(4) Ја поставуваме лед-диодата на протоплочката, катодата преку отпорникот R3 ја поврзуваме со заземјувањето и со зелената краткоспојница ја поврзуваме анодата со седмиот дигитален пин.



Слика 2.44. Монтажна шема за И логичко коло реализирано со Arduino Uno R3

6. Програма за Arduino Uno R3 за реализација на И логичка функција

Ова практична вежба го содржи програмскиот код, но тој може да се употреби откако ќе се запознаеме со развојната средина и програмскиот јазик за програмирање на Arduino Uno R3 платформата. Засега само ќе го споменеме операндот && за извршување на И логичката инструкцијата. Во програмскиот код, овој операнд е обележан со жолта боја.

За целосна реализација на вежбата потребно е да се поврзе Arduino Uno R3 платформата со компјутер преку USB кабел, да се отвори развојната средина и напише програмата, истата да се впише во меморијата на Arduino Uno R3 и да се притиснат тастерите.

```
1  int S1 = 3;                 // Тастерот S1 е поврзан на третиот пин.
2  int S2 = 4;                 // Тастерот S2 е поврзан на четвртиот пин.
3  const int Led=  7;          // Лед-диодата е поврзана на седмиот пин.
4  int S1Sostojba = 0;         // S1Sostojba е променлива за читање на
                               // состојбата на тастерот S1.
```

```
5  int S2Sostojba = 0;          // S2Sostojba е променлива за читање на
                                // состојбата на тастерот S2.
6  void setup() {               // Почеток на функцијата за
                                // конфигурирање на пинови.
7      pinMode(Led, OUTPUT);    // Пинот за поврзување на лед-диодата е
                                // излезен.
8      pinMode(S1, INPUT);      Пиновите за поврзување на тастери се
9      pinMode(S2, INPUT);      // влезни.
                                //
10 }
11 void loop(){
12    S1Sostojba=digitalRead(A);  // Читање на состојба на пинови.
13    BStatus = digitalRead(B);
   //Во условот на структурата за гранење If е искористена инструкцијата за И
   //логичка функција. Со ова инструкција се проверува состојбата на тастерите.
14    if (S1Sosotojba == HIGH && S2Sostojba == HIGH) {
          digitalWrite(Led, HIGH);  // Вклучување на лед диодата ако е
                                    // исполнет дадениот услов.
15          }
16    else {
17        digitalWrite(Led, LOW);   // Ако условот не е исполнет диодата нема
                                    // да биде вклучена.
18      }
19 }
```

7. **Очекувани резултати**

Лед-диодата треба да светне само кога ќе бидат притиснати двата тастери истовремено. Кога тастерите не се притиснати или е притиснат само еден тастер лед-диодата не свети. Ако добиените резултати се разликуваат од очекуваните и развојната средина не пријави софтверска грешка при преведувањето и впишувањето на програмата потребно е да се изврши проверка на поврзаноста на електронските елементи.

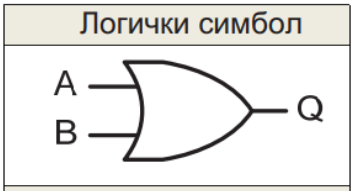
2.8.3.5. Практична вежба: Реализација на ИЛИ логичко коло со употреба на прекинувачи или Arduino Uno R3

1. **Целта на оваа вежба** е слична со претходната со таа разлика што наместо И треба да се реализира **ИЛИ логичка функција**. И оваа вежба е поделена на два дела, дизајн на електрично коло со употреба на паралелно поврзани тастери и со употреба на Arduino Uno R3. Да напомниме дека електричното коло со два паралелно поврзани тастери

претставува хардверска изведба на ИЛИ логичката функција, додека електричното коло со Arduino Uno R3 е софтверска изведба и е доволно да се промени само една инструкција во програмата за И логичка функција и да се добие нова програма за ИЛИ логичка функција. Електричното коло со Arduino Uno R3 останува непроменето.

1. **Потребни компоненти** за реализација на оваа вежба се:
   - Arduino Uno R3
   - протоплочка
   - два тастери за печатена плочка
   - лед диода
   - три отпорници (R1=R2=1KΩ, R3=220Ω,)
   - жици за поврзување (краткоспојници)

2. **Подготовка за вежбата**

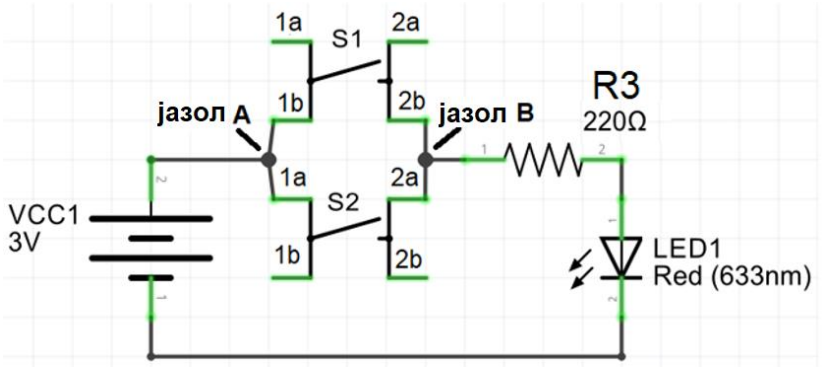


Табела на вистинитост		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Слика 2.45. ИЛИ коло

Истото што беше наведено во претходната вежба за реализација на И логичка функција важи и за оваа вежба. На слика 2.45. се дадени симболот и таблицата на вистинитост за ИЛИ логичко коло. За да на излез од ИЛИ колото добиеме високо ниво односно логичка единица доволно е еден од влезовите да биде на високо ниво, а може ѝ сите влезови да бидат на високо ниво. За да на излез добиеме ниско ниво односно логичка нула потребно е сите влезови да бидат на ниско ниво.

3. **Реализација на ИЛИ логичка функција со употреба на тастери**

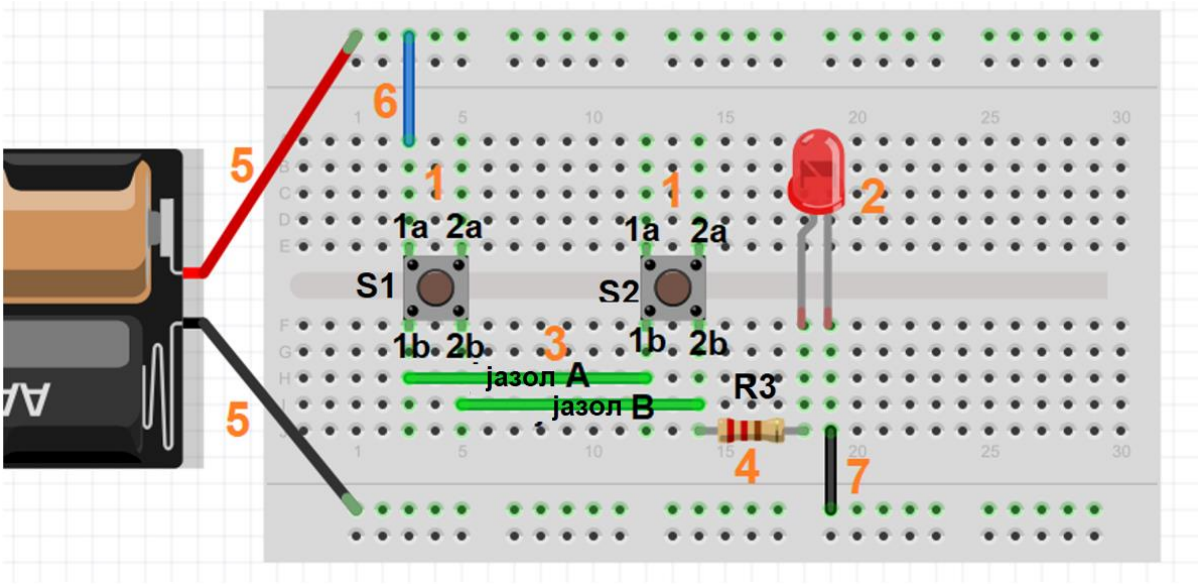


Слика 2.46. функционалната шема за два паралелно поврзани тастери

На слика 2.44. е прикажана функционалната шема за реализација на ИЛИ логичка функција со употреба на два паралелно поврзани тастери

и една лед диода. Доволно е само еден од тастерите да биде притиснат за да се добие затворено струјно коло и лед-диодата да светне.

На слика 2.47. е претставена монтажната шема за реализација на вежбата. При поврзувањето на електронските компоненти во паралелна врска со употреба на протоплочка треба да се внимава на изборот на контактни точки. Јазлите на паралелната врска се формираат со употреба на краткоспојници кои ги поврзуваат приклучоците на електронските елементи. Краткоспојницата треба да ја поставиме во контакна точка која припаѓа на металната лента на која е закачен ѝ приклучокот на електронскиот елемент кој треба да биде дел од паралелната врска. Да напомниме дека контактните точки кои припаѓаат на иста метална лета се обележани со ист број и различна буква.



Слика 2.47. Монтажна шема на ИЛИ коло со употреба на тастери

(1) Најнапред ги поставуваме двата тастери на средината од протоплочката
(2) Ја поставуваме лед-диодата при што треба да се внимаваме анодата да биде поврзана со позитивниот пол на напојувањето.
(3) Со зелените краткоспојници ги поврзуваме приклучоците на двата тастери означени со број 1a-1b и број 2a-2b и ги формираме јазлите A и B.
(4) Со отпорникот R3 ја нагодуваме работната струја на лед-диодата. Истовремено преку отпорникот лед-диодата ја поврзуваме со јазелот A.
(5) Ја поврзуваме батеријата со лентите за напојување на протоплочката. Со црвена краткоспојница го поврзуваме позитивниот пол на батеријата со горната лента за напојување, а со црната жица го поврзуваме негативниот пол со долната лента.
(6) Со сините краткоспојници ги поврзуваме приклучоците 1a-1b на тастерот S1 со позитивниот пол на напојувањето.

(7) Катодата на лед-диодата ја поврзуваме со негативниот пол на напојувањето.

За да светне лед-диодата доволно е да биде притиснат еден од двата тестери.

4. **Реализација на ИЛИ логичка функција со употреба на Arduino Uno R3**

Софтверските решенија често пати се многу поедноставни од хардверските. Наместо ново електрично коло за реализација на оваа вежба ќе го искористиме електричното коло од претходната вежба, чија функционална шема е прикажана на слика 2.43., а монтажна шема на слика 2.44. Кога електричното коло е едноставно, лесно може да се направи промена во ожичувањето. Но, во случај на голем број електронски компоненти и краткоспојници промената на дизајнот може да биде сложена.

5. **Програма за Arduino Uno R3 за реализација на И логичка функција**

За да реализираме ИЛИ логичка функција доволно е да промениме само една инструкција во програмата за И логичка функција. Операторот && за И логичка функција ќе го замениме со нов оператор ||за ИЛИ логичка операција.

```
1  int S1 = 3;                    // Тастерот S1 е поврзан на третиот пин.
2  int S2 = 4;                    // Тастерот S2 е поврзан на четвртиот пин.
3  const int Led=  7;             // Лед-диодата е поврзана на седмиот пин.
4  int S1Sostojba = 0;            // S1Sostojba е променлива за читање на
                                  // состојбата на тастерот S1.
5  int S2Sostojba = 0;            // S2Sostojba е променлива за читање на
                                  // состојбата на тастерот S2.
6  void setup() {                 // Почеток на функцијата за
                                  // конфигурирање на пинови.
7      pinMode(Led, OUTPUT);      // Пинот за поврзување на лед-диодата е
                                  // излезен.
8      pinMode(S1, INPUT);            Пиновите за поврзување на тастери се
9      pinMode(S2, INPUT);        // влезни.
                                  //
10 }
11 void loop(){
12    S1Sostojba=digitalRead(A);  // Читање на состојба на пинови.
13    BStatus = digitalRead(B);
//Во условот на структурата за гранење If е искористена инструкцијата за И
//логичка функција. Со ова инструкција се проверува состојбата на тастерите.
14    if (S1Sosotojba == HIGH || S2Sostojba == HIGH) {
          digitalWrite(Led, HIGH);  // Вклучување на лед диодата ако е
                                    // исполнет дадениниот услов.
```

```
15          }
16      else {
17          digitalWrite(Led, LOW);       //  Ако условот не е исполнет диодата нема
                                          //  да биде вклучена.
18        }
19  }
```

6. Очекувани резултати

Исто како претходната вежба, за целосна реализација на ИЛИ логичката функција со Arduino Uno R3 потребно е да се проучи неговата развојна средина и инструкциското множество. После поврзувањето на електронските компоненти и впишувањето на програмата лед-диодата треба да светне кога ќе биде притиснат барем еден од тастерите или двата истовремено.

Заклучоци

Составни хардверски компоненти на компјутер се: микропроцесор, мемории, магистрали и периферни уреди. Микропроцесорот ги обработува податоците, мемориите ги чуваат, а магистралите ги пренесуваат податоците. Периферните уреди се познати и под името влезно-излезни единици.

Оперативниот систем е посредник меѓу хардверот и софтверот. Оперативниот систем е софтвер кој менаџира со хардверските компоненти: процесорот, мемориите, влезно излезните уреди. Развојна програма е софтвер за изработка на нов софтвер.

Микропроцесорот има две основни функции: извршува програми и управува со другите уреди на матичната плоча. Составни делови на еден микропроцесор се: регистри, аритметичко-логичката единица и управувачката единица со декодер. Најважни карактеристики на микропроцесор се: работната фреквенција, големината на податоците, проточниот концепт, комплексност на инструкциско множество, број на јадра, распоред на пинови и др

RAM-меморијата го снабдува процесорот со информации, а пак таа добива информации од хард-дискот. Сите програми се сместени на хард-дискот и тие се пренесуваат во RAM кога треба да бидат обработени.

Хард-дискот, SSD уредите и надворешните мемории (компакт дискови, USB мемории, SD картички и др) се користат за трајно чување на податоци.

Изборот на матична плоча зависи од изборот на процесор бидејќи секој тип на процесор има уникатен распоред на пинови кои бараат и соодветно подножје.

Чипсетот е множество од специјални интегрирани кола кои го контролираат протокот на податоци.

DDR4, DDR3, DDR2 и DDR RAM модулите се разликуваат по бројот и распоредот на пинови, напојувањето и работната фреквенција односно брзината.

PCI Express×16 и AGP се најдобри конектори за поврзување на графички картички. PCI слотот се користи за поврзување на звучни картички, мрежни картички. SATA2 и SATA3 конекторите се за поврзување на хард дискот и SSD меморија.

При инсталацијата на процесорот и RAM меморијата не смее да се притиска и да се допираат златните пинови.

Повеќето кабли за напојување имаат уникатен дизајн, па не може да се згреши при нивното поврзување на матичната плоча со другите уреди. .

SATA2 конекторот е со сина боја и се користи за поврзување на хард дискот. SATA3 конекторот е обично со бела боја и се користи за приклучување на SSD меморијата.

Основна компонента во Arduino Uno R3 платформа е микроконтролерот ATmega328. Програмска меморија е со капацитет 32KB, податочна EEPROM меморија е со 1KB и внатрешната RAM меморија изнесува 2KB. Максималната фреквенција изнесува 20MHz..

Arduino Uno R3 располага со 14 дигитални пинови, шест аналогни влезови, три лед-диоди индикатори и една вградена лед-диода. Со аналогните сигнали треба да се изврши импулсно ширинска модулација.

За Arduino Uno R3 влезни единици се: сензори (пиезо, тилт сензор, фотоотпорник, температурен сензор), тастери, потенциометри. Излезни единици се: дисплеи, мотори, бипери.

За да лед-диодата свети потребно е подолгиот приклучок (анода) да се поврзе на повисок потенцијал во однос на пократкиот приклучок (катода). Приклучоците од иста страна на тастерот не се електрично поврзани.

Кондензаторите се поврзуваат паралелно со сензор или мотор, со цел да се спречи брза промена на напонот. Отпорниците се користат за нагодување на јачината на работната струја.

Заради лична заштита и заштита на елементите од оштетување, пред почетокот на секоја монтажа треба да се исклучи изворот за напојување, без оглед дали се работи за батерија или компјутер.

Од Arduino додатоците ќе ги споменеме: ESP8266-01 Wi-fi модулот, Ethernet штитот, SM/GPRS Arduino штитот, мотор и релеј-штитот.

Максималниот напон за влезно-излезните пинови на Arduino Uno R3 изнесува 5,5V, а максималната влезна струја за еден пин на Arduino Uno R3 изнесува 40mA.

Составни делови на Raspberry Pi 3B+ се: системскиот чип Broadcom BCM2837 SoC (System on Chip), RAM меморијата, радиопредавател, четирите USB приклучоци, RJ45 приклучок, HDMI, специјални конектори за камера и LCD дисплеј и 40 пинско подножје за влезно излезни уреди.

Под пластичната маска на протоплочката се наоѓаат проводници, поставени паралелно во неколку групи, кои вршат електрично поврзување на електронските елементи. На самите проводници се наоѓаат контакти во облик на штипки.

Прашања за повторување

1. Кои се четирите составни делови на микрокомпјутерот и која е нивната функција?
2. Кои се трите составни делови на микропроцесорот и која е нивната функција?
3. Набројги најважните карактеристики на микропроцесорот?
4. Кои се двете најважни карактеристики на мемориите?
5. RAM е работна, привремена меморија со случаен пристап? Објасни?
6. Кои се предности и недостатоци на хард-диск меморискиот уред во однос на SDD уредот?
7. Што претставува и за што служи чип сетот на матичната плоча на персоналниот компјутер?

8. Како може да се препознае видот на DDR RAM модул?
9. Кои слотови од матичната плоча се користат за поврзување на графичката картичка?
10. Што извршува BIOS-от после секое вклучување на компјутерот?
11. Објасни за што се користат SATA2 и SATA3 конекторите на матичната плоча?
12. Кој е најдобар приклучок за поврзување на видео уредите?
13. Во четири чекори објасни ја инсталацијата на процесорот во неговото подножје?
14. На што треба да се внимава при инсталацијата на RAM меморијата во слотовите на матичната плоча?
15. За што служат F (Front), SYS_FAN и PWR-FAN приклучоците на матичната плоча?
16. Именувај ги каблите за напојување на матична плоча и процесор? По колку пинови има секој кабел!
17. Кои се најважни карактеристики на микроконтролерот ATmega 328?
18. Кои се трите начини за напојување на Arduino Uno R3 платформа?
19. Кои дигитални пинови на Arduino Uno R3 можат да се искористат за пренесување на аналогни сигнали?
20. Колку лед-диоди индикатори содржи Arduino Uno R3 и за што служи секој од нив?
21. Набројнеколку влезни уреди за Arduino платформите?
22. Кои отвори на површината од протоплочката се електрично поврзани?
23. Колку пинови има еден тастер и кои од нив се електрично поврзани?
24. На што треба да се внимава при поврзувањето на лед-диодата во електрично коло?

25. Објасни како работи серво моторот?

26. Каква намена има пиезо компонентата?

27. Објасни ја мемориската организација на Arduino Uno R3 платформа!

28. За што служат кондензаторите и отпорниците при поврзување на влезно-излезни уреди со Arduino или Raspberry Pi?

29. Набројнеколку функции на Arduino штитовите?

30. Колку изнесува максималната влезна струја за еден пин на Arduino Uno R3?

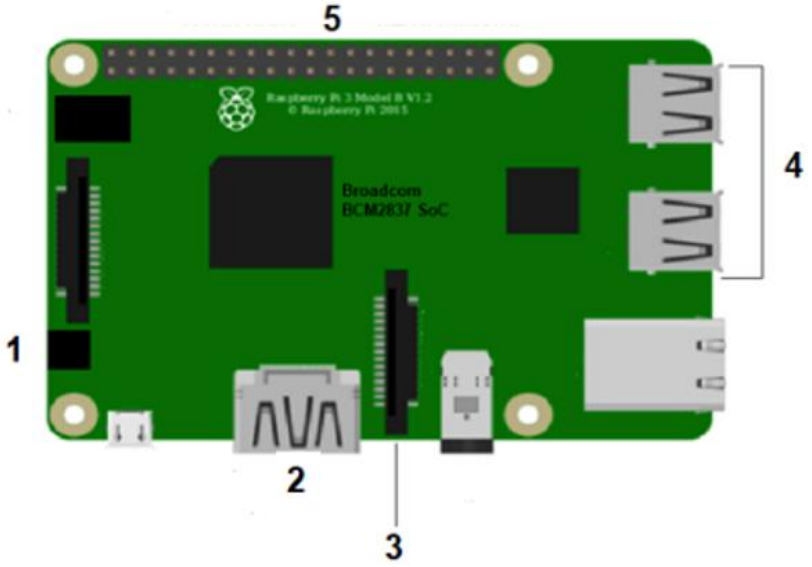
31. Објасни како се поврзува протоплочка со напојувањето на Arduino Uno R3?

32. Какви апликации може да се креираат со микрокомпјутерот Raspberry Pi?

33. GPIO (General Purpose Input Output) подножјето на Raspberry Pi 3 има 40 пинови, но GPIO броевите се разликуваат од физичките броеви на пиновите. Објасни!

34. Набројги компонентите во составот на додатокот со сензори и дисплеј (анг. Sense hat) за Raspberry Pi!

35. Именувај ги составните делови на Raspberry Pi 3B+ на долната слика 2.48. означени со броевите?



Слика 2.48. Составни делови на Raspberry Pi 3B+

3. Инсталација на оперативен систем

3.1. Видови оперативни системи

Оперативниот систем е софтвер потребен за извршување на апликативни програми и за координирање на активностите на компјутерскиот систем. Тој обезбедува околина во која другите програми можат да извршуваат корисна работа. Тој е програма како и секоја друга програма, но многу покомплексна, помоќна, поголема. Поради тоа тој мора да се креира дел по дел, со добро дефинирана функција, влез и излез за секој дел. Оперативните системи може да се проучуваат од две гледни точки: од онаа на корисникот и онаа на системот.

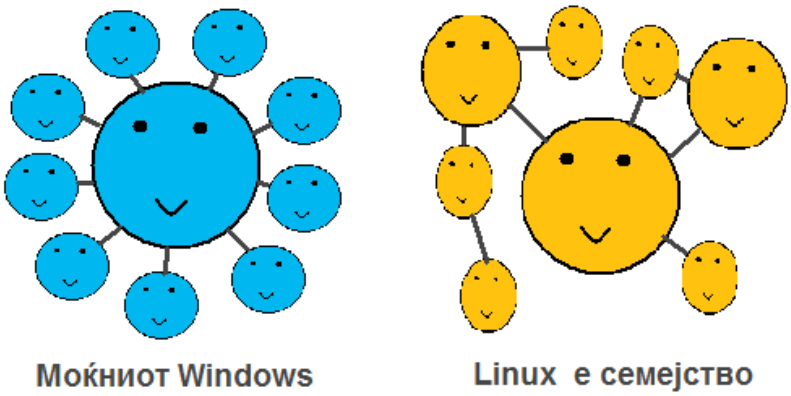
Од кориснички поглед оперативниот систем е дизајниран со цел да се минимизира работата на корисникот при искористувањето на хардверските ресурси односно да се обезбеди удобна употреба на микрокомпјутерските системи и подобрување на нивните перформанси. Компјутерот прима инструкции од корисникот преку тастатурата и дава пораки преку мониторот. Во зависност од обликот на овие наредби и методот на нивно внесување разликуваме два вида кориснички интерфејс, преку командна линија (анг. interface command line) или графички интерфејс (анг. graphics user interface). Кај интерфејсот со командна линија се употребуваат текстуални наредби, а кај графичкиот интерфејс се користи систем од прозорци, менија и листи за селекција. За пристап до интерфејсот со командна линија, во Windows оперативниот систем, во полето за пребарување (анг. Search) треба да се напише cmd (анг. Command Prompt).

Од системски поглед, оперативниот систем е контролно-управувачки софтвер и е тесно поврзан со хардверот. Оперативниот систем одлучува која програма ќе се изврши и кога, колку меморија ќе им се додели на програмите, се грижи за правилно извршување на командите дадени од корисникот и слично. Накратко ќе се запознаеме со функциите на оперативниот систем. [10]

- Извршување на програмите. Оперативниот систем е задолжен за преносот на инструкциите од RAM меморијата во микропроцесорот, нивно декодирање и извршување.
- Програми за распределба на хардверски ресурси. За ефикасна работа на микрокомпјутерскиот систем оперативниот систем мора да одлучи како да ги додели хардверските ресурси (микропроцесор, мемориски простор, влезно-излезни единици) на специфичните корисници и програми. На пример, микропроцесорот може да извршува повеќе задачи истовремено така што постојано ќе се префрла од една на друга задача, но префрлувањето се извршува толку често што корисниците и нивните програми се во постојано заемно дејство.
- Програми за управување со меморијата. Оперативниот систем одлучува кои програми и податоци ќе се пренесат од секундарната во примарната меморија, доделува и ослободува мемориски простор по потреба, ги следи тековно употребуваните делови на меморијата, служи за форматирање, чистење и одржување на хард-дискот, се грижи за работата на кеш меморијата.
- Програми за управување со влезно-излезни операции и уреди. За контрола на преносот на податоци од или кон периферните уреди и нивно привремено складирање се користат специјални интегрирани кола наречени контролери за уреди. За да започне преносот на податоци потребно е во регистрите на контролерот да се запишат бинарни кодови со точно определена вредност. Ова го извршува драјверот кој всушност е „двигател на уредот". Драјверите се помошни програми кои посредуваат меѓу оперативниот систем и контролерите.
- Програми за заштита и безбедност. Заштитата е механизам за контрола на пристап на корисниците до хардверските ресурси и авторизирана употреба на истите. Одбраната на системот од внатрешни и надворешни напади е работа на безбедност. Во оваа категорија спаѓаат антивирусните програми.
- Програми за комуникација. Овие програми овозможуваат пренос на датотеки на далечина од еден на друг компјутер, пребарување на веб-страници, испраќање на електронска пошта и др.
- Програми за управување со датотеки. Овие програми креираат, селектираат, бришат, копираат, преименуваат датотеки и директориуми.
- Програми за работа со статусни информации. Статусни информации се време, датум, количество на достапна меморија, број на корисници и слично.
- Поддршка за програмски јазици. Оперативниот систем му обезбедува на корисникот едитори, компајлери, асемблери, дебагери и толкувачи за најчесто користените програмски јазици. Овие алатки се неопходни за создавање на апликативен софтвер.

Оперативниот систем Windows е првиот оперативен систем со графички интерфејс и е производ на компанијата Microsoft. Тој е еден од најкористените и најпопуларните оперативни системи. Денес дури 80% од корисниците на персонални компјутери го користат оперативниот систем Windows. Во почетокот, Windows бил само надградба на тогаш популарниот оперативен систем DOS. Уште на самиот почеток постоела софтверска поддршка за работа со глувче, икони, мени и едноставен премин од една во друга апликација. **Предности на оперативниот систем Windows** се: едноставност во работењето (user friendly), најголема поддршка за поврзување со периферни уреди, достапност до апликативен софтвер, автоматска инсталација (plug and play), најдобри перформанси за видеоигри, компатибилност со најголем број на веб-страници**. Недостатоци** се: потреба од скапи компјутерски конфигурации, затворен код, слаба сигурност во поглед на интернет-напади и административна заштита, чувствителност на компјутерски вируси, потребна е лиценца, слаба експертска поддршка и честопати за да се зголеми брзината на работата, потребна е преинсталација на оперативниот систем.

Linux е понов оперативен систем. Идејата ја дал финскиот студент Линус Торвалдс, кој во 1991 година објавува труд за создавање јадро на нов бесплатен, стабилен оперативен систем со отворен код, кој ќе може да се применува на различни платформи. **Отворен или слободен код** значи сите корисници да имаат пристап до изворниот код и да можат да го менуваат во зависност од своите потреби. Отворениот код е посигурен развоен систем. При самата анализа, програмерите ги откриваат грешките и вирусите, ги поправаат и своите решенија ги споделуваат со заедницата на Linux. Самото јадро на оперативниот систем има заштита од неколку нивоа. Пред да се направи извршна верзија на која било програма, таа се проверува. Исто така, веб-пребарувачот е независен од оперативниот систем. На слика 3.1. симболично е прикажана организацијата на оперативниот систем Linux, споредена со организацијата на оперативниот систем Windows.



Слика 3.1. Два различни, но успешни концепти

Архитектурата на оперативниот систем Linux е како разгрането дрво, секоја гранка претставува еден член од заедницата и секој може да придонесе во создавањето на побрз, посилен и посигурен систем. Можеби Windows е господар во светот на оперативните системи, но револуцијата на Linux дефинитивно го промени начинот на генерирање на оперативните системи. На почетокот, оперативниот систем Linux бил наменет за понапредни корисници, но со појавата на ефикасниот интерфејс GUI (Graphical User Interface) Linux станува сè повеќе популарен. Денес Linux се користи како оперативен систем за сервери на големи компании, како што се Google, Facebook, Twitter. Многу компании го земаат јадрото на Linux како основа, го развиваат и создаваат нови лиценцирани оперативни системи. Таков е оперативниот систем Android на компанијата Google и оперативниот систем **Raspbian,** креиран од група ентузијасти, љубители на Raspberry Pi микрокомпјутерите.

Денес 74,3% од корисниците на мобилни телефони користат **Android** оперативен систем. **Предности на оперативниот систем Android** се: огромен број бесплатни апликации (Google Play Store е апликација вградена во самиот оперативен систем), компатибилност со уреди од различни производители (Samsung, Huawei, Motorola), отворен код, располага со едноставни развојни средини за создавање нови апликации, меморискиот простор лесно се проширува, можност за споделување на интернет, едноставна комуникација со многу уреди, слобода при конфигурацијата и избор на апликации**. Недостатоци** на овој оперативен систем се: апликациите се потешки за изработка поради различните димензии на екраните, помала брзина поради заднинските апликации, помала сигурност и осетливост на вируси.

Raspbian е оперативен систем само за Raspberry Pi. Подоцна ќе се запознаеме со неговата инсталација и начин на користење.

3.2. Практични вежби за модуларната единица: Инсталација на оперативен систем

3.2.1. Подготовка за инсталација на софтвер

Денес, инсталацијата на апликативниот и системскиот софтвер е мошне едноставна, истата се извршува во неколку чекори и секој чекор е детално објаснет во текот на самата постапка. Сепак за успешна инсталација корисникот треба да се придржува до следните правила.

- Треба да бидеме сигурни дека програмата што ја инсталираме ќе ни биде корисна. Секоја инсталирана програма, без оглед дали ја користиме или не, зафаќа меморија и ја успорува работата на компјутерот.
- За преземање на софтверот за инсталација треба да користиме сигурни и проверени линкови.
- Треба да провериме дали компјутерската конфигурација ги задоволува барањата за правилна работа на софтверот.
- Треба да провериме дали имаме доволно слободен простор во тајната меморија
- Треба да провериме дали постои можност за деинсталација
- Се препорачува корисникот да избере инсталација по негов избор (анг. Custom) бидејќи на таков начин ги селектира можностите кои му се нудат
- Треба да се обезбеди стабилно напојување и сигурен интернет пристап за да на дојде до прекин во текот на инсталацијата
- Практичните вежби за инсталација на софтвер да не се извршуваат без одобрение од предметен наставник

3.2.2. Практична вежба: Инсталација на Windows 10 оперативен систем

1. **Подготовки за инсталација на Windows 10 оперативен систем**

   Најнапред треба да се избере видот на Windows 10 оперативниот систем. Windows 10 Home е за домашна употреба, Windows 10 Pro е за професионалци, Windows 10 Enterprise за компании и Windows 10 Education за студенти и ученици. Сите видови оперативни системи се

достапни во **32 и 64-битна верзија**. Да нагласиме, 64-битната верзија не може да се инсталира на компјутер со 32-битен процесор. Податоците се обработувани побрзо со 64-битните процесори, кои овозможуваат и поефикасно искористување на RAM-меморијата над 4 GB.

Оперативниот систем Windows 10 може да работи со истата компјутерска конфигурација како ѝ Windows 7: процесор со работна фреквенција над 1 GHz, 2 GB капацитет на RAM-меморијата, хард диск со капацитет 16 GB за 32-битна верзија и 20 GB за 64-битна верзија, графичка картичка со WDDM (анг. Windows Display Driver Model), екран со резолуција 800 × 600 пиксели.

Доколку компјутерот располага со некоја постара лиценцирана верзија на Windows тогаш истата може да се надгради до Windows 10 и самата надградба е бесплатна. Но доколку се работи за нов компјутер тогаш Windows 10 оперативниот систем мора да се инсталира. За таа цел потребен е инсталациски мемориски модул (USB или DVD).

За активација на Windows 10 потребен е клуч на производ . Клучот на производот е код од 25 цифри за потврда на лиценцата. На веб-страната на Microsoft може да се најде официјална листа на добавувачи на Windows 10 кои го испорачуваат кодот заедно со доставата на оперативниот систем, или истиот може да биде испратен преку е-пошта, доколку програмата за инсталација е преземена од интернет. Друг начин за добивање на дигитална лиценца е преку корисничкиот профил на Microsoft. За таа цел се избира Settings→Update and Security→Activation. Го притискаме копчето Додај сметка (анг. Add an account) и се најавуваме со нашето корисничко име и лозинка.

Постапката за инсталација на Windows 10 ќе ја поделиме на три дела: креирање на инсталациски USB мемориски модул, негово повикување преку BIOS управувачката програма и конечно инсталација на самиот оперативен систем. [11]

2. Креирање на инсталациски USB мемориски модул

(1) USB меморискиот модул го поврзуваме со компјутер. Тој треба да има минимален капацитет од 8GB.

(2) https://www.microsoft.com/en-us/software-download/windows10%20 е линк за преземање на алатката за креирање на инсталациски модул (анг. Media Creation Tool). Се притиска копчето Download tool now (преземи ја алатката сега) и за неколку секунди алатката се појавува во папката за преземање (анг. Download).

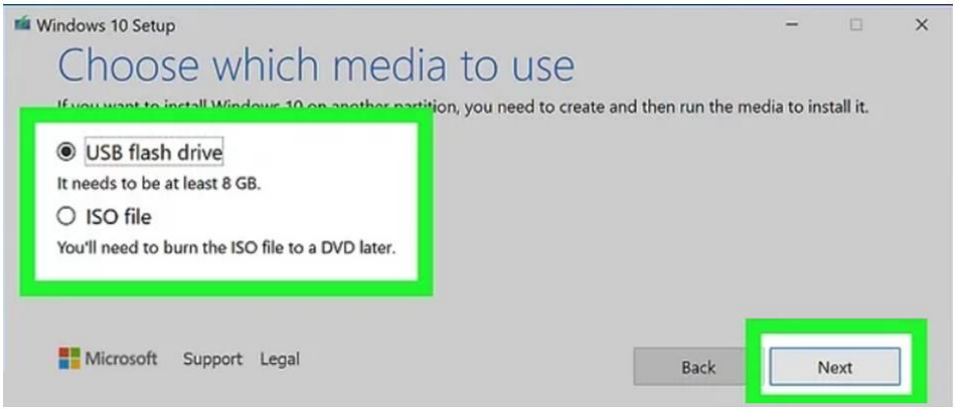
(3) Со двоен клик на алатката започнува креирањето на инсталациониот модул. Ги прифаќаме наведените услови (анг. Accept)

(4) Со следниот прозорец ја избираме опцијата за креирање на инсталациски модул.

Слика 3.2. Почеток на креирање на инсталациски модул за Windows

(5) Избираме јазик, оперативен систем и архитектура

(6) Избираме вид на модул и притискаме Следно (анг. Next)



Слика 3.3. Изборна мемориски модул, USB или DVD

(7) Доколку има повеќе USB мемориски модули избираме еден.

Слика 3.4. Избор на USB модул за сочувување на инсталацијата

(8) Чекаме да се инсталираат документите и на крај притискаме Заврши (анг. Finish).

3. Повикување на инсталициониот USB мемориски модул

(1) По приклучувањето на модулот и вклучувањето на компјутерот ја отвораме BIOS програмата, каде што се наоѓа менито за избор на

уред што ја содржи инсталацијата. За таа цел, непосредно по вклучувањето на компјутерот треба да го притискаме копчето **F1, F2 или Del**. Ова копче е различно за различни производители на компјутери.

(2) Од горното мени ја избираме опцијата Boot и потоа со помош на тастатурата се движиме горе-долу, го избираме уредот и притискаме Enter.

Слика 3.5 Избор на уред за инсталација

(3) По притискањето на Enter, се отвора прозорецот за инсталација (анг. Windows Setup Wizard).

4. Инсталација на Windows 10 оперативен систем

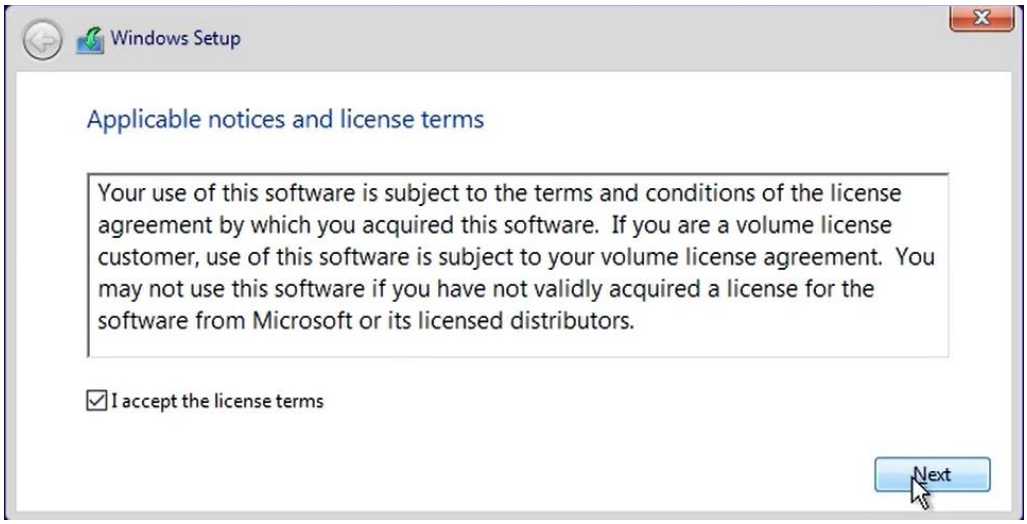
(1) Со првиот прозорец се избира јазикот, времето и форматот на датумот, јазикот на тастатурата за внесување податоци (US или UK). Притискаме Next и се отвора вториот прозорец.

Слика 3.6. Избор на стандарден јазик, време и јазикот за внесување податоци

(2) Со изборот на опцијата „Инсталирај сега“ (Install Now), започнува процесот на инсталација.

(3) На почетокот од инсталацијата се отвора прозорец во којшто се бара од корисникот да го внесе таканаречениот **клуч на производот**. Се работи за код од 25 цифри со кои се потврдува лиценцата. Овој код го испорачува Microsoft, заедно со доставата на оперативниот систем, или може да биде испратен преку е-пошта, доколку програмата за инсталација е преземена од интернет.

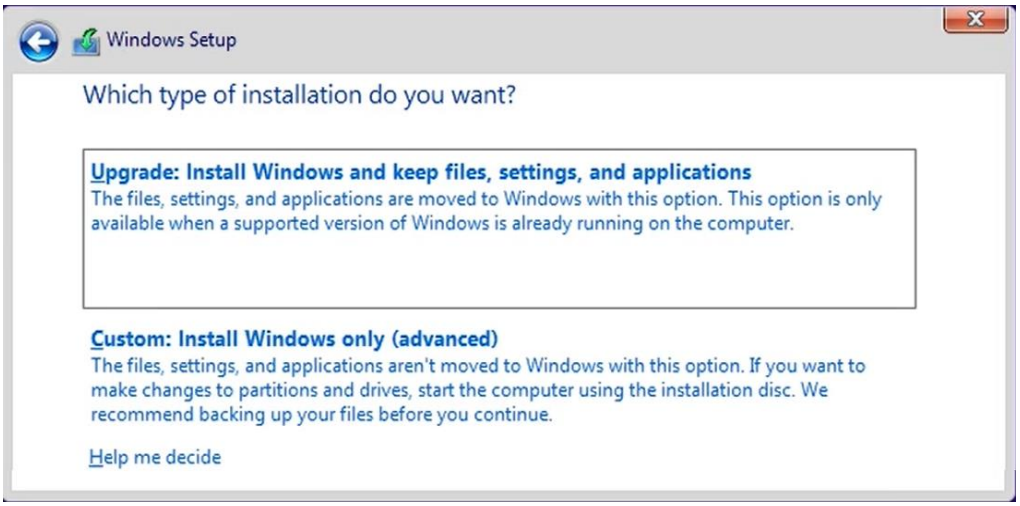
(4) Третиот прозорец е за прифаќање на условите што се поврзани со лиценцата на оперативниот систем.



Слика 3.7. Прифаќање на условите за лиценца

Условите ги прифаќаме со штиклирање на квадратчето од левата страна и притискање Next.

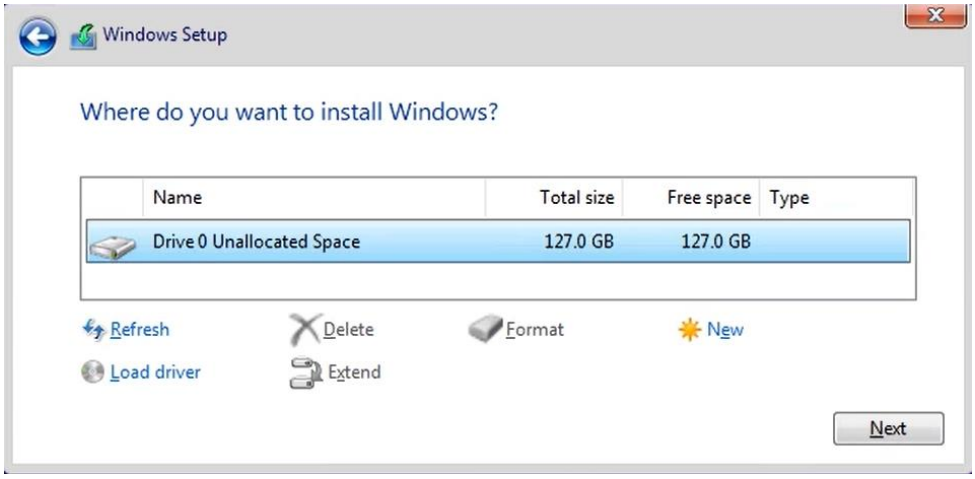
(5)



Слика 3.8. Избор меѓу надградба или нова инсталација на Windows

Четвртиот прозорец е за избор на вид на инсталација. Постојат две можности: Upgrade или Custom. Првата опција значи надградба на веќе постоечкиот оперативен систем Windows. Оваа инсталација е бесплатна, не бара клуч на производот, при што не се бришат корисничките податоци. Со втората опција се бришат сите податоци сочувани во компјутерот и ова значи нова инсталација.

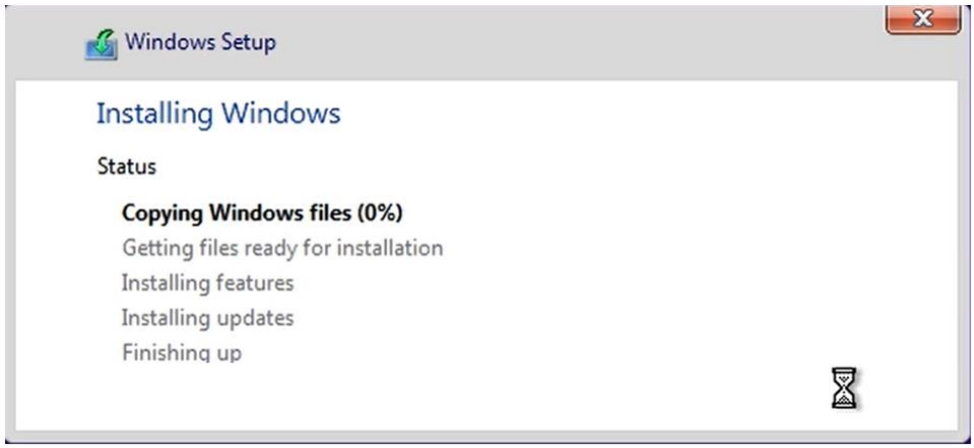
(6) Петтиот прозорец е за избор на дискот на којшто ќе го инсталираме оперативниот систем.



Слика 3.9. Избор на партиција од хард-дискот

Со притискање на копчето New може да се креира нова партиција на дискот, при што треба да се одреди и капацитетот на новата партиција. Во првата партиција се сместува оперативниот систем, а во втората се чуваат корисничките податоци. Да нагласиме дека со креирањето на втора партиција, автоматски се формира уште една системска партиција.

(7) Започнува инсталацијата на оперативниот систем и откако истата ќе заврши компјутерот ќе се ресетира



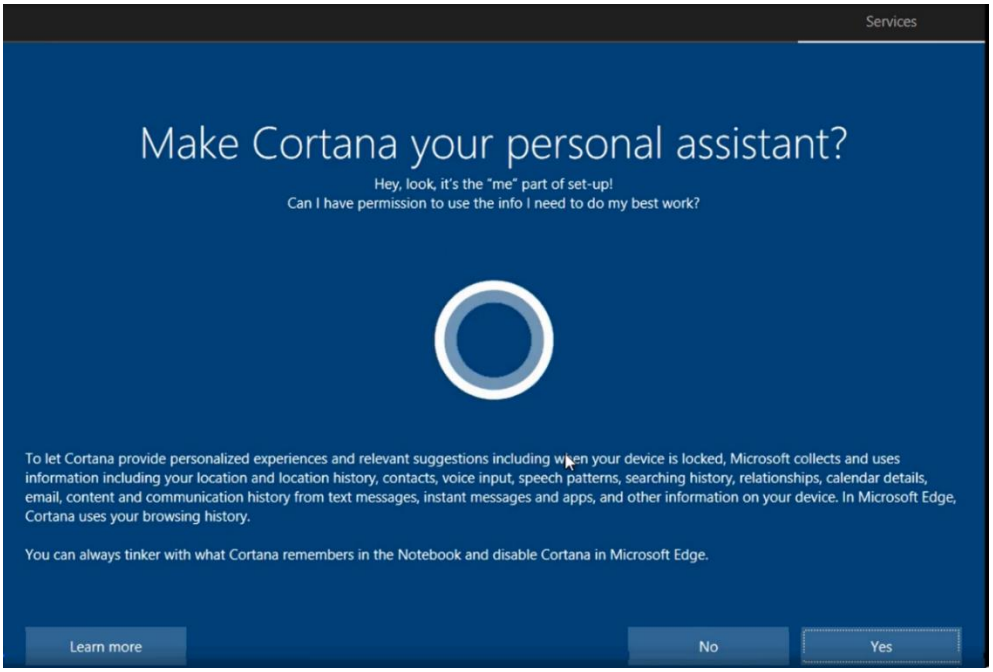
Слика 3.10. Почеток на инсталација на Windows

(8) Потоа следуваат неколку прозорци со коишто се врши нагодување на повеќе работи: земја или регион и јазик на тастатурата за внесување податоци.

(9) Прозорецот за поврзување во мрежа може да се прескокне и овој избор ќе го направиме по инсталацијата на Windows 10.

(10) Со следните два прозорци се дефинираат корисничкото име, лозинката и поим (анг. hint) кој треба да ни помогне да ја запаметиме лозинката. Без лозинка не ќе можеме да ја отвориме работната површина на оперативниот систем. Кај одредени инсталации, покрај корисничкото име, потребно е да се наведе и профилот на Microsoft, а доколку корисникот нема таков профил, може да го креира. Кај оперативниот систем Windows 10 има опција корисничките податоци и датотеки автоматски да се сочувуваат во облак – OneDrive. На таков начин, корисникот ќе има пристап до своите податоци и преку мобилен телефон.

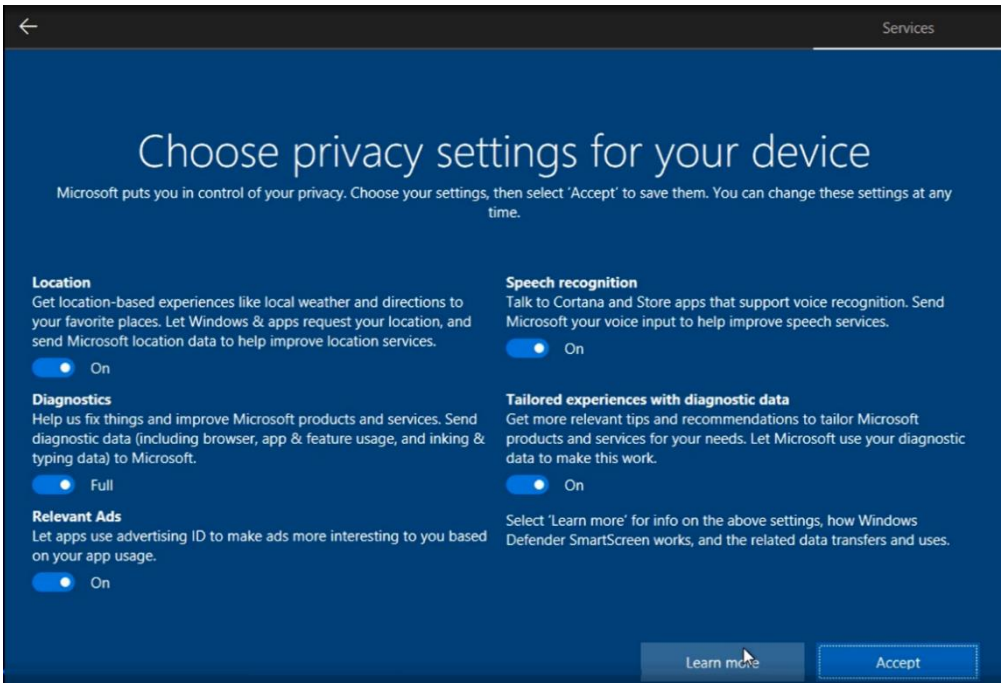
(11) Претпоследниот прозорец е за избор на апликацијата Кортана, како личен асистент.



Слика 3.11. Избор на апликација за личен асистент

Оваа апликација е новитет кај Windows 10; се користи за пребарување на интернет и за управување со хардверските компоненти, но не е достапна за сите земји.

(12) Последнот чекор од инсталацијата е поставување приватни ставки, како одредување локација, дијагностика на софтверот и апликациите, препознавање на говор и слично.



Слика 3.12. Поставување на можности за одредување локација и дијагностика

Инсталацијата е завршена и треба да се почека неколку минути додека да се отвори работната површина на Windows 10.

3.2.3. Практична вежба:Конфигурација на Windows 10 оперативен систем

По инсталацијата на оперативниот систем Windows 10, потребно е да се **конфигурираат уредите според личните барања**. Тоа може да го сториме со употреба на BIOS програмата или самиот оперативен систем.

Да се потсетиме **BIOS-от** е специјален софтвер кој овозможува комуникација меѓу хардверските компоненти и оперативниот систем. До него пристапуваме со притискање на едно од копчињата F1, F2 или Del после вклучувањето на напојувањето. После отворањето на BIOS-от се појавува основното мени за поставување (анг. Setup) во горниот дел од екранот. BIOS-от не поддржува работа со глувче и за навигација се користат копчињата од тастатурата: горе, долу, лево, десно. Менито за поставување е составено од следните подменија: главно, напредно, безбедност и подмени за инсталација на оперативниот систем и вклучување на напојувањето. Накратко ќе се запознаеме

со секое од подменијата. Да нагласиме дека менито за поставување е различно кај различни компјутери и зависи од видот на матична плоча.

Главно подмени (анг. Main) ⟶ Ова подмени дава информација за BIOS верзијата, видот на процесор и неговата работна фреквенција, капацитетот на меморијата. Тоа содржи алатки за менување на системското време и датум.

Напредно подмени (анг. Advanced) ⟶ Со ова подмени се врши поставување на мемориските уреди (хард-диск или SSD уред), приклучоците за графичката, звучната или мрежната картичка, USB уредите и другите уреди кои се директно поврзани на матичната плоча. Треба да бидеме многу внимателни и подготвени бидејќи неправилното поставување на одредени параметри во BIOS-от може да предизвика неработење на целиот систем.

Безбедност (анг. Security) ⟶ BIOS-от содржи две лозинки, супервизорска и корисничка. Супервизорската лозинка се нарекува лозинка за поставување бидејќи служи за пристап до самиот BIOS. Корисничката лозинка се нарекува системска лозинка и служи за пристап до оперативниот систем. Подменито за безбедност содржи алатки за промена или отстранување на лозинката за поставување. Во случај да се заборави лозинката тогаш сите лозинки можат да се избришат со употреба на посебна краткоспојница на самата матична плоча.

Подмени за поставување на оперативен систем (анг. Boot) ⟶ Да се потсетиме ова подмени се користи за подредување и избор на уред за инсталација на Windows оперативен систем

Излез (анг. Exit) Ова подмени служи за сочувување на направените измени и излез од BIOS програмата.

Windows 10 содржи две алатки за конфигурирање: Control Panel и новата апликација Settings. Подолу се набројани и објаснети категориите во апликацијата Settings.

Систем (анг. System) ⟶ Со алатките на оваа категорија може да се менува резолуцијата и осветленоста на екранот, да се ослободува простор за складирање податоци, да се избира режим на напојување итн.

Уреди (анг. Devices) → Оваа категорија дава листа на уреди со коишто може да комуницира компјутерот: Bluetooth-уреди, безжични екрани, екрани на допир, графички таблети, печатачи и слично.

Телефон (анг. Phone) → Овозможува пристап до содржината на нашиот Android или iPhone мобилен телефон.

Мрежа и интернет (анг. Network & Internet) → Може да се избере Wi-Fi или Ethernet мрежен адаптер, да се вклучи или да се исклучи авионскиот режим на работа, да се додаде нова виртуелна приватна мрежа (VPN) итн.

Персонализација (анг. Personalization) → За секој профил може да се одбере заднина, да ги промени боите на темата, да ја смени сликата за заштита на екранот (screen saver), да го измени изгледот на стартното менито или работната лента.

Апликации (анг. Apps) → Оваа категорија ги контролира програмите: бришење, менување на стартирачки (анг. start-up) програми, замена на стандардни програми, карактеристики на екран за видеоигри (анг. video mode).

Корисничка сметка (анг. Account) → Само администраторот може да креира кориснички профили. Кај Windows 10, можностите за семејна заштита и набљудување се многу поголеми.

Време и Јазик (анг. Time & Language → Се нагодуваат времето, датумот, регионот, јазикот и опциите за говор.

Лесен пристап (анг. Ease of access) → Оваа категорија нуди можности за различен пристап до екранот, тастатурата, глувчето, оратор за аудио објаснување на елементите на екранот.

Личен виртуелен помошник (анг. Cortana) → Ова е нова категорија и претставува личен виртуелен помошник за пребарување на интернет и искористување на компонентите.

Приватност, ажурирање и безбедност (анг. Privacy, Update & → Се користи за надградба на оперативниот систем, креирање резервна копија на личните документи (анг. backup), скенирање и откривање на вируси и потенцијално штетен софтвер, заштита од недозволен пристап и злоупотреба преку интернет, семејни опции. На пример, во категоријата Privacy → Camera може да

Security)	се изберат програмите што ќе имаат пристап до вградената камера на лаптопот.

Со текот на времето, перформансите на оперативниот систем се намалуваат од најразлични причини: некомпатибилност, вируси, апликации кои не се користат итн. **Цел на оптимизацијата е да се зголеми брзината на оперативниот систем**. Подолу се дадени неколку начини за остварување на оваа цел, нивното значење и чекорите за реализација.

Стартирачки апликации апликации (анг. Startup)	⟶	Startup апликациите автоматски се активираат при вклучување на компјутерот и го зголемуваат времето на чекање. Поради тоа е потребно да се редуцира нивниот број.	⟶ • Settings • Apps • Startup • Sort by: Startup impact • Избор на апликации • Ресетирање на компјутерот
Неискористени апликации	⟶	Оперативните системи Windows не даваат известувања (нотификации) за оптовареноста на системот, па поради тоа самите корисници треба да ги отстранат програмите што не ги користат.	⟶ • Settings • Apps • Apps & feature • Избор на апликации • Uninstall
Ослободување на меморија во хард дискот	⟶	Вообичаено откако ќе се постигне 70% искористеност на меморискиот простор во хард дискот. Во контролниот панел (Control Panel) оваа можност е под категоријата системско-административни алатки (System-Administrative Tools).	⟶ • Settings • System • Storage • Temporary files • Избор на датотеки • Removes files
Организација на податоци во хард дискот (анг. defragmentations)	⟶	Кога на хард дискот ќе се намали слободниот простор, тогаш податоците се запишуваат на различни места во хард дискот, со што се зголемува времето	⟶ • Settings • System • Storage • More settings • Optimize Drivers

	на вчитување. Со дефрегментизацијата, овие податоци се поставуваат еден до друг и се ослободува мемориски простор.		• Избираме диск (C: или D:) • Optimize
Антивирусни програми	Доколку оперативниот систем работи со намалена брзина, потребно е да се повика и да се активира антивирусната програма. Доколку компјутерот не располага со посебна антивирусна програма, може да се искористи програмата Windows Defender.	→	• Control Panel • Security • Windows Defender • Scan
Надградба на оперативен систем	Погрешно е кога корисниците ја исклучуваат автоматската надградба бидејќи надградбите се многу важни за сигурноста на системот. Драјверите на уредите можат да се надградуваат во категоријата System-Device manager.	→	• Settings • Update & Security • Windows Update • Check for Update • Ресетирање на компјутерот
Зголемување на виртуелните страници	Виртуелната меморија значи проширување на примарната меморија преку секундарната. Имено, RAM-меморијата зема податоци од хард-дискот во форма на виртуелни страници со фиксна големина и нив ги вметнува во виртуелна рамка. Ако се зголеми капацитетот на виртуелните страници, ќе се намали	→	• Control Panel • System • Advanced System (Performance) • Settings • Advanced • Change (Virtual memory)

бројот на трансфери меѓу
RAM и хард-диск.

Оперативниот систем Windows 10 нуди многу можности и не можеме сите да ги објасниме и проучиме. Но, секогаш треба да се истражува и да се настојува самостојно да се конфигурираат компјутерските системи.

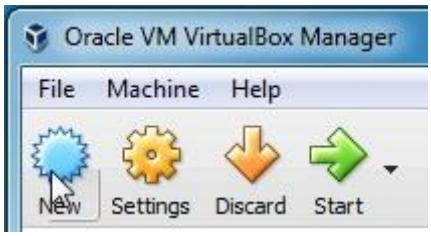
3.2.4. Практична вежба: Инсталација на оперативен систем Ubuntu 17 Linux во виртуелна машина

3.2.4.1. Упатство за креирање на виртуелна машина

Ubuntu е многу популарен дистрибутер на оперативните системи Linux. За преземање на инсталацијата на оперативниот систем, потребно е да се отвори www.ubuntu.com на овој дистрибутер и потоа со кликнување на Download, ја избираме десктоп-верзијата.

VirtualBox е апликација за креирање, конфигурација и администрација на виртуелни машини. Виртуелна машина претставува компјутерски системски гостин (guest) што ги користи ресурсите на компјутерот-домаќин (host). На компјутерот-домаќин ќе биде инсталиран оперативниот систем Windows, а на виртуелната машина, односно guest-компјутерот ќе биде инсталиран оперативниот систем Ubuntu. На таков начин, **на ист компјутер** ќе имаме инсталирано **два оперативни система** без да си влијаат еден на друг.
Програмата за инсталирање виртуелна машина се презема од сајтот www.virtualbox.org, со кликнување на опцијата Download и избор на линкот Windows host.

Откако ќе ја инсталираме апликацијата VirtualBox, ја отвораме и се кликнува на опцијата New за креирање нова виртуелна машина.

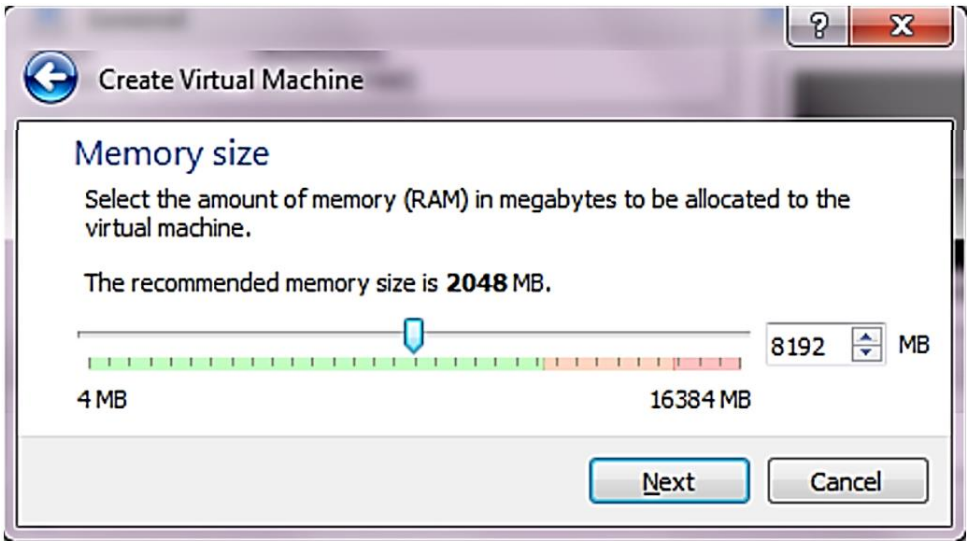


Слика 3.13. Креирање виртуелна машина

Се отвора прозорецот **Create Virtual Machine**. Во полето **Name** се внесува името на виртуелната машина (на пример Ubuntu), во полето **Type** се внесува видот на оперативниот систем што ќе се инсталира во виртуелната машина

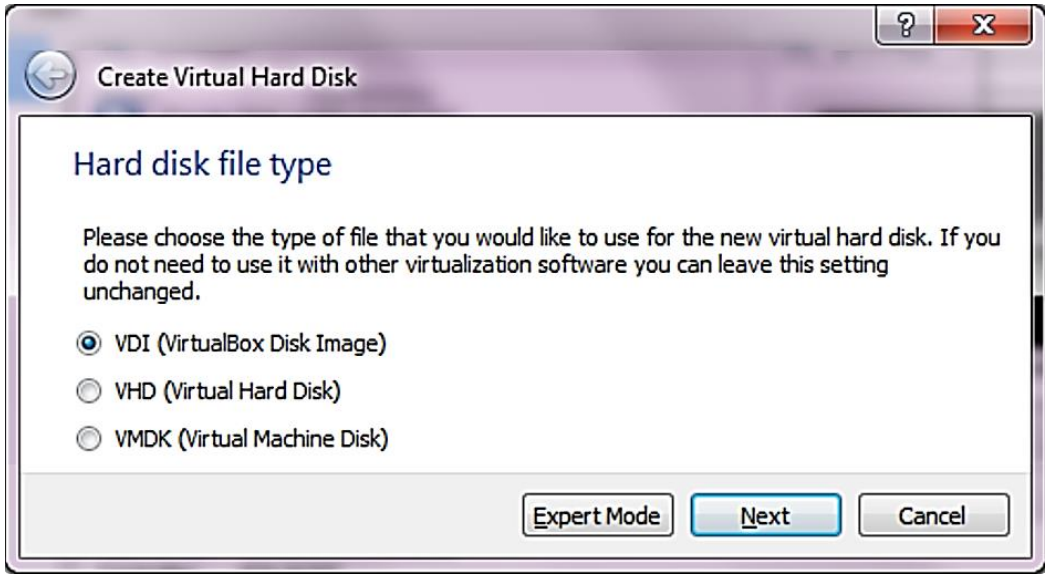
(Linux) и во полето **Version** се внесува верзијата на оперативниот систем (32-битен или 64-битен). На крајот се кликнува на Next.

Следниот прозорец служи за определување на капацитетот на RAM-меморијата за виртуелната машина. Најдобро е да се следат препораките, но треба да се земе предвид и вкупниот капацитет на меморијата.



Слика 3.14. Определување на капацитетот на RAM-меморијата за виртуелната машина

Следните три прозорци служат за креирање **виртуелен хард-диск**, дефинирање на неговиот капацитет и определување на начинот на управување со меморискиот простор.



Слика 3.15. Избор на вид на хард-диск

Слика 3.16. Избор на капацитет на хард-диск

Креирањето на виртуелната машина го потврдуваме со кликање на Create. Во овој последен чекор имаме можност да ги промениме името и локацијата на виртуелната машина.

3.2.4.2. Инсталација на оперативниот систем Ubuntu

За инсталација на оперативниот систем Ubuntu кликаме на опцијата Start во прозорецот VirtualBox Menager, прикажан на слика 3.17.

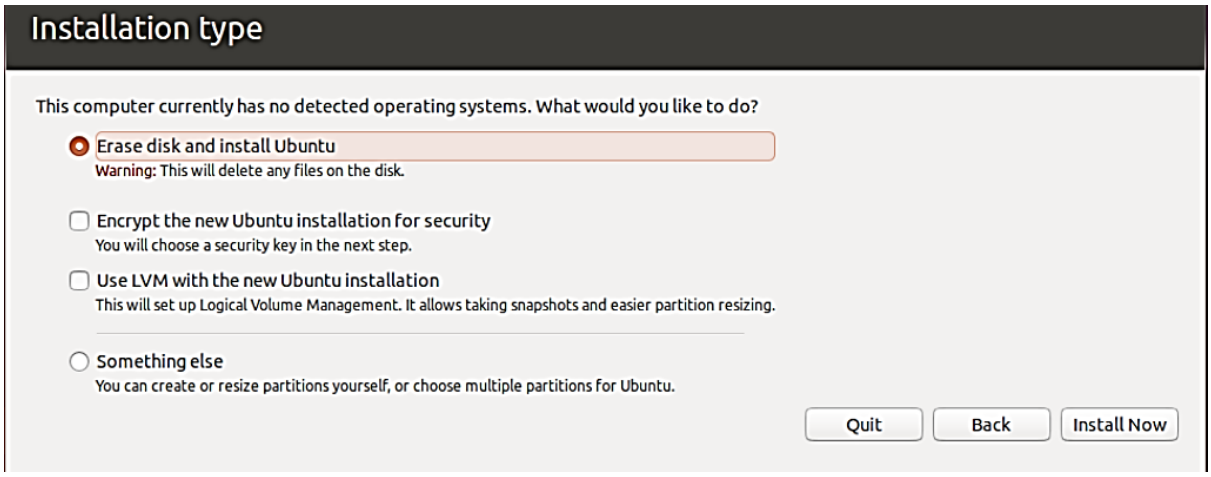
Слика 3.17. Почеток на инсталација на оперативен систем Ubuntu

Во прозорецот Select start-up disk, со кликање на жолтата папка (фолдер) ја наоѓаме локацијата на којашто се наоѓа инсталацијата на оперативниот систем, ја избираме и кликаме на копчето Start.

Во прозорецот Welcome, дадени се **два начина за инсталација** на оперативниот систем Ubuntu. Со начинот Try Ubuntu, оперативниот систем може да се користи без инсталација. Со овој начин, всушност, го инсталираме оперативниот систем на виртуелниот хард диск којшто претходно е креиран и пред инсталацијата е празен, така што нема бришење на постоечкиот оперативен систем.

Слика 3.18. Избор за начин на инсталација на оперативен систем Ubuntu

Во прозорецот **Installation type**, со кликање на копчето **Install Now започнува инсталацијата** . Опцијата **Erase disk and install Ubuntu** е однапред обележана, но веќе објаснивме дека бидејќи се работи за виртуелен хард-диск, нема да дојде до бришење и губење на податоците.

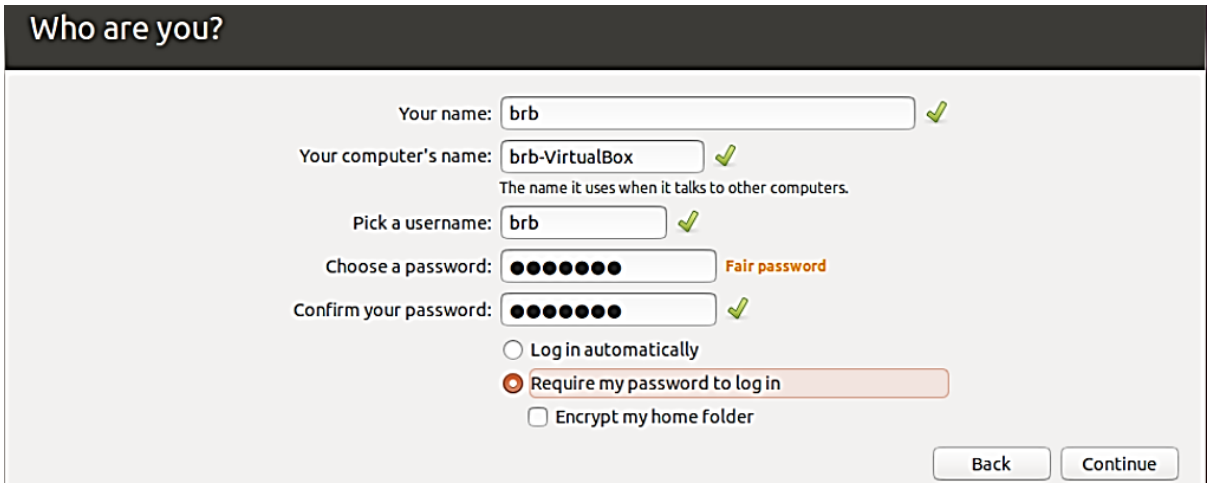


Слика 3.19. Поради виртуелната машина, нема да дојде до бришење на постоечкиот оперативен систем

Следниот прозорец е Preparing to Install Ubuntu и во него постои можност да се избере автоматска надградба на оперативниот систем и инсталација на дополнителен мултимедијален софтвер за мултимедија. На крајот кликаме на копчето Continue.

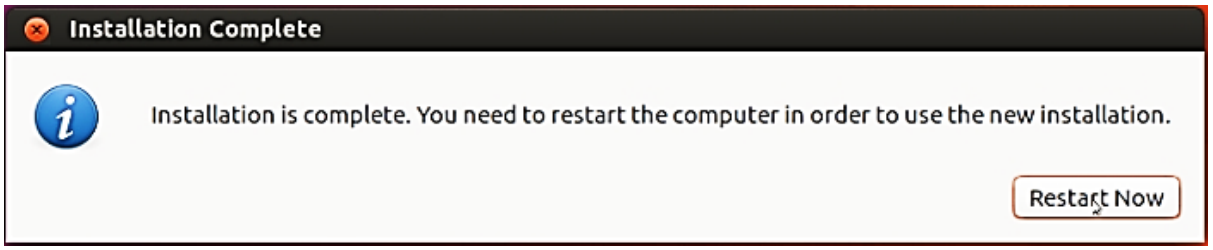
Следните два прозорци се за избор на земја или регион и јазик на тастатурата за внесување податоци, слично како и при инсталацијата на оперативниот систем Windows 10.

Следува прозорец за определување на корисничкото име и лозинката. Со кликнување на Continue, започнува инсталацијата.



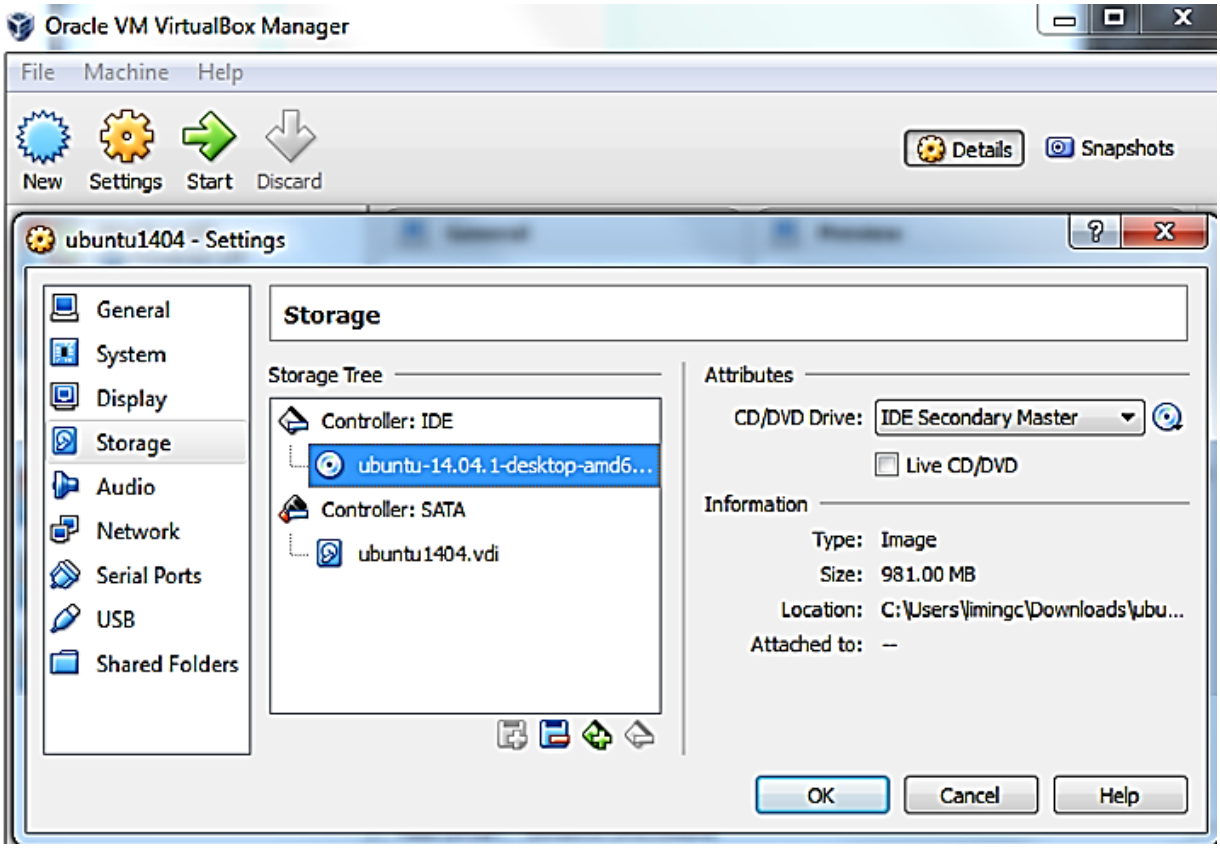
Слика 3.20 Избор на корисничко име и лозинка

По информацијата за успешната инсталација, го рестартираме системот и ја добиваме работната површина на Ubuntu.



Слика 3.21. Рестартирање на компјутерот за да се добие работната површина на Ubuntu

По извршената инсталација, потребно е да го избришиме инсталацискиот документ. За таа цел се избира оперативниот систем Ubuntu во виртуелната машина, се притиска на опцијата Storage во категоријата Settings. Потоа во делот Storage Tree треба да се избере инсталацискиот документ и да се притисне (анг. click) Remove selected storage attachment. На крајот го притискаме копчето OK.



Слика 3.22. Бришење на инсталациски документ

3.2.5. Практична вежба: Инсталација и конфигурација на Raspbian оперативен систем за Raspberry Pi

За разлика од Arduino, Raspberry Pi е микрокомпјутер со оперативен систем. Оперативниот систем дозволува инсталација и менаџирање на најразлични апликации, што го прави Raspberry Pi софтверски многу моќен. За Raspberry Pi постојат повеќе видови оперативни системи, но за почеток ќе го користиме **оперативниот систем Raspbian**. [7]

3.2.5.1. Преземање на NOOBS софтвер за инсталација

За инсталација на оперативниот систем Raspbian ќе користиме софтвер со симболично име Нов софтвер од кутија (анг. NOOBS- New Out Of the Box Software). Тоа е специјален софтвер што овозможува избор на еден од повеќе оперативни системи и автоматска инсталација со неколку кликнувања на глувчето. Софтверот за инсталација се сочувува на микро SD-картичка. Веќе

спомнавме дека Raspberry Pi нема вградена трајна меморија, туку за чување на податоците и на оперативниот систем се користи **SD-картичка со минимум 16 GB меморија.**

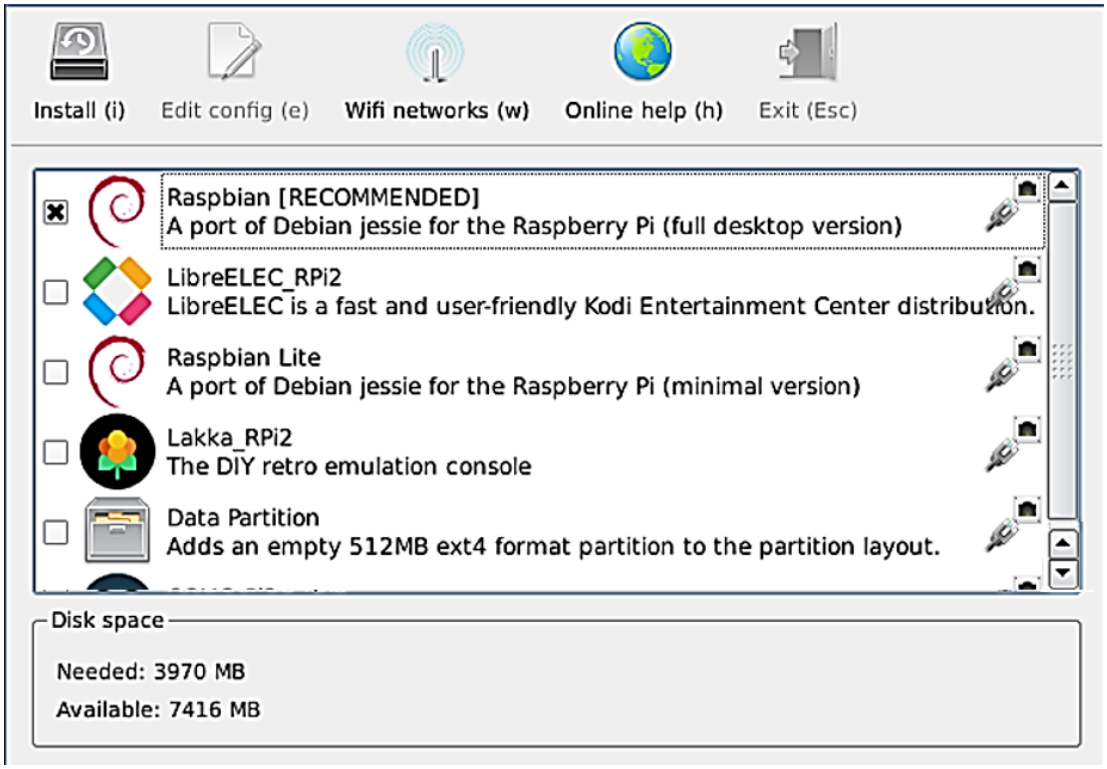
За да го преземеме софтверот NOOBS, во веб-пребарувачот треба да се напише rpf.io/download. На страницата што ќе се отвори, во категоријата Download, прво се притиска на NOOBS со логото Raspberry, а потоа се притиска (click) на Download Zip, што се наоѓа под „NOOBS offline and network install“.

Слика 3.23. Интернет-страница за преземање на инсталациски документ за NOOBS

Ја внесуваме микро SD-картичката во соодветниот слот на персоналниот компјутер. Да напомниме дека доколку не се работи за нова картичка, таа треба да се форматира, при што треба да се води сметка за податоците што биле запишани на неа. Потоа во папката Преземања (анг. Downloads) ја бараме компресираната датотека со инсталацијата. Овој датотека е позната како архивска датотека (анг. archive) и содржи голем број посебни документи. Со двоен клик го „отпакуваме“ компресираниот документ, ги селектираме сите документи во него и нив ги префрламе, ги копираме во микро SD-картичката.

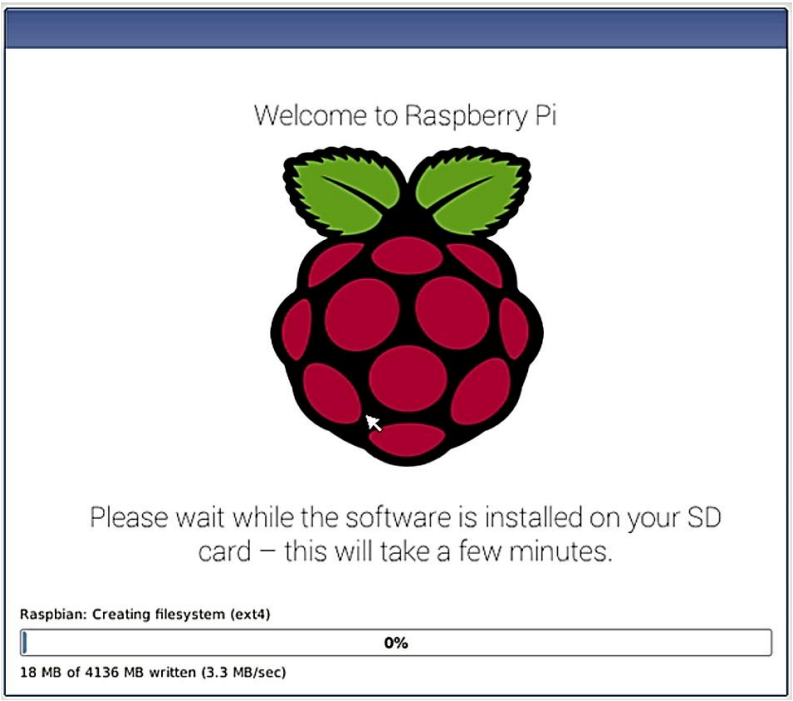
3.2.5.2. Инсталација на Raspbian оперативен систем

Потоа **микро SD-картичката** со инсталираниот софтвер NOOBS **треба да ја извадиме од персоналниот компјутер и да ја вметнеме во слотот на микрокомпјутерот** Raspberry Pi. Кога картичката за првпат ќе ја поврземе со Raspberry Pi, се појавува менито на NOOBS прикажано на слика 3.24. Со глувчето **го бележиме** квадратчето на оперативниот систем што сакаме да го инсталираме, а тоа во овој случај е **Raspbian**. Ќе забележиме дека иконата Install ќе добие боја, што значи дека оперативниот систем е подготвен за инсталација.



Слика 3.24. Избор на оперативен систем Raspbian

Со притискање на иконата Install, започнува процесот на инсталација, при што се бришат сите податоци од микро SD-картичката, освен софтверот NOOBS. Самата инсталација може да трае од 10 до 30 минути.



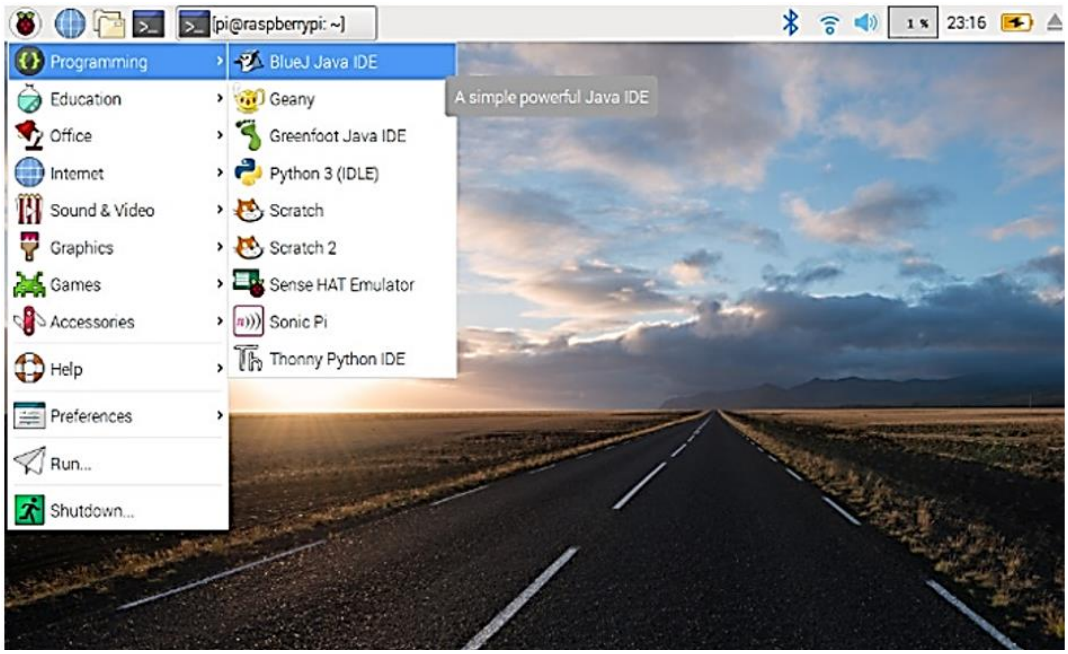
Слика 3.25. Инсталација на оперативен систем Raspbian

По завршувањето на инсталацијата притискаме (анг. click) на ОК, и уредот Raspberry Pi се рестартира. Само при првото отворање на оперативниот систем се појавуваат boot-пораки, кои траат една до две минути. Конечно се појавува работната површина на Raspbian и во неа прозорецот со којшто се најавува конфигурацијата на оперативниот систем. Кликнуваме на Next.

Слика 3.26. Прозорец за најава за инсталација

Исто како при инсталацијата на оперативните системи Windows или Linux, така и при инсталацијата на оперативниот систем Raspbian избираме: земја, јазик, временска зона, лозинка, избор на Wi-Fi мрежа и надградба на оперативниот систем. Во последниот прозорец треба да се притисне Reboot, по што системот се ресетира и се конфигурира со направените измени.

Работната површина на оперативниот систем Raspberry изгледа многу слично со работната површина на оперативниот систем Windows. Во горниот десен агол се наоѓа лентата со алатки за: Bluetooth, избор на мрежа, јачина на звук, мониторинг на процесор, часовник, вклучување и исклучување на USB-уреди. Во левиот горен агол се наоѓа Raspberry и со нејзино притискање се отвора паѓачко мени со повеќе категории. На пример, категоријата **Programming** содржи развојни програми меѓу кои е **Thonny Python IDE**. За пребарување на интернет, во категоријата Internet го избираме Chromium Web Browser. За работа со документи и датотеки, потребен ни е File Manager и за негово активирање ја избираме категоријата Accessories. Во категоријата Office се наоѓа програмата LibreOffice Writer за обработка на текст. Може да се направи инсталација на дополнителен софтвер со притискање на Recommended Software во категоријата Preferences. За додавање нов хардвер (на пример, софтвер за работа со камера, анг. Raspberry Pi Camera Module) или за поврзување во мрежа (анг. VNC- Virtual Network Computer) може да се употреби опцијата Interface, до која доаѓаме преку притискање (click) на Raspberry Pi Configuration во категоријата Preferences.



Слика 3.27. Изглед на работната површина на оперативниот систем Raspbian

3.2.6. Практична вежба: Инсталација на развојна средина за Arduino платформа и ставање во употреба

3.2.6.1. Упатство за инсталација на развојна средина за Arduino платформа

Arduino платформата не располага со оперативен систем и затоа неа не можеме да ја програмираме без да ја **поврземе со персонален компјутер.** За пишување и внесување на програмите во микроконтролерот на платформата Arduino, потребна ни е развојна програма, или уште позната како **развојна средина (**анг. IDE – Integrated Development Environment). Заради подобро сфаќање на развојната средина, ќе објасниме неколку видови алатки што спаѓаат во овој вид на програми:

- Едитори се програми што овозможуваат внесување, менување, обликување, паметење и печатење текст на програмата;
- Преведувачите ги преведуваат програмите напишани во програмски јазици од повисок ред во машински јазик. Преведувачите се познати и под името компајлери.

- Дебагери се програми за отстранување грешки. Се поставуваат контролни точки на одделни места во програмата и во текот на програмата се застанува на овие точки и се испитува состојбата на програмата;
- Поврзувачи се програми што вршат поврзување на новиот софтвер со стариот, што е многу важно за негово извршување.

Развојната програма се презема од **официјалната веб-страница на Arduino,** поточно од следниов линк https://www.arduino.cc/en/software . Може да одбереме една од неколкуте различни верзии на развојни програми Arduino, но доколку користиме оперативен систем Windows 10, најдобро е да се преземе најновата верзија. Да нагласиме дека оваа верзија нема да функционира на постар оперативен систем, како што се Windows XP или Windows 7. По преземањето на документот за инсталација, ја инсталираме развојната програма. Притискаме (click) на **„Run“,** ги прифаќаме условите за лиценцирање, избираме папка за чување на програмата и на крајот притискаме „Install“. [6]

По инсталацијата, ја поврзуваме платформата Arduino Uno R3 со персоналниот компјутер преку USB-кабел со два различни приклучоци, од A и B-тип. Платформата добива напојување од персоналниот компјутер и светнува зелената вградена диода обележана со буквата L (слика 2.10.). Кога уредот Arduino Uno R3 за првпат го поврзуваме со компјутер**, автоматски се инсталира драјвер**. Драјверот е софтвер што му овозможува на компјутерот да комуницира со новиот уред.

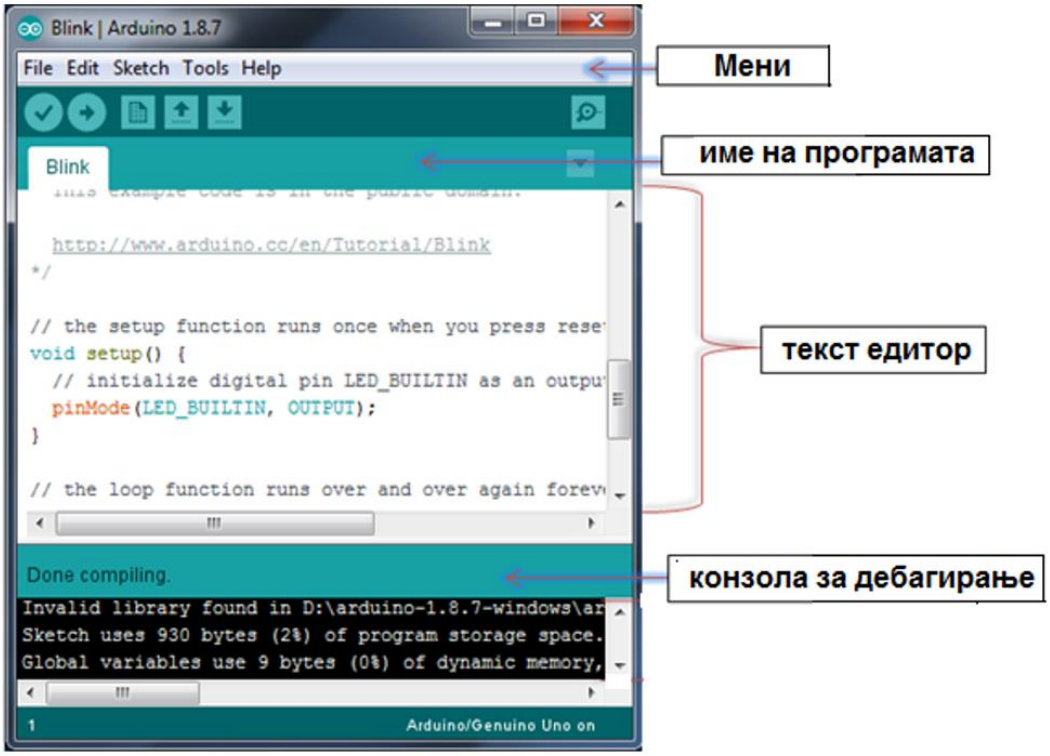
Ако се работи за постара верзија на оперативен систем Windows, може да се појави прозорец со барање да се посочи локацијата од каде што треба да се инсталира драјверот. Ако не започне инсталацијата, потребно е да се отвори програмата **Control Panel**, да се избере категоријата **Device Manager**, потоа поткатегоријата **„Other Devices“ или „Unknown Devices“,** и да се притисне (анг. click) **„Update Drivers“ или „Update Driver Software“.** Потоа го посочуваме драјверот за поврзување на Arduino платформата, така што се избира папката „Drivers“ во состав на самата Arduino папка.

Како потврда за успешната инсталација, во категоријата Device Manager, со притискање на опцијата „PORT (COM&LPT)“ треба да се појави Arduino Uno. Ако не знаеме на која порта е поврзана платформата Arduino Uno тогаш можеме да ја исклучиме платформата од компјутер, а потоа повторно да ја вклучиме и да видиме која од сериските порти ќе се појави како нова.

Платформата Arduino користи програмски кодови од отворен тип, што значи програмите се бесплатни. Секој програмер може да ги објави своите програмски кодови и на таков начин да придонесе за развој на оваа платформа. Самата развојна средина за Arduino содржи многу готови програми и листата на овие програми можеме да ја видиме со притискање File→**Exampes.** Готова програма можеме да преземеме и од некоја Arduino веб-страница и да ја отвориме со притискање File→Open→Look in и на крај фолдерот во кој сме ја сочувале програмата.

3.2.6.2. Мени и алатки на развојна средина и впишување на програма во Arduino Uno R3

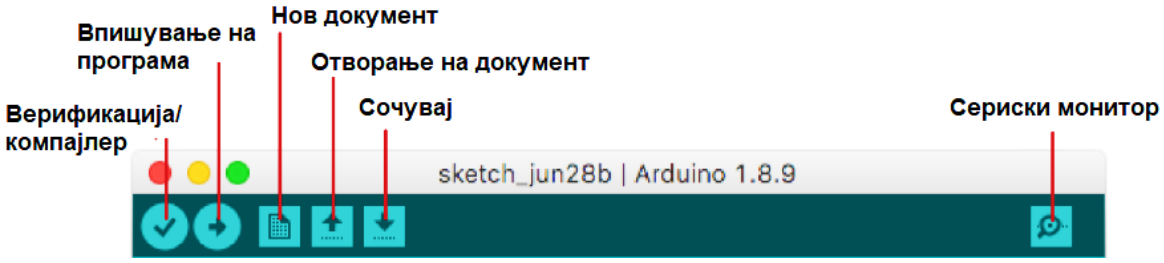
На слика 3.29. е прикажана развојната програма за платформата Arduino**. Поважни составни делови се: менито, лентата со алатки, уредувачот на текст, серискиот монитор, библиотеките и конзолата за дебагирање.**



Слика 3.29. Изглед на развојна средина за Arduino развојна платформа

Мени ⟶ Во менито се наоѓаат следниве категории: File, Edit, Sketch, Tools, Help.

Лента со алатки ⟶ Во лентата со алатки се наоѓаат копчињата: верификација/компајлер (анг. Verify/Compile), прикачување т.е. впишување (анг.Upload), отворање на нов документ (анг. New), отворање на стар документ (анг. Open) и сочувај (анг. Save). Со алатката Verify се проверува кодот, ред по ред, со цел да се откријат евентуалните грешки. Со алатката Upload, верификуваниот код се впишува во програмската меморија на микроконтролерската платформа.



Слика 3.28. Лента со алатки за Arduino развојна средина

Уредувач на текст → Ова е најважниот дел од развојната програма бидејќи во него ги пишуваме кодовите, програмите во програмскиот јазик C/C++.

Сериски монитор → Со негово притискање се овозможува преглед на податоците што се испраќаат или се добиваат од Arduino платформата. Има голема примена кај апликациите во реално време.

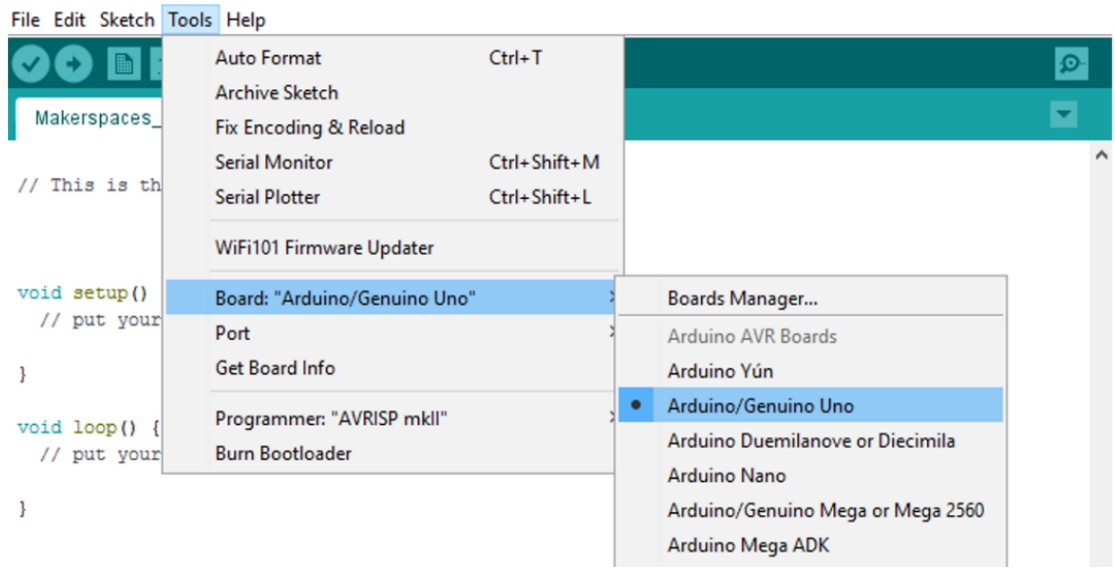
Конзола за дебагирање → Овој прозорец ни дава информација за успешното компајлирање или укажува на редниот број на редицата што предизвикува грешка во компајлирањето. Компајлирањето е постапка на преведување на програмата од програмски јазик од повисок ред во машински јазик.

Библиотеки (libraries) → Со притискање (анг. click) на оваа категорија, се отвора листа на потпрограми за работа со најразлични влезно-излезни уреди.

После инсталирањето на развојната средина и запознавањето со менито и алатките ќе ја објасниме постапката за впишување на програма во меморијата на Arduino Uno R3 платформата. За таа цел ќе ја искористиме вградената лед диода која е поврзана со 13 пин на оваа развојна платформа. Исто така ќе употребиме готова програма која не бара употреба на други електронски компоненти. Оваа вежба е една од наједноставните вежби и претставува еден вид тест за работата на Arduino Uno R3.

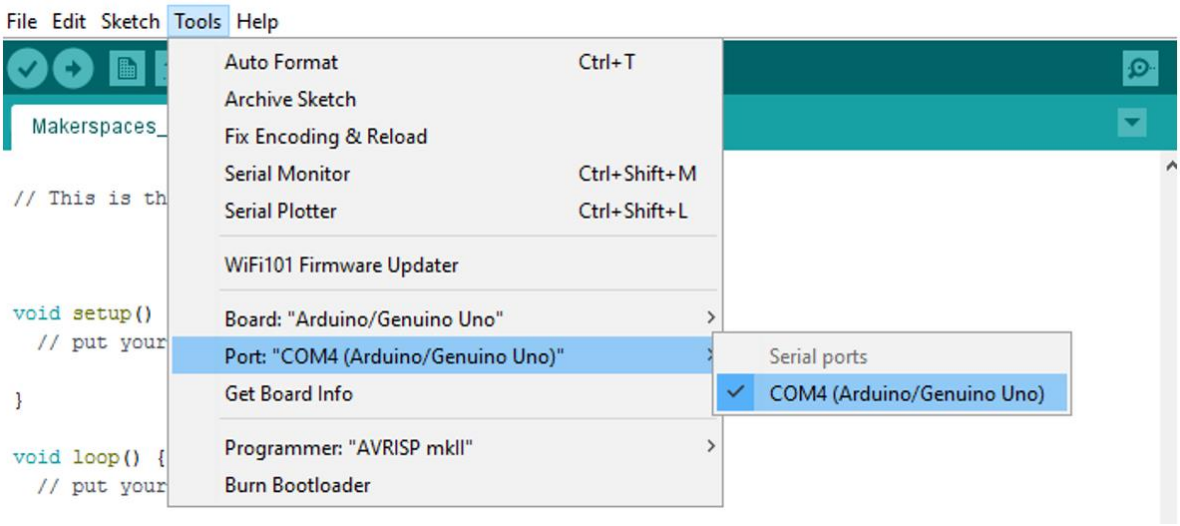
1. По инсталацијата, го поврзуваме Arduino Uno R3 со персоналниот компјутер преку USB-кабел со два различни приклучоци, од A и B-тип.

2. Прв чекор е избор на микроконтролерска платформа и избор на сериска порта. Да се потсетиме, фамилијата Arduino беше составена од повеќе различни платформи: Uno, Mega, Nano, Leonardo и др. За избор на платформа треба да се притисне на опциите Tools → Board →Arduino Uno. Ова е прикажано на слика 3.30.



Слика 3.30. Постапка за избор на платформа Arduino

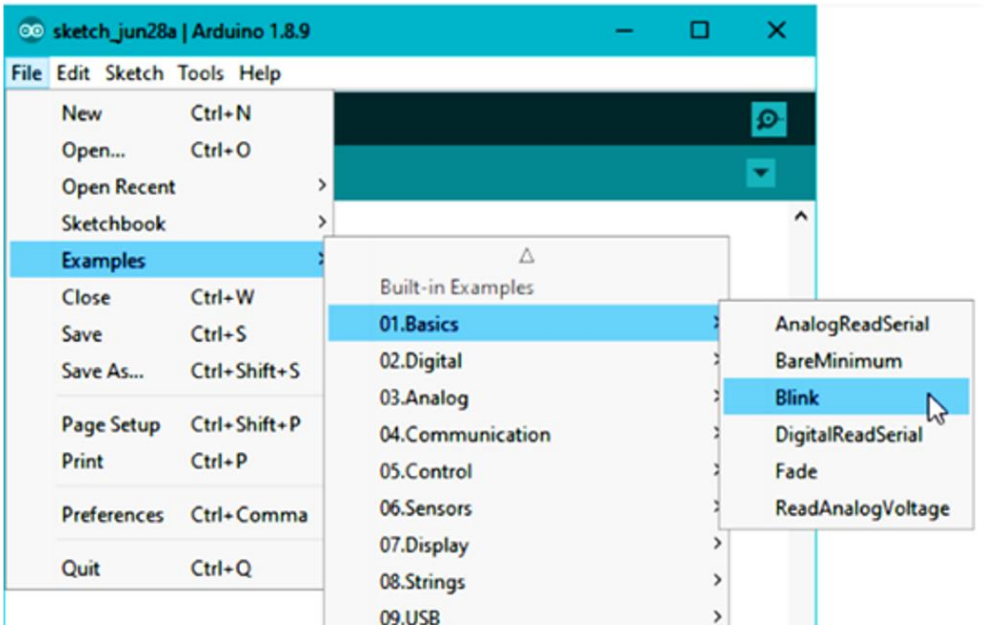
3. За избор на порта, потребно е да се притисне опцијата Tool→Serial Port. Ова е прикажано на слика 3.31. При тоа се појавува паѓачко мени на достапни сериски порти и тоа се разликува во зависност од уредите кои се поврзани со компјутерот.



Слика 3.31. Постапка за избор на порта за Arduino Uno

4. Програмскиот код го наоѓаме со притискање на File→Example, а потоа избираме 01. Basic→Blink. (слика 3.32.)

5. Пред да се впише програмата во меморијата на Arduino Uno R3 истата треба да се преведе на машински јазик односно компајлира. За таа цел во листата на алатки го притискаме копчето Verify/Compile или во менито избираме Sketch→ Verify/Compile



Слика 3.32. Отворање на програмата Blink за Arduino Uno R3 платформа

6. Во конзолата за дебагирање добиваме известување “Compiling Sketch….” и се појавува лентата за напредок на постапката и на крај треба да се добие известување “Done Compiling” (слика 3.33.).

Слика 3.33. Известување за успешно компајлирање

7. Ако нема грешки во програмскиот код, во конзолата за дебагирање добиваме информација за големината на програмата што сме ја отвориле изразена во бајти и информација за големината на останатиот слободен меморискипростор .

8. Со притискање на копчето за прикачување (анг. Upload) започнува впишувањето на програмата во меморијата. Вградената лед-диода со ознака L ќе престане да свети, а ќе започнат да светат двете диоди за сериски пренос, со ознаки RX и TX (види слика 2.10.). Впишувањето трае неколку секунди после што почнува да трепка (анг. blink) лед-диодата со ознака L.

Заклучоци

Оперативниот систем е познат под името контролно управувачки софтвер. Софтверот опфаќа програми за распределба на хардверските ресурси, контрола на влезно- излезните операции, управување со меморијата, управување со податоците, преведување на програмски јазици.

Предности на Windows оперативниот систем се: едноставност во работењето (анг. user friendly), најголема поддршка за поврзување со периферни уреди, достапност до апликативен софтвер, автоматска инсталација (plug and play), најдобри перформанси за видеоигри, компатибилност со најголем број на веб страници. Недостатоци се: потреба од скапи компјутерски конфигурации, затворен код, слаба сигурност во поглед на интернет напади и административна заштита, чувствителност на компјутерски вируси, потребна е лиценца, слаба експертска поддршка и често пати за да се зголеми брзината на работа потребна е преинсталација на оперативниот систем.

Отворен или слободен код значи сите корисници да имаат пристап до изворниот код и да можат да го менуваат во зависност од своите потреби.

Android на компанијата Google и оперативниот систем Raspbian се примери за примена и проширување на јадрото на оперативниот систем Linux.

Предности на оперативниот систем Android се: огромен број на бесплатни апликации, компатибилност со уреди од различни производители, отворен код, располага со едноставни развојни средини за создавање на нови апликации, меморискиот простор лесно се проширува, можност за споделување на интернет, едноставна комуникација со многу уреди, слобода при конфигурација и избор на апликации. Недостатоци на овој оперативен систем се: апликациите се потешки за изработка поради различните димензии на екраните, помала брзина поради позадинските апликации, помала сигурност и осетливост на вируси.

Инсталацијата на оперативниот систем Windows може да се изврши на различни начини, преку мемориски модул (USB или DVD) и преку интернет (анг. PHE boot). Доколку користиме мемориски модул потребно е после приклучувањето на модулот и вклучувањето на компјутерот да ја активираме BIOS програмата.

Клуч на продукт е код од 25 цифри со кои се потврдува лиценцата. Овој код го испорачува Microsoft заедно со доставата на оперативниот систем Windows или може да биде испратен преку е-пошта доколку програмата за инсталација е превземена од интернет.

Постојат две можности за инсталација на оперативен систем Windows: Upgrade или Custom. Првата опција значи надградба на веќе постоечкиот оперативен систем Windows. Оваа инсталација е бесплатна, не бара клуч на продукт , при што не се бришат корисничките податоци.

Windows 10 содржи две алатки за конфигурирање: Control Panel и новата апликаација Settings.

Категории во апликацијата Settings се: System, Devices, Phone,Network&Internet, Personalization, Apps, Account, Time & Language, Ease of access, Cortana и Privacy, Update &Security.

Начини за оптимизација на оперативниот систем односно зголемување на брзината на работа на компјутерот се: отстранување на неискористените и стартирачки апликации, ослободување на меморија во хард дискот и негова дефрегментизација, употреба на антивирусни програми, надградба на оперативен систем и зголемување на виртуелните страници.

Виртуелна машина преставува компјутерски систем-гостин (guest) кој ги користи ресурсите на компјутерот-домаќин (host). Во компјутерот-домаќин ќе биде инсталиран оперативниот систем Windows, а во виртуелната машина односно компјутерот-гостин ќе биде инсталиран оперативниот систем Ubuntu.

Во прозорецот Create Virtual Machine, во полето Name се внесува името на виртуелната машина (на пример Ubuntu), во полето Type се внесува видот на оперативен систем кој ќе се инсталира во виртуелна машина (Linux) и во полето Version се внесува верзијата на оперативен систем (32- битен или 64-битен).

При инсталација на оперативен систем Linux во виртуелна машина инсталацијата започнува со кликање на копчето Install Now, во прозорецот Installation type. Erase disk and install Ubuntu опцијата е однапред обележана, но веќе објаснивме дека бидејќи се работи за виртуелен хард-диск нема да дојде до бришење и губење на податоци.

Веќе спомнавме дека Raspberry Pi нема вградена трајна меморија туку за чување на податоците и оперативниот систем се користи SD картичка со минимум 16GB меморија. NOOBS (New Out-Of-Box Software) е специјален софтвер кој овозможува избор на еден од повеќе оперативни системи за Raspberry Pi и автоматска инсталација со неколку клика на глувчето. Меѓу понудените оперативни системи е оперативниот систем Raspbian.

Категоријата Programming во менито на оперативниот систем Raspbian содржи развојни програми меѓу кои е Thonny Python IDE.

Развојната програма за Arduino микроконтролерските платформи се презема од официјалната веб страна на Arduino поточно од следниов линк arduino.org/download. После преземањето на инсталацијата истата ја инсталираме. Го притискаме (click) „Run”. Ако не започне инсталацијата

потребно е да се отвори програмата Control Panel, да се избере категоријата Device Manager, потоа поткатегоријата „Other Devices” или „Unknown Devices” и да се притисне (click) „Update Drivers” или „Update Driver Software”.

Прашања за повторување

1. Кои програми влегуваат во состав на оперативниот систем?
2. Кои се предностите и недостатоците на оперативниот систем Windows?
3. Кои се почетоците на развој на оперативниот систем Linux?
4. Наведи примери за примена на оперативниот систем Linux!
5. Објасни ја постапката за избор на уред за инсталација на оперативен систем Windows 10!
6. Што претставува таканаречениот клуч на продукт за оперативен систем Windows?
7. Објасни ги инсталациските можности Upgrade и Custom за оперативен систем Windows 10!
8. Кои се двете апликации за конфигурација на оперативен систем Windows 10?
9. Набројг ги категориите во апликацијата Settings!
10. Која категорија од апликацијата Settings се користи за промена на резолуција и осветленост на екран?
11. За што служи категоријата Account во апликацијата Settings?
12. Во која категорија од апликацијата Settings се наоѓаат програмите за откривање на вируси и заштита од недозволен пристап?
13. Која е основна цел на оптимизацијата на оперативниот систем?
14. Што подразбираме под поимот стартирачки (start-up) апликации?
15. Објасни ја постапката за дефрегментизација на хард-диск?

16. Во која категорија се надградуваат драјверите за периферните уреди?

17. Која е намената на апликацијата VirtualBox?

18. Објасни што се внесува во полињата Name, Type и Version при креирање на нова виртуелна машина?

19. Набројте кои категории се наоѓаат во менито на работната површина на оперативниот систем Raspbian?

20. Објасни ја постапката за инсталација на драјвер после инсталацијата на развојната програма и поврзувањето на Arduino платформа со персонален компјутер?

21. Набројте ги составните делови на развојната средина на Arduino платформата?

22. За што служи серискиот монитор во состав на развојната средина на Arduino платформа?

4. Програмирање на микрокомпјутери

4.1. Програмски јазик C/C++ и негова примена

Денес, програмирањето е предизвик за многу млади луѓе, токму поради појавата на многу софтверски алатки, развојни средини кои се лесни за употреба (анг. user friendly) и коишто овозможуваат да се создадат апликации без примена на сложени математички модели. Постојат голем број програмски јазици и секој од нив има различна намена и цел. Ние ќе се запознаеме со C/C++ и Python, кои се најчесто користени програмски јазици за апликативен софтвер, особено во областа на техничките науки. **Програмите** напишани **во** програмскиот јазик C/C++ се со чиста структура, можат лесно да се приспособуваат според потребите и **се дел** од оперативните системи, графичкиот кориснички интерфејс и вградливите програмабилни системи (анг. embedded systems)**.**

Во третата тема се запознавме со начинот на инсталирање на интегрираната развојна средина за Arduino платформата и нејзините системски програми. На почетокот, се користат готови програмски кодови, но за понатамошна успешна работа неопходна е анализа на истите, со цел нивно менување и приспособување. Предизвик е учениците да научат да составуваат свои програми согласно карактеристиките на електронските компоненти кои ќе ги користат.

Да споменеме дека Arduino платформата може да извршува само една програма и нејзиното извршување започнува веднаш по вклучувањето на напојувањето. За полесно совладување на програмирањето на Arduino платформата се препорачува употреба на Tinkercad симулаторот кој беше објаснет во наставната единица 2.8.3.3.Компјутерска симулација за Arduino Uno R3 платформа. За следење на резултатите, добиени од извршувањето на програмата, може да се користи серискиот монитор кој е составен дел од симулаторот или развојната средина на Arduino платформата.

За програмите напишани за Arduino се користи називот скица (анг. sketch). На ваков начин се нагласува едноставната структура на програмите, за брза и

лесна реализација на идеите. Скицата е составена од инструкции. Инструкциите се искази што иницираат одредено дејство за да се добие посакуваниот резултат. Преку инструкциите, ние ја контролираме Arduino платформата и вршиме пресметки. [6]

Скицата содржи две задолжителни структури: setup и loop. Пример 4.1 е општ приказ на двете задолжителни структури во скицата.

setup() ⟶ Setup е функција без влезни аргументи. Со неа **се задаваат почетни вредности на променливите, се конфигурираат пиновите** како влезни или излезни, **се избира брзината за пренос на серискиот монитор, се специфицираат вклучените библиотеки**. Оваа функција се извршува само еднаш, кога ќе се вклучи напојувањето или по рестартирањето на Arduino платформата.

loop() ⟶ Со оваа функција, всушност, вршиме контрола на Arduino платформата. Англискиот збор loop во превод значи јамка и како што самото име покажува, оваа функција постојано се извршува, се врти во круг, **постојано ги следи промените на состојбата на влезните пинови и соодветно реагира.** Всушност функцијата loop претставува бесконечен циклус кој се повторува сè додека Arduino има напојување.

Пример 4.1.

```
1  int buttonPin = 3;                 // Прво се декларираат променливите.
2  void setup() {                     // setup е секогаш почетна функција.Ознаката void
                                      // значи дека функцијата нема да даде повратна
                                      // вредност.
3    Serial.begin(9600);              // Дефинираме брзината за пренос на серискиот
                                      // монитор.
4    pinMode(buttonPin, INPUT);       // Конфигурација на влезен пин.
5  }
6  void loop() {                      // Почеток на функцијата loop.
   .......
   }                                  // Крај на функцијата loop.
```

Но, пред да започнеме со креирање и анализа на програми за Arduino платформата, ќе се запознаеме **со основните градбени елементи** на програмскиот јазик C/C++: променливите, инструкциите и структурите. Потребно е да се разбере нивното значење за компјутерот и да се запамети нивната синтакса. **Синтакса** во програмирањето значи множество правила за подредување на ознаките, симболичните имиња, операторите, интерпункциските знаци, коментарите, со цел да се добие исказ разбирлив за компјутерот.

4.2. Променливи и оператори во програмскиот јазик C/C++ за Arduino платформа

Променливите ги чуваат податоците што се обработуваат во компјутерот. За секоја променлива се предвидува и се резервира место во меморијата. Секое резервирано место има своја адреса. Бидејќи адресите се тешки за паметење, на променливите им се даваат симболични имиња. Освен симболичното име, на секоја променлива мора да ѝ се додели и ознака за вид на податок. Големината на резервираниот мемориски простор зависи од видот на податокот. Исто така, компјутерот различно се однесува спрема различни видови на податоци. На пример, со аритметичките инструкции можат да се обработуваат броеви, но не и текстуални податоци. Подолу се дадени ознаките на основните **видови на податоци**.

int ⟶ Целите броеви се најчесто користени податоци. Arduino платформата резервира два бајти мемориски простор за секој цел број. Опсегот на цели броеви изнесува од -32768 до +32 767.

float ⟶ Децималните броеви се користат за прикажување аналогни вредности, бидејќи тие имаат поголем број на можни вредности од целите броеви. Децималните броеви со 7 децимални места зафаќаат простор од 4 бајти во меморијата на Arduino и опсегот на броеви изнесува од 3,4028235E+38 до -3.4028235E+38. Бројот по буквата E претставува експонент на степенот со основа 10 (пример $35E3=35\cdot10^3=35\cdot1000=35000$)

double ⟶ Децималните броеви со 15 децимални места се попрецизни од децималните броеви со 7 децимални места и тие зафаќаат мемориски простор од 8 бајти.

char ⟶ Еден симбол (анг. chara) претставува една буква, цифра или алфанумерички знак. Еден симбол зафаќа меморија од еден бајт. Симболите се пишуваат помеѓу единечни наводници (пример ‘A’).

string ⟶ Текстот претставува низа од симболи и тој се пишува помеѓу двојни наводници (пример “dobar den”)

bool ⟶ Логичките податоци имаат само две вредности, нула или еден, неточно (false) или точно (true). Една логичка променлива зафаќа простор од еден бајт.

Освен **името и видот на податокот**, променливите имаат и свои **вредности**. Вредноста на променливата може да биде определена на почетокот од програмата, кога ги декларираме (најавуваме) променливите, или подоцна во текот на програмата. На променливите им се доделува вредност со знакот еднакво (=) и тој знак е познат и под името оператор за доделување. Вредноста на променливата може да се менува или да биде константна (анг. read only). Ако се работи за константа, тогаш пред видот на променливата треба да стои зборот const.

Пример 4.2.

```
1  int countUp = 0;                  // Целобројна променлива со симболично
                                     // име 'countUp'.
2  const float pi = 3.14;            // Децимален број со константна вредност.
3  bool isCodingFun = true;          // Логичка променлива со симболично име
                                     // isCodingFun и вистинита вредност.
4  string stringOne = "Hello String"; // stringOne е името на текстуална
                                     // променлива, а Hello String е неговата
                                     // вредност.
```

Низите се групи од променливи од ист тип. Декларацијата на низата содржи: вид на променлива, име на низата и број на елементи. Средната заграда го содржи бројот на елементи, а во големата заграда се набројани елементите. За пристап до елементите повторно се користи средната заграда односно се пишува името на низата и редниот број на елементот во средна заграда.

Пример 4.3.

```
1  string avto[3]={"Volvo", "BMW", "Ford"}; // Низата е група од текстуални
                                            // променливи, avto е името на низата,
                                            // бројот 3 во средната заграда значи
                                            // дека низата содржи 3 елементи, а
                                            // елементите се  во големата заграда.
                                            //
2  avto[0] = "Opel";                        // Го повикуваме елементот со реден
                                            // број нула и ја менуваме неговата
                                            // вредност.
```

Низите може да бидат ѝ дводимензионални и тогаш елементите се распоредени во редици и колони, а броевите во двете средни загради означуваат број на колони и број на редици.

Пример 4.4..

```
1  int Table [2][3] = {{3, 42,1}, {7, 3, 12}};   // Низата е составена од 2 колони и 3
                                                  // редици.
```

Операторите се знаци со кои програмерот, според точно определени правила, гради искази, инструкции. Постојат повеќе видови оператори: математички, логички, споредбени, оператори за доделување. Накратко ќе се задржиме на секој од нив.

Математичките оператори се познати под името аритметички оператори.

+	Оператор за одземање
-	Оператор за одземање
*	Оператор за множење
/	Оператор за делење
%	Оператор за пресметка на остаток при делење

Пример 4.5.

```
1  float a = 5.5;
2  float b = 6.6;
3  int c = 0;
4  c = a - b;        // Бидејќи 'c' е целобројна променлива, нејзината вредност ќе
                     биде  -1 иако резултатот од одземањето е  -1,1
```

Пример 4.6.

```
1  int a = 5;
2  int b = 10;
3  int c = 0;
4  c = a * b;        // Променливата 'c' добива вредност 50 .
```

Пример 4.7.

```
1  float a = 55.5;
2  float b = 6.6;
3  int c = 0;
4  c = a / b;        // Променливата 'c'  добива вредност 8, иако резултатот од
                     делењето е  8.409
```

Пример 4.8.

```
1  int x = 0;
2  x = 7 % 5;        //  x = 2
3  x = 9 % 5;        //  x = 4
```

Пример 4.9.

```
1  float a = 55.5;
2  float b = 6.6;
3  int c = 0;        // Променливата 'c' е целобројна.
4  c = a / b;        // Променливата 'c' добива вредност 8 иако резултатот од
                     // делењето е 8.409
```

Исказите што содржат **споредбени инструкции** даваат резултат true (точно) или false (неточно). Најчесто се користат во условни функции. Честопати, со цел да се намали должината на исказите, се користат посебни оператори.

!= ⟶ Ова е оператор за „не е еднакво“. Се споредуваат две променливи (x и y) или променлива и константа и ако се различни, исказот е вистинит. Променливите можат да бидат целобројни, децимални или бинарни броеви. Примената на овој оператор е прикажана во пример 4.10.

< ⟶ Оператор за помало. Променливата од левата страна на операторот треба да има помала вредност од променливата од десната страна.

> ⟶ Оператор за поголемо

<= ⟶ Оператор за помало или еднакво

>= ⟶ Оператор за поголемо или еднакво

== ⟶ Оператор за еднаквост

Пример 4.10.

```
1  if (x != y);  {        //  Дали x е различно од y?
2  .....                  //  Програмскиот код меѓу големите загради  се извршува
:  }                      //  само ако исказот во малите загради е вистинит.
```

Пример 4.11.

```
1  if (x <y); {           //  Дали x е помало од y?
2  .....                  //  Блок од инструкции чие извршување зависи од
:  }                      //  исполнетоста на горниот услов.
                          //  Крај на if структурата.
```

Логичките инструкции се нарекуваат уште и булови инструкции бидејќи тие можат да се применат на бинарни променливи (анг. bool). Резултатот е исто така бинарна вредност, нула или еден, неточно (анг. false) или точно (анг.true).

&& ⟶ Ова е оператор за логичка И операција (анг. and). За да се добие резултат логичка единица, двете променливи треба да бидат единици, односно точни (вистинити). Ако една од променливите е нула, тогаш и резултатот е нула.

|| ⟶ Оператор за логичка ИЛИ операција (анг. or). За да се добие логичка единица како резултат, барем една од двете променливи треба да биде единица, односно точна (вистинита).

! ⟶ Оператор за негација. Оваа операција има една променлива. Со неа се менува состојбата. Ако променливата била нула, резултатот ќе биде еден и обратно.

Пример 4.12.

```
1  if (!x)  {             //  Блокот од инструкции  во големата заграда се извршува
                          //  ако x е логичка нула, бидејќи по негацијата ќе добиеме
2  ……..                   //  резултат еден.
:  }
```

Подолу се дадени **скратените искази** и нивните еквивалентни аритметички и логички инструкции.

x+=y ⟶ x=x+y
x-=y ⟶ x=x-y
x*=y ⟶ x=x*y

x/=y ⟶ x=x/y
x++ ⟶ x=x+1
x- - ⟶ x=x-1
x&=y ⟶ x=x&&y
x|=y ⟶ x=x||y

4.3. Инструкции во програмскиот јазик C/C++ за Arduino платформа

Според функцијата, инструкциите ќе ги поделиме во неколку групи:

- Инструкции за работа со влезно-излезни пинови
- Инструкции за контрола на време
- Математички инструкции
- Бит и бајт инструкции
- Инструкции за сериска комуникација
- Инструкции за работа со библиотеки

4.3.1. Инструкции за работа со влезно-излезни пинови

Инструкциите за работа со влезно-излезните пинови служат за конфигурирање на пиновите (влез или излез) и за запишување или читање на нивните вредности. [12]

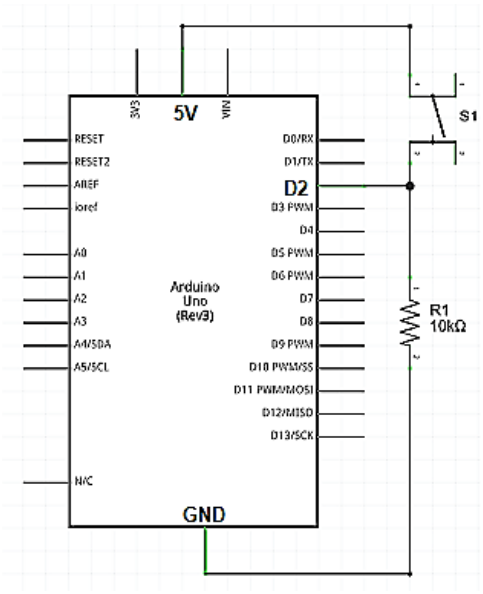
pinMode(pin,mode) ⟶ Под режим на дигитален пин подразбираме влезен или излезен режим на работа. Ако пинот е влезен тогаш Arduino прима податоци од електронската компонента, а ако е излезен тогаш Arduino испраќа податоци. За излез ознаката е OUTPUT, а за влез INPUT.

digitalRead(pin) ⟶ Со оваа инструкција **се чита вредноста** на влезниот пин. Вредноста може да биде HIGH (високо ниво) или LOW (ниско ниво). Во средната заграда се запишува бројот на пинот или неговото симболично име, доколку претходно сме го декларирале пинот како целобројна променлива. Ако пинот е излезен, тогаш инструкцијата читање е невозможна.

Пример 4.13.

```
1  int taster=2;                    //  taster е симболичното име на вториот пин.
2  int sostojba=0;                  //  Декларираме променлива со симболично
   /* ... */                        //  име sostojba и почетна вредност нула.
3  sostojba=digitalRead(taster);    //  Читање на вредноста на вториот пин и
```

Функционалната шема прикажана на слика 4.1. се однесува на програмскиот код од пример 4.13. Како влезна компонента е употребен тастер кој е приклучен на вториот пин на Arduino Uno R3. Кога тастерот е притиснат, вториот пин се поврзува со напојувањето и прочитаната вредност изнесува HIGH. Кога тастерот не е притиснат вториот пин е поврзан со заземјувањето и прочитаната вредност е LOW.

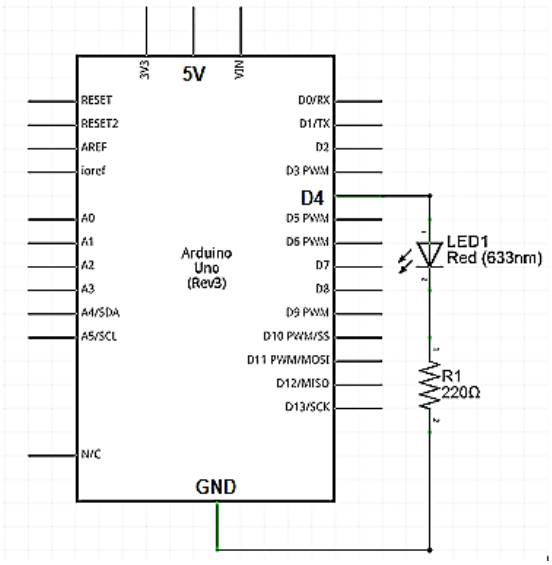


Слика 4.1. Поврзување на тастер со Arduino Uno R3

Ако отпорникот и тастерот во шемата на слика 4.1. си ги заменат местата тогаш кога тастерот ќе биде притиснат вториот пин ќе се поврзе со заземјувањето и прочитаната вредност ќе биде LOW односно логичка нула.

digitalWrite(pin,vrednost) → Со оваа инструкција, дигиталниот излезен пин се поставува на ниско или високо ниво. За разлика од инструкцијата digitalRead, инструкцијата digitalWrite содржи два параметри: симболично име на пинот (или неговиот број) и вредноста HIGH или LOW.

Функционалната шема прикажана на слика 4.2. се однесува на програмскиот код од пример 4.14. Како излезна компонента е употребена лед диода кој е приклучен на четвртиот дигитален пин на Arduino Uno R3. Кога четвртиот пин ќе се постави на високо ниво (HIGH) тогаш лед диодата ќе светне. Ако лед диодата ја поставиме обратно, со катодата свртена кон четвртиот пин тогаш таа ќе биде активна на ниско ниво (LOW).



Слика 4.2. Поврзување на лед-диода со Arduino Uno R3

Пример 4.14.

```
1  int ledDioda=4;                    //  ledDioda е симболично име на четвртиот
                                      //  пин.
2  digitalWrite(int ledDioda,HIGH);   //  Четвртиот пин се поставува на високо
                                      //  ниво и диодата свети.
```

Пример 4.15. претставува програма за примена на трите инструкции за работа со дигитални пинови. Само кога тастерот е притиснат лед-диодата свети.

Пример 4.15.

```
1   int ledPin = 13;                  //  Лед-диодата е поврзана на тринаесеттиот
                                      //  пин на Arduino Uno.
2   int inPin = 7;                    //  Тастерот е поврзан на седмиот пин.
3   int val = 0;                      //  Целобројната променлива val ја памети
                                      //  прочитаната вредност.
4   void setup() {                    //
5     pinMode(ledPin, OUTPUT);        //  Конфигурација на тринаесеттиот пин како
                                      //  излезен
6     pinMode(inPin, INPUT);          //  Конфигурација на седмиот пин како влезен
7   }                                 //
8   void loop() {                     //
9     val = digitalRead(inPin);       //  Читање на влезниот пин.
10    digitalWrite(ledPin, val);      //  Пинот на којшто е поврзана лед-диодата
    }                                 //  има иста вредност како и пинот поврзан со
                                      //  тастерот
```

Во наставната единица 2.3. Составни делови на Arduino Uno R3 се запознавме со аналогните влезови и излези на оваа платформа. Истакнавме дека за аналогните пинови од суштинско значење е ширинско импулсната модулација.

analogRead(pin) ⟶ Со оваа инструкција **се чита вредноста на еден аналоген влез**. Напонот на аналогниот влез може да има бесконечно многу различни вредности и се движи во опсегот од 0 до 5 V. Аналогно-дигиталниот конвертор, во составот на Arduino Uno R3, ги претвора аналогните вредности во цели броеви од 0 до 1023. Целиот број потоа се претставува како бинарен код од 10 битови. Ако вредноста 5 V се подели на 1024 дела, тогаш еден дел ќе одговара на напон од 4,9 mV. Значи сите вредности на напонот од 0 V до 4,9 mV ќе одговараат на бројот 0, од 4,9 mV до 9,8 mV на бројот 1, од 9,8 mV до 14,7 mV на бројот 2 и така

натаму, сè до напонот 5 V, кој ќе одговара на бројот 1023.

analogWrite(pin,value) ⟶ Со аналогните излези може да се контролира јачината на светлината на LED-диода или брзината на вртење на мотор**. На излезниот пин се запишува аналогна вредност и истата се претвора во низа од нули и единици.** Ширината односно времетрањето на импулсите и големината на излезниот напон зависат правопропорционално од запишаната целобројна вредност, која се движи во опсегот од 0 до 255. (слика 2.12)

Во примерот 4.16. на излезниот пин број 9 е поврзана лед-диода чиј интензитет на светлина зависи од положбата на лизгачот на потенциометарот кој е поврзан на третиот пин од Arduino Uno R3. Во програмскиот ред број 9 од истиот пример, прочитаната вредност од потенциометарот се дели со четири, бидејќи излезниот сигнал има четирипати помал опсег на вредности од оној на влезниот.

Пример 4.16.

```
1  int ledPin = 9;                  //  Лед-диодата е поврзана со
                                    //  деветтиот пин.
2  int analogPin = 3;               //  Потенциометарот е поврзан со
                                    //  третиот пин.
3  int val = 0;                     //  val е променлива за чување на
                                    //  прочитаната вредност.
4  void setup () {
5    pinMode(ledPin, OUTPUT);       //  Конфигурација на излезен пин.
6    }
7  void loop() {
8     val = analogRead(analogPin);  //  Читање на вредноста на влезниот
                                    //  сигнал. Прочитаната вредност е во
                                    //  опсегот од 0 до 1023.
9     analogWrite(ledPin, val / 4); //  Запишаната вредност е во опсегот
                                    //  од 0 до 255.
10 }
```

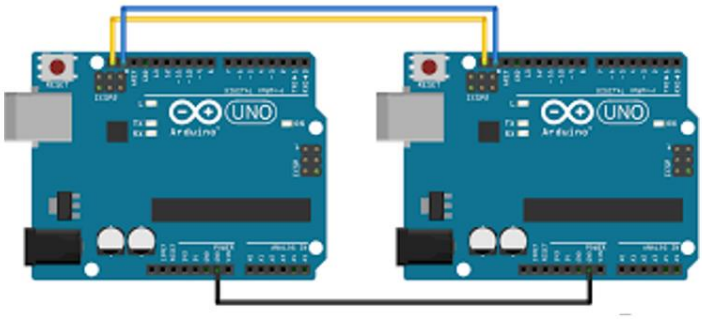
tone(pin, frequency)

tone(pin, frequency,duration) ⟶ Со инструкцијата се генерира периодична поворка од импулси за пинот 3 или 11 на Arduino Uno R3. Програмерот ја избира фреквенцијата. Постои можност да се избере ѝ времетраењето на побудниот сигнал. Во спротивно, тој ќе трае сè додека не се изврши инструкцијата notone(). На

излезниот пин може да се поврзе зујалица која ќе генерира едноличен тон.

4.3.2. Инструкции за сериска комуникација

Пиновите RX (Receive) и TX (Transmit) на Arduino Uno R3 се пинови за сериска комуникација. На сликата 4.3. е прикажан начинот на поврзување на две Arduino Uno R3 платформи, при што RX-пинот на првата платформа е поврзан со TX-пинот на втората платформа и обратно.



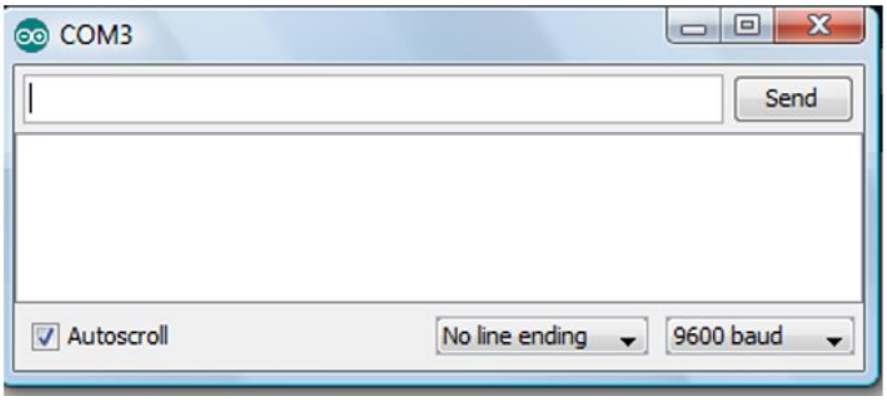
Слика 4.3. Сериска врска меѓу две Arduino Uno платформи

Преку овие пинови, платформата може да се поврзе со микрокомпјутер Raspberry Pi, Wi-Fi-модул, Bluetooth-модул и со други уреди што поддржуваат сериски пренос. Голема предност на сериската комуникација е што размената на податоци може да се следи преку вградениот сериски монитор. Сите податоци што Arduino Uno R3 ги прима или ги испраќа со други уреди можеме да ги видиме испечатени на екран. Се разбира, за ова е потребно Arduino платформата да биде поврзана со компјутер преку USB-кабел и да биде отворен серискиот монитор во рамките на развојната програма Arduino. Серискиот монитор е прикажан на слика 4.4.

Постојат дваесетина инструкции за работа со серискиот монитор, но ние ќе спомнеме само две.

Serial.begin(brzina) ⟶ Со оваа инструкција се поставува пропусниот опсег односно брзината на сериски пренос и таа е **наведена во долниот десен агол на серискиот монитор**. Стандардна брзина е 9600 boud што значи 9600 битови во секунди, но корисникот може да ја промени брзината на сериски пренос доколку тоа го наложува уредот поврзан на пиновите RX и TX.

Serial.print (x) ⟶ На екранот од серискиот монитор **се печати вредноста на променливата**.



Слика 4.4. Сериски монитор на развојна средина за Arduino платформа

Во пример 4.17. серискиот монитор го прикажува изминатото време, изразено во милисекунди.

Пример 4.17.

```
1  unsigned long vreme;
2  void setup() {
3    Serial.begin(9600);
4  }
5  void loop() {
6    Serial.print("Izminato vreme: ");
7    vreme = millis();
8    Serial.println(vreme);
9    delay(1000);                    // Пауза од една секунда со цел да се
10 }                                 // намали бројот на податоци прикажани
                                     // на серискиот монитор.
```

Во пример 4.18. Arduino Uno R3 ја мери вредноста на аналогниот напон на пинот A0, а серискиот монитор ја покажува истата. Да споменеме дека максималниот напон за влезно-излезните пинови на Arduino Uno R3 изнесува 5V и не е дозволено да се надмине оваа вредност. Во примерот е употребена инструкцијата analogRead. Бидејќи аналогно- дигиталниот конвертор ги претвора аналогните вредности во цели броеви од 0 до 1023, за да се добие вредноста на измерениот напон се применува формулата.

$$\mathrm{U[V]} = \frac{\text{Целобројна прочитана вредност}}{1023} \cdot 5\mathrm{[V]}$$

Пример 4.18.

```
1  const float maxVolti = 5.0;          // Максимален напон за пинот A0
4  void setup() {
5    Serial.begin(9600);                // Брзина на сериски пренос
7  }
8  void loop() {
9    int vrednost = analogRead(A0);     // Читање на влезниот пин.
```

```
10    float napon = (vrednost /1023) * maxVolti;    //  Пресметка на измерен напон
      Serial.println(volts);                        //  Прикажување на резултат
11  }
```

4.3.3. Инструкции за контрола на време

Програмските кодови се извршуваат со многу голема брзина. Ако на излезот поврземе лед-диода и таа треба да трепка, тогаш меѓу двете нејзини состојби (анг. ON/OFF) мора да се воведе време на доцнење (анг. delay). Во спротивно, корисникот нема да забележи дека диодата наизменично се вклучува и се исклучува.

delay (ms) ⟶ **Времето на доцнење** е дадено во милисекунди. Во една секуна има 1 000 милисекунди или 1000000 микросекунди.

Пример 4.19.

```
1   int ledPin = 13;                  //  Лед-диодата е поврзана со тринаесеттиот
                                      //  пин.
2   void setup() {
3     pinMode(ledPin, OUTPUT);        //  Пинот на лед-диодата се конфигурира
                                      //  како излезен.
4   }
5   void loop() {
6     digitalWrite(ledPin, HIGH);     //  Лед-диодата се вклучува.
7     delay(1000);                    //  Пауза од една секунда.
8     digitalWrite(ledPin, LOW);      //  Лед-диодата се исклучува.
9     delay(1000);                    //  Пауза од една секунда.
10  }
```

time=micros()
time=milis() ⟶ Со овие две инструкции се мери времето изразено во микросекунди и милисекунди од моментот кога Arduino Uno платформата ќе почне да ја извршува програмата. За прикажување на изминатото време, потребно е да се вклучи серискиот монитор.

4.3.4. Математички инструкции

max(x,y) ⟶ Инструкцијата го избира поголемиот број од двата броја x и y. Во пример 4.21, инструкцијата не дозволува вредноста на сензорот да биде помала од 20.

min(x,y) ⟶ Инструкцијата го избира помалиот број од двата броја x и y. Во пример 4.20. инструкцијата не дозволува вредноста на сензорот да биде поголема од 100.

constraim (x,a,b) ⟶ Инструкцијата го ограничува опсегот на променливата x; a е минималната вредност, a b е максималната вредност.

Пример 4.20.

```
1  sensVal = min(sensVal, 100);   // Ако вредноста на сензорот е помала од
                                  // сто, тогаш се задржува таа вредност. Ако
                                  // вредноста на сензорот е поголема од сто,
                                  // тогаш вредноста се изедначува со сто.
```

Пример 4.21.

```
1  sensVal = max(sensVal, 20);    // Ја избира поголемата вредност меѓу 20 и
                                  // вредноста на сензорот.
```

map(vrednost, fromLow,fromHihg,toLow, toHigh) ⟶ Инструкцијата **го менува опсегот на вредности** на променливата. fromLow и fromHihg се минималната и максималната вредност на стариот опсег, а toLow и toHigh се минималната и максималната вредност на новиот опсег.

Во пример 4.22. се менува опсегот на вредности на аналогниот сигнал. Во почетокот, кодирањето на влезните примероци се врши со 10 битови и опсегот е од 0 до 1023, но потоа го менуваме опсегот од 0 до 255, за кодирањето да биде со 8 битови.

Пример 4.22.

```
1  void loop() {                  //
2    int val = analogRead(0);     // Се чита аналогната вредност од нултиот
                                  // пин и се сочувува во променливата val..
3    val=map(val, 0, 1023, 0, 255); // Кога ќе се промени опсегот, се менува и
                                  // променливата. Бидејќи опсегот се
                                  // намалил, треба да очекуваме и вредноста
                                  // сразмерно да се намали.
4    analogWrite(9, val);         // Новодобиената вредност се запишува на
                                  // излезниот пин број 9.
5  }
```

4.3.5. Бит и бајт инструкции

bitClear (x,n) ⟶ Инструкцијата го **ресетира (поставува на нула) битот** со реден број n во променливата x. Ако целобројната променлива е во декаден броен систем, треба да се изврши претворање во бинарен броен систем. Битот со најмало значење (првиот од десно) е на нулта позиција. Резултат од оваа инструкција ќе биде променливата x со ресетиран n-ти бит.

bitSet (x,n) ⟶ Инструкцијата го сетира (поставува на високо ниво) битот со реден број n во променливата x.

bitRead (x,n) ⟶ Се чита вредноста на n-от бит од променливата x. Резултатот ќе биде еден бит 0 или 1, во зависност од прочитаната вредност.

bitWrite (x,n,b,) ⟶ Оваа инструкција има три параметри. X е целобројна променлива, n е реден број на битот и b е вредноста на битот што треба да се запише. Во пример 4.23., со инструкцијата bitWrite (x,0,1) се менува состојбата на битот со најмала тежина од нула на еден.

bit (n) ⟶ Со оваа инструкција се пресметува тежината на битот во зависност од неговиот реден број, односно неговата позиција во бинарниот број. Тежината на битот се пресметува според изразот 2^n. Така, почнувајќи од десно кон лево, првиот бит има тежина 2^0=1, вториот 2^1=2, третиот 2^2=4, четвртиот 2^3=8 итн.

Пример 4.23.

```
1  void setup() {
2    Serial.begin(9600);      //  нагодување на пропусен опсег на сериски монитор
3    byte x = 0b10000000;     //  префиксот 0b се користи за бинарни променливи
4    Serial.println(x,  BIN); //  Серискиот монитор го прикажува бинарниот број
     ,                        //  10000000.
5    bitWrite(x, 0, 1);       //  Запишување 1 на местото од најнезначајниот бит
6    Serial.println(x, BIN);  //  Серискиот монитор го прикажува бинарниот број
                              //  10000001.
7  }
```

highByte (x) ⟶ Ако податокот е 16-битен, со оваа инструкција се издвојуваат најзначајните осум битови.

lowByte (x) ⟶ Ако податокот е 16-битен, со оваа инструкција се издвојуваат најмалку значајните осум битови.

Пример 4.24.

```
1  int test = 0xABCD;   // Декларираме 16-битна променлива во
                        // хексадекаден броен систем.
2  byte hi, lo;         // Декларираме две бајт променливи.
3  hi = HighByte(test); // Добиваме резултат 0xAB
4  lo = LowByte(test);  // Добиваме резултат 0xCD
```

4.3.6. Структури во програмскиот јазик C/C++ за Arduino платформа

Структурите се блокови од инструкции со точно дефинирана функција. Блоковите од инструкции се напишани помеѓу големи загради { }. Ваквата поделба на програмата на структури и употребата на големите загради многу ја олеснува анализата на програмите.

4.3.6.1. Структури за избор на можности

Структурите за избор на можности се составени од исказите: if, else if и else.

```
if (услов) {
  инструкција 1
  инструкција 2
  инструкција 3
  .....
}
```

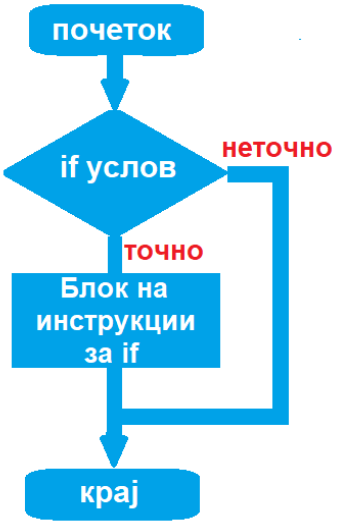
⟶ **Исказот If го проверува условот и ако тој е исполнет (вистинит) инструкциите се извршуваат**. Условот може да биде некој логички израз или израз со оператори за споредба. Синтаксата е таква што условот е запишан во мала заграда, а блокот инструкции во голема заграда. На слика 4.5. е прикажан блок дијаграмот на оваа структура.

```
if (услов1) {
 //блок
 //инструкции 1
}
else if (услов2) {
  //блок
 //инструкции 2
}
else {
  //блок
```

⟶ Структурата If…else овозможува подобра контрола на кодот бидејќи врши **проверка на повеќекратни услови**. Доколку условот под исказот if не биде исполнет, се проверува условот под исказот else if. Бројот на else if искази може да биде неограничен. Секој невистинит услов предизвикува нова проверка, сè додека не се добие вистинит услов. Кога ќе се добие вистинит услов, се извршуваат неговите инструкции и се прекинува проверката на следните услови. Доколку не се добие вистинит услов, се извршуваат

```
//инструкции 3
}
```

инструкциите под исказот else. Исказот else е последен исказ во структурата за избор на можности и може да се забележи дека под исказот else нема услов за проверка. На слика 4.6. е прикажан блок дијаграмот на if…else структурата и за разлика од if структурата постои блок инструкции кои се извршуваат ѝ кога if условот не е исполнет.



Слика 4.5. Блок-дијаграм на if структура

Слика 4.6. Блок-дијаграм на if…else структура

Пример 4.25. е пример за примена на структура за избор на три можности: температура поголема од 70 степени, температура меѓу 60 и 70 степени и температура помала од 60 степени.

Пример 4.25.

```
1  if (temperature >= 70) {
2    // Опасност! Исклучи го системот.
3  }
4  else if (temperature >= 60 && temperature < 70) {
5    // Внимание! Провери ги вредностите на другите параметри.
6  }
7  else {
8     // температурата е пониска од 60 степени.
9    //  Системот работи безбедно.
10 }
```

Во пример 4.26., кога тастерот не е притиснат лед-диодата свети, а кога истиот ќе се притисне диодата престанува да свети. Претпоставуваме дека кога тастерот е притиснат неговиот пин е на високо ниво и дека диодата е со активна анода.

Пример 4.26.

```
1  int val = digitalRead(taster);     //  Проверка на состојба на тастер.
2    if (val == LOW) {                //  Ја испитуваме вистинитоста на условот
3      digitalWrite(led, HIGH);       //  Пинот на диодата се поставува на високо
                                      //  ниво.
4    }
```

Во пример 4.27. како влезни компоненти се употребени два тастери. Во условот на if структурата е употребена логичката инструкција И и за да се изврши if структурата пиновите на двата тастери треба да бидат поставени на високо ниво.

Пример 4.27.

```
1  if (tasterA == HIGH && tasterB ==HIGH) {
   // Проверка на состојба на тастери и испитување на вистинитост на услов.
2     digitalWrite(led, HIGH);       //  Лед-диодата свети.
3    }                               //
4    else {                          //  Програмскиот код под else структурата се
                                     //  извршува ако не е исполнет условот под if
                                     //  структурата.
5     digitalWrite(led, LOW);        //  Лед-диодата не свети.
6  }
```

Структури за избор на можности може да се креираат и преку исказите **swich…case**.

```
switch (promenliva)
 {
  case 1:
   // блок инструкции
   break;
  case 2:
   // блок инструкции
   break;
  default:
   // блок инструкции
   break;
 }
```

→ Исто како исказот if, исказот switch…case го контролира текот на програмата, дозволувајќи да се креираат различни програмски кодови за различни случаи (анг. case). Веднаш до исказот switch, во мала заграда е наведено името на променливата од чија вредност зависи на кој случај ќе се префрли (анг. switch) извршувањето на програмата. Англискиот збор default во превод значи „стандардно“ и кодот под исказот default се извршува ако не се изврши ниеден од горните случаи. Исказот break се користи заедно со исказот switch и означува крај за секој случај. Без исказот break, структурата ќе продолжи да се извршува сè до нејзиниот крај.

Во пример 4.28. како влезна компонента е употребен сензор, фотоотпорник кој е поврзан со аналогниот влез A0. Да се потсетиме, аналогните сигнали од сензорите се влезни сигнали за аналогно-дигиталниот конвертор кој на својот излез генерира целобројни вредност во опсегот од 0 до 1023. Под претпоставка дека фотоотпорникот ќе го користиме во затворена просторија максималната вредност е намалена на 600. Во овој пример структурата switch

заменува три if структури. Во програмата се чита вредноста на сензорот, потоа со употреба на инструкцијата map() се менува бројот на можни вредности на вкупно три и на крај се прикажува една од трите пораки на серискиот монитор, во зависност од добиената вредност y.

Пример 4.28.

```
1  const int sensorMin=0;                    //  Се дефинираат максималната и
2  const int sensorMax=600;                  //  минималната вредност на сензорот.
                                             //
3  void setup() {
4     Serial.begin(9600);                    //  Се нагодува пропусниот опсег
5  }                                         //  на серискиот монитор.
6  void loop() {
7    int x=analogRead(A0);                   //  Се чита вредноста на сензорот
8    int y=map(x,sensorMax, sensorMin, 0,2);
   //Се врши намалување на опсегот на вредности на сензорот. Бројот на //можни
   вредности се намалува од 0-600 на 0-2.
9    switch(y) {                             //  Започнува switch структурата.
10   case 0:
11     Serialprintln(“temnica”);
12     break;
13   case 1:
14     Serialprintln(“polutemnica”);
15     break;
16   case 2:
17     Serialprintln(“svetlina”);
18     break;
19    }
20    delay(10);                             //  За постабилна работа се внесува
21 }                                         //  доцнење од 10 милисекунди меѓу
                                             //  секои две читање на сензорот.
```

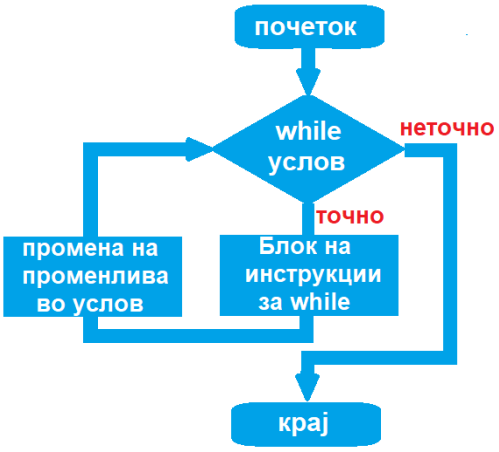
4.3.6.2. Структури за повторување (Циклични структури)

Структурите за повторување на циклус се составени од исказите: while, do…while и for. Заедно со овие искази може да се употреби и исказот continue.

```
while (услов) {
  инструкција 1
  инструкција 2
  инструкција 3
  ...
```

⟶ Англискиот збор while во превод значи „сè додека“ и се мисли на вистинитоста на условот во малата заграда. Сè додека условот е вистинит, циклусот ќе се повторува, односно блокот инструкции под исказот while повеќекратно ќе се извршува. Треба да се внимава

} циклусот while да не биде бесконечен. Ова ќе се случи ако променливата во условот не се менува со инструкциите во големата заграда на структурата. За таа цел, променливата во условот треба да се менува преку намалување или зголемување или да е зависна од состојбата на некој сензор.



Слика 4.7. Блок-дијаграм на while структура

Слика 4.8. Блок-дијаграм на do...while структура

Во пример 4.29. диодата трепка сè додека вредноста која ја измерил сензорот е поголема од 100.

Пример 4.29.

```
// Најнапред се врши аналогно-дигитално претворање, а потоа се испитува
// вистинитоста на условот во циклусот while.
1  while(analogRead(sensor) > 100) {
2    digitalWrite(LED, HIGH);        //  Лед-диодата свети
3    delay(100);                     //  Се чека 100 ms.
4    digitalWrite(LED, LOW);         //  Лед-диодата не свети.
5    delay(100);                     //  Се чека 100 ms.
6  }
```

do{
инструкција 1
инструкција 2
инструкција 3
....
} **while (услов)**

→ Разликата меѓу исказите do…while и while е во тоа што кај исказот do…while вистинитоста на условот се проверува на крајот од циклусот. Ова значи дека циклусот мора еднаш да се изврши, без оглед на вистинитоста на условот. Слика 4.7. и слика 4.8. претставуваат споредба меѓу двете структури while и do…while.

Во пример 4.30. повторно како влезна компонента се користи сензор. Циклусот do…while сè повторува сè додека на излез од аналогно-дигиталниот конвертор не се добие вредност поголема од 100.

Пример 4.30.

```
1  int x = 0;
2  do {
3  delay(50);                  //  На сензорот му се дава потребното време за да
                               //  изврши детекција.
4    x = analogRead(Sensor);   //  Читање на вредноста на сензорот.
6  } while (x < 100);          //  Испитување на вистинитост на услов.
```

Слика 4.9. Блок-дијаграм на for структура

for (декларација на променлива; услов; зголемување)
{
инструкција 1
инструкција 2
инструкција 3
.....
}

Исказот for се користи за **повеќекратно повторување на блок инструкции напишани во големата заграда.** Најчесто се користи бројач чија вредност постојано се зголемува и кога ќе го достигне максимумот, престанува извршувањето на циклусот. Честопати структурата за повторување со исказ for се користи за исто конфигурирање на повеќе пинови или за обработка на повеќе податоци на идентичен начин.

Во пример 4.31. лед диодата е поврзана на еден од аналогните излези кои даваат ширински модулирани импулси. Како се зголемува излезната вредност, од 0 до 255, така се зголемува времетраењето на импулсите и средната вредност на излезниот напон, а со тоа и интензитетот на светлината.

Пример 4.31.

```
1  int PWMpin = 10;                  //  Лед-диодата е поврзана на десеттиот пин.
2  void loop() {                     //
3     for (int i = 0; i <= 255; i++) //  Декларација на променлива за for
   {                                 //  структурата, услов за нејзино извршување и
                                     //  инструкција за зголемување на
                                         променливата.
4     analogWrite(PWMpin, i);        //  Вредноста на променливата се запишува на
                                     //  излезниот пин.
5     delay(10);
6      }
7  }
```

continue ⟶ Со исказот continue можат условно да се прескокнат неколку циклуси во структурите for, while или do…while.

Пример 4.32. е пример за примена на исказот continue. Во овој пример се прескокнуваат вредностите од 41 до 119, споредбено со пример 4.31.

Пример 4.32.

```
1  for (int x = 0; x <= 255; x ++) {
2    if (x > 40 && x < 120) {          //  Прескокнување на вредности.
3       continue;
4     }
5    analogWrite(PWMpin, x);
6    delay(50);
7  }
```

Пример 4.33. е комбинација од две структури, for и if. Циклусот for може да се повтори четири пати, но услов е вториот дигитален пин да биде на ниско ниво.

Пример 4.33.

```
1  for (int i=0; i < 4;)             //  Декларација на променлива и услов за
                                     //  повторување на циклусот
2  {                                 //  Почеток на for структура.
3     if(digitalRead(2) == LOW) {    //  Услов и почеток на if структура.
4       i++;
5     }                              //  Крај на if структура.
6  }                                 //  Крај на for структура.
```

4.3.6.3. Структури за гранење

Сè досега, програмите се извршуваа линеарно, односно инструкциите беа наредени последователно една врз друга. Со структурите за гранење се менува

текот на програмата, може да се прескокнуваат делови од програмскиот код и на таков начин да се оди напред-назад низ програмата.

goto oznaka; ⟶ Структурите за гранење се реализираат преку исказот goto, што во превод значи „оди на“. По исказот goto следува ознака (анг. label) којашто, всушност, претставува симболично име на местото каде што треба да се скокне.
.....
oznaka:

Во пример 4.34. се употребени повеќе структури: три for структури, една if структура и една goto структура. Ова е пример за вгнездување на една структура во друга. Ако е исполнет условот на if структурата тогаш ќе се изврши структурата за гранење, ќе се прескокне блокот на инструкции број 1 и директно ќе се изврши блокот на инструкции број 2. Ако условот на if структурата не е исполнет тогаш прво се извршува блокот инструкции број 1 и тоа повеќекратно. Колку пати ќе се изврши блокот на инструкции број 1 зависи од параметрите во for структурите за повторување и е еднакво на r×g×b пати.

Пример 4.34.

```
1  for (byte r = 0; r < 255; r++) {
2    for (byte g = 255; g > 0; g--) {
3      for (byte b = 0; b < 255; b++) {
4        if (analogRead(0) > 250) {
5          goto bailout;
.        }
.        // блок инструкции број 1
.      }
.    }
.  }
.  bailout:
.  // блок инструкции број 2
```

4.3.7. Инструкции за работа со библиотеки

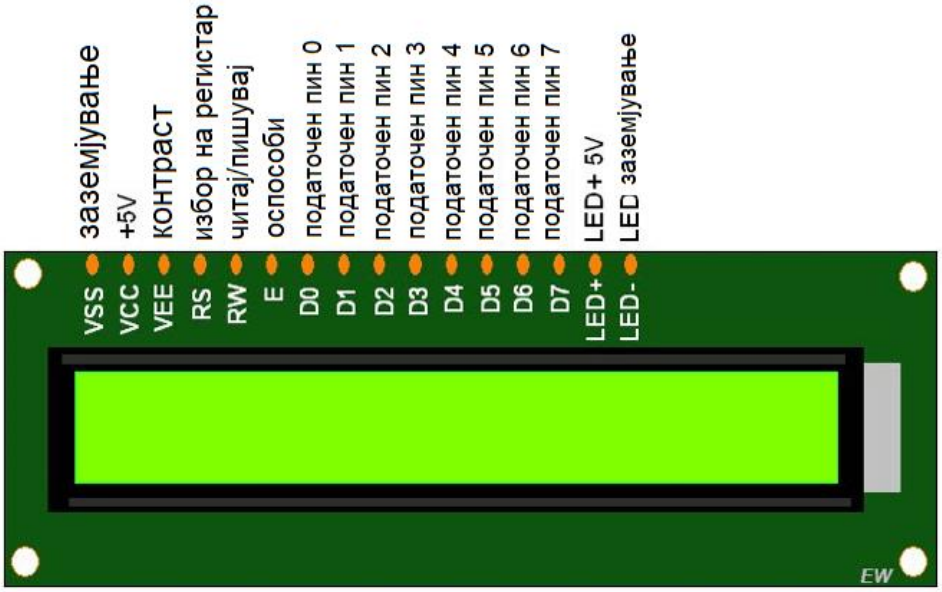
Како и повеќето програмски платформи, така и развојната средина на Arduino може да се прошири преку употреба на библиотеки. **Библиотеките се потпрограми што овозможуваат комуникација меѓу периферните уреди и Arduino платформата**. Тие содржат инструкции со кои се намалува пишувањето на долги програмски кодови и значително се олеснува работата. Според функционалноста, библиотеките можат да се поделат во неколку групи: комуникација (Wi-Fi , Ethernet, GSM), обработка на податоци, чување на податоци (Cloud Data, работа со SD-картички), дисплеи (LCD), управување со уреди (мотори), сензори (температурни, оптички, UV, притисок), тајминг итн. Arduino развојната средина содржи неколку стандардни библиотеки и нив

можеме да ги употребиме со притискање (анг. click) на **Sketch→ Import Library** во самото мени. Доколку развојна средина не ја содржи бараната библиотеката тогаш таа може да се преземеме од интернет.

#include < ime > ⟶ Со оваа инструкција се повикува библиотеката во главната програма, при што треба да се наведе името на библиотеката.

Ние ќе се запознаеме со следниве библиотеки: LiquidCrystal, Servo и CapacitiveSensor .

LCD-екраните (анг. Liquid crystal display) се излезни компоненти и служат за визуелно прикажување на резултатите добиени од извршувањето на програмата. На слика 4.10. е прикажан изгледот на алфанумерички LCD-екран со две редици со по 16 симболи. На истата слика е прикажан и неговиот пин-дијаграм.



Слика 4.10. Надворешен изглед на LCD-екран и пин-дијаграм

Библиотеката LiquidCrystal е стандардна библиотека во состав на развојната средина и преку неа Arduino платформата управува со LCD-екранот. Оваа библиотека содржи 20 различни инструкции, но ние ќе објасниме само три, кои се најчесто користени.

LiquidCrystal (rs, rw, enable, d4, d5, d6, d7) ⟶ **Со оваа инструкција се дефинираат пиновите на** Arduino платформата кои се користат за поврзување со истоимените изводи на LCD-екранот. Оваа инструкција следува по вклучувањето на библиотеката во програмата како што е наведено во пример 4.35.

lcd.begin(kol,red) ⟶ Со оваа инструкција започнува комуникацијата со екранот, а броевите во средната заграда го опишуваат бројот на редици и колони.

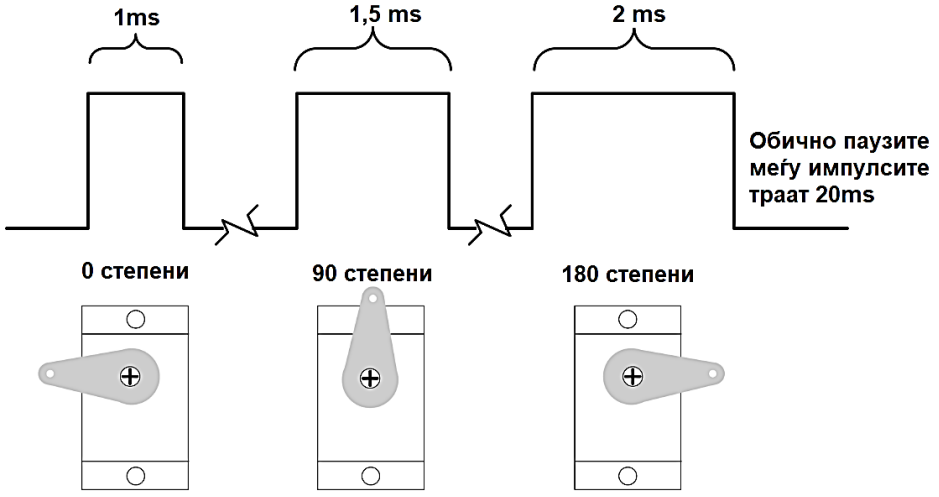
lcd.print (podatok) ⟶ Оваа инструкција има само еден параметар, а тоа е текстот што треба да се прикаже на LCD-екранот.

Пример 4.35.

```
1  #include <LiquidCrystal.h>                  // Повикување на библиотека за
                                               // работа со LCD-екран.
2  LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);  // Избор на пинови за поврзување
                                               // на Arduino Uno R3 со LCD-
                                               // екранот.
3  void setup() {
4    lcd.begin(16,1);                          // Екранот има 16 колони и една
                                               // редица.
5    lcd.print("hello, world!");
6  }
7  void loop() {}
```

Во практичниот дел на оваа модуларна единица предвидена е практична вежба со употреба на LCD-екран и тогаш подетално ќе се запознаеме со начинот на негово поврзување со Arduino Uno R3 и програмата за прикажување на текст на екранот.

Серво-моторот е посебен вид на мотор што не врти во круг, туку врши аголни поместувања и останува во одредена позиција до следната инструкција. Вообичаено, серво-моторот ротира за 180 степени. Серво-моторите многу често се користат во роботиката, на пример, за придвижување на роботска рака.



Слика 4.11. Зависност на аголот на вртење на серво моторот од ширината на импулсите

За да серво-моторот се заврти за определен агол, потребни му се широчинско модулирани импулси, чија ширина варира меѓу една и две милисекунди. Поради чувствителноста на серво моторите наместо инструкцијата anologWrite и нејзините ширински модулирани импулси се користи Серво библиотеката и нејзините инструкции.

 # include (Servo.h) ⟶ Со оваа инструкција се вклучува библиотеката за контрола на серво-моторот во програмата.

Servo ime ⟶ Креираме серво-објект на кој ќе се извршуваат функциите содржани во Серво библиотеката. Исто како променливите, и на објектите им се задаваат симболични, уникатни имиња.

servo.attach (pin, min, max) ⟶ pin е бројот на пинот на Arduino платформата на кој е поврзан серво-моторот**. Само пиновите 9 и 10 имаат поддршка за поврзување на серво-мотор.** Min и max не се задолжителни параметри и претставуваат ширина на импулси, изразена во микросекунди, за минимален и максимален агол на завртување на серво-моторот. Колку е поголема ширината на импулсите, толку е поголем и аголот.

servo.write(agol) ⟶ Аголот на завртување на серво-моторот може да има вредност од 0 до 180 степени. Кога аголот е 0 тогаш поместувањето е најголемо во една насока, а за агол од 180˚ поместувањето е најголемо во друга насока. За 90 степени нема поместување.

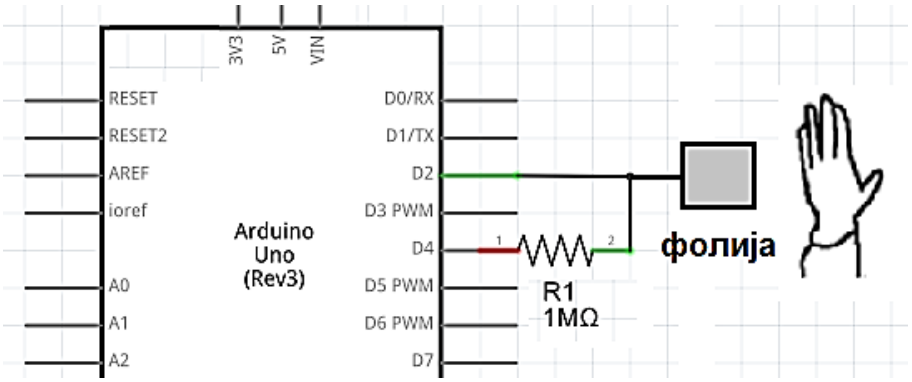
Во примерот 4.36. употребени се горните инструкции, при што за поврзување на серво-моторот е избран деветтиот пин и серво-моторот е во средината, на положба 90 степени.

Пример 4.36.

```
1  #include <Servo.h>
2  Servo myservo;          // Креираме серво-објект со симболично име
                           // myservo
3  void setup(){
4    myservo.attach(9);
5    myservo.write(90);    // Серво-моторот е позициониран точно на
                           // средина.
6  }
7  void loop() {}
```

Во практичниот дел предвидена е практична вежба со употреба на серво мотор.

Со **CapacitiveSensor** библиотеката два или повеќе пинови на Arduino Uno R3 платформата може да се претворат во капацитивен сензор за детектирање на капацитивноста на човечкото тело. За таа цел потребно е меѓу влезниот и излезниот пин да се поврзе отпорник со голема отпорност и парче алуминиумска фолија на крајот од влезниот пин. [5]



Слика 4.12. Примена на CapacitiveSensor библиотеката

Ако до парчето фолија се доближи човек, неговото тело ќе апсорбира дел од електрицитетот, поради што ќе се промени состојбата на еден од пиновите. Arduino Uno платформата го мери времето потребно за да се изедначи состојбата на двата пинови. Ако телото апсорбира поголемо количество електричество, тогаш и времето ќе биде подолго. Ќе се запознаеме со три инструкции од библиотеката CapacitiveSensor.

 # include (CapacitiveSensor.h) → Со оваа инструкција се вклучува библиотеката за контрола на капацитивноста во програмата.

CapacitiveSensor ime (pin sender, pin reciver) → Креираме објект со симболично име. Името на објектот се наведува пред секоја употребена инструкција од CapacitiveSensor библиотеката. Исто така, со оваа инструкција ги конфигурираме пиновите што ќе бидат поврзани со алуминиумска фолија. Помеѓу нив се поставува отпорник со отпорност од 1МΩ до 40МΩ. Чувствителноста на сензорот се зголемува со зголемување на отпорноста. Со отпорник од 1МΩ сензорот работи како сензор за допир, а со отпорник од 10МΩ како сензор за блискост, со максимално растојание до 15cm.

CapacitiveSensor(broj na primeroci) ⟶	Ова е инструкција за читање на вредноста на сензорот. Пред инструкцијата се наведува името на објектот, разделен со точка од инструкцијата. Во малата заграда се пишува бројот на примероци кој вообичаено е 30.

За пример 4.37. употребен е капацитивен сензор, кој е поврзан на вториот и четвртиот пин на Arduino Uno R3. Ако ја допреме фолијата тогаш вредностите на екранот од серискиот монитор треба да бидат поголеми.

Пример 4.37.

```
1  #include <CapacitiveSensor.h>        //  Повикување на библиотека за работа
                                        //  со капацитивен сензор.
2  CapacitiveSensor senDopir = CapacitiveSensor(4,2);
3  //Креирање на објект со симболично има senDopir. Избор на пинови за
   //поврзување на Arduino Uno платформата со капацитивниот сензор.
4  void setup() {                       //  Избираме пропусен опсег за сериски
5     Serial.begin(9600);               //  монитор.
6  }
7  void loop() {
8  long sensorValue = senDopir.capacitiveSensor(30);
9  // Читање на вредност на сензор.
10 Serial.println(sensorValue);         //  Прикажување на вредност на сензор
11 }                                    //  на серискиот монитор.
```

4.4. Програмирање на микрокомпјутер Raspberry Pi со програмски јазик Python

Програмскиот јазик Python е еден од најпопуларните програмски јазици, особено меѓу младата популација. Причина за ова е неговата **лесна разбирливост**, што се должи на синтаксата која е многу слична со синтаксата на англискиот јазик. Познавањето на англискиот јазик е предуслов за брзо совладување на програмскиот јазик Python. Пример 4.38. претставува **краток програмски код** за Raspberry Pi, на чиј излезен пин е поврзана диода. Програмскиот кодот е многу пофлуентен. Во исказите не се користат кратенки за опишување на исказите, туку цели зборови. Исказите не завршуваат со знакот точка запирка како што тоа беше случај со програмскиот јазик C/C++.

Пример 4.38.

```
1  from gpiozero import LED
2  from time import sleep
3  led = LED(17)
4  while True:
5      led.on()
6      sleep(1)
7      led.off()
8      sleep(1)
```

Програмскиот јазик Python е од повисок ред во однос на програмскиот јазик C/C++. Преведување на програмата напишана во програмскиот јазик Python го врши системска програма **интерпретер**, а не компајлер. Компајлерот го претвора програмскиот код во машински јазик одеднаш, додека интерпретерот ги преведува исказите еден по еден, додека се извршува програмата. Програмските кодови преведени со компајлер се извршуваат побрзо. Програмскиот јазик Python е покомпатибилен со оперативниот систем Linux и инсталацијата во оперативниот систем Windows е малку посложена. Да се потсетиме, јадро (анг. cernal) на оперативниот систем Raspbian е Linux.

Програмските кодови во програмскиот јазик Python се пократки зашто **не постои декларирање на променливи.** Во програмскиот јазик C/C++ , за променливата x запишуваме int x=5;, а во програмскиот јазик Python само x=5. Исто така, нема ограничување во видот на податоци, односно ако делиме два цели броја, можеме веднаш да добиеме резултат децимален број, а не само цел број.

Кога пишуваме блок на инструкции во рамките на една структура, како што се if….else или while, не се користат големи загради, туку инструкциите се вовлекуваат за четири празни места навнатре. Откако ќе ја декларира структурата, програмерот треба да притисне две точки (:) и уредувачот на текст

самиот го вовлекува текстот. Коментарите од програмскиот код не се разделуваат со две коси црти, туку со знакот тараба „#“.

Слика 4.13. Карактеристики на програмскиот јазик Python

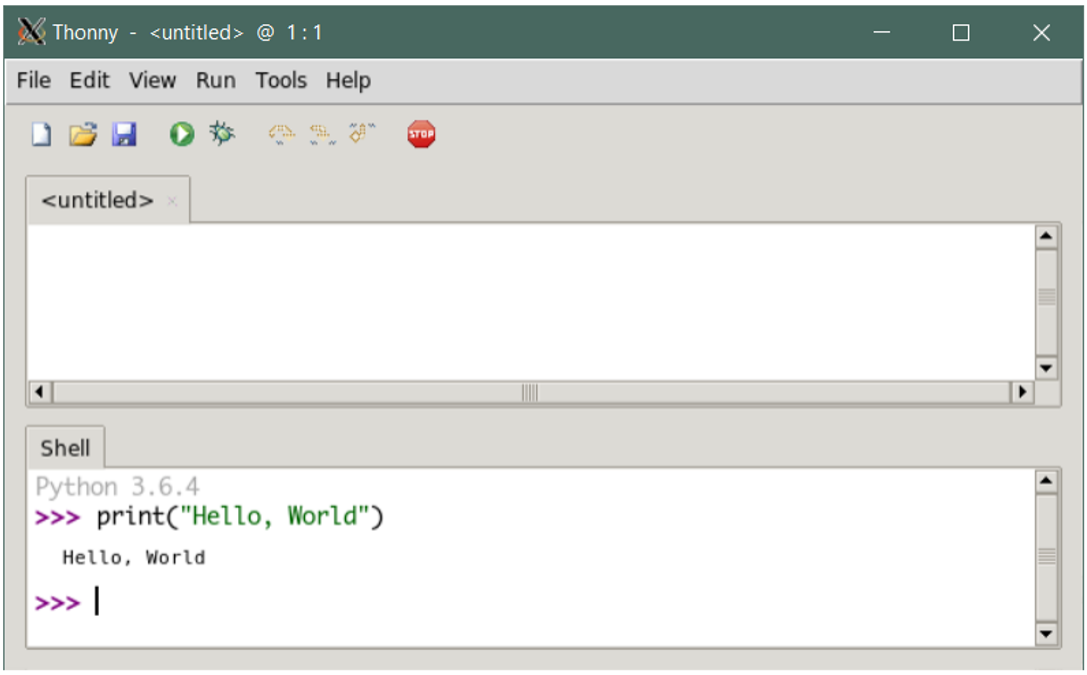
Во програмскиот јазик Python, **програмерот може да креира своја функција** преку инструкцијата def и потоа неа да ја повикува на различни места во програмата. Таков е програмскиот код прикажан во примерот 4.39..

Пример 4.39.

```
1  def my_function(fname, lname):
2      print(fname + " " + lname)
3  my_function("Bill", "Gates")
```

Програмскиот јазик C/C++ е подобар за програмирање вградливи системи, а програмскиот јазик Python е поуниверзален и со него можеме да креираме и графички апликации. C/C++ е хардверски, а Python е софтверски ориентиран. Програмскиот јазик Python располага со голем број библиотеки со најразлична намена. Секоја библиотека располага со огромно инструкциско множество. Библиотеката turtle се користи за графички дизајн (на пример, да нацртаме ѕвезда што ќе ротира). Библиотеката pygame се користи за креирање видеоигри со визуелни и звучни ефекти. OpenCV е библиотека за обработка на слики. Cloud4rpi е библиотека за складирање податоци во облак технологија (анг. cloud). Доволно е да истакнеме дека во моментот постојат 137 000 библиотеки за програмскиот јазик Python. Не е доволно само да се одбере соодветната библиотека, туку потребно е да се проучат и наредбите во нејзини рамки. Ние ќе ги проучиме можностите што ги нуди библиотеката GPIO Zero, која се користи кога Raspberry Pi се поврзува со влезно-излезни уреди (диоди, тастери, сензори, мотори, дисплеи).

Во третата тема се запознавме со инсталацијата на оперативниот систем Raspbian и со неговата работна површина. Во менито, или поточно во категоријата Programming, се наоѓа интегрираната развојната средина Thonny Python IDE која служи за програмирање на Raspberry Pi во програмскиот јазик Pyton. На слика 4.14. е прикажан прозорецот после отворањето на развојната програма.



Слика 4.14. Развојна програма Thonny за програмирање на Raspberry pi во програмски јазик Python

Исто како Arduino развојната средина, така и развојната програма Thonny Python IDE содржи уредувач на текст и конзола за дебагирање. Во уредувачот на текст ја пишуваме програмата, а во конзолата за дебагирање го гледаме резултатот добиен со извршувањето на програмата. Со притискање на копчето New започнуваме нова програма и по нејзиното креирање ја сочувуваме со притискање на копчето Save as.

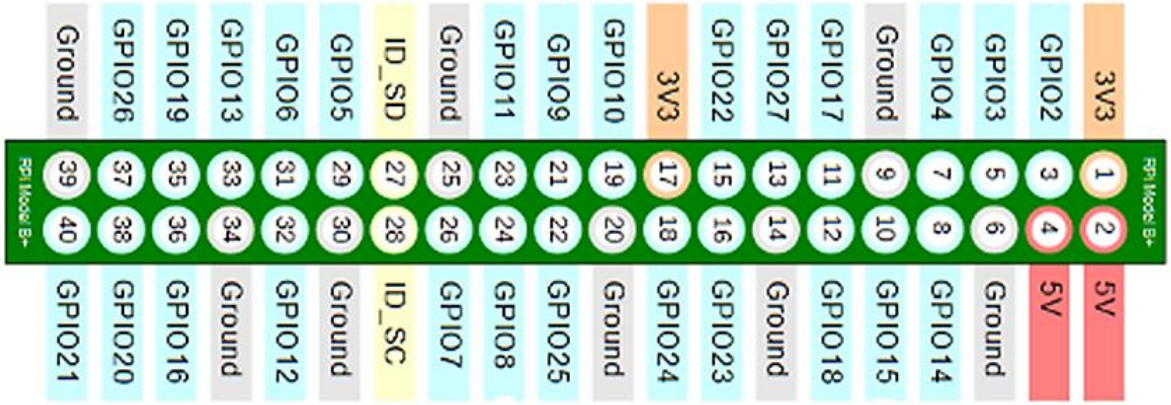
4.5. Библиотека GPIO Zero

Во втората тема, кога се запознавме со хардверот, истакнавме дека за поврзување на Raspberry Pi 3B+ со влезно-излезни уреди се користи **40-пински приклучок,** познат под кратенката GPIO (анг. General Purpose Input Output). На овој приклучок**, 26 пинови се за поврзување со влезно-излезни уреди, а останатите се за заземјување и за напојување од 3,3 V и 5 V.**

За да може Raspberry Pi 3B+ да комуницира со влезно-излезните уреди поврзани на GPIO пиновите, потребно е да се вклучи библиотека наречена GPIO

Zero. Тоа е стандардна библиотека за оперативниот систем Raspbian и нема потреба од дополнителна инсталација. [13]

Броевите на пиновите според оваа библиотека се различни од физичките броеви на пиновите. На сликата е прикажано нумерирањето на таканаречените GPIO-пинови. На пример, на пинот со физички број 15 одговара GPIO-пин со број 22.



Слика 4.15. Означување и функционален опис на GPIO пиновите

Исто како програмирањето на Arduino Uno R3 во C/C++, така при програмирањето на Raspberry Pi 3B+ во Python, на почетокот од програмата треба да се вклучат библиотеките што ќе се користат во неа. За оваа цел се користи инструкцијата include. Во пример 4.40., при конфигурирање на пинот за поврзување на тастер, додаден е префиксот gpiozero.

Пример 4.40.

```
1 import gpiozero
2 button = gpiozero.Button(2)
```

Во програмскиот јазик Python, дозволено е наместо целата библиотека, да се вклучи една класа од неа. Во примерот 4.41. се вклучува само делот од библиотеката што ги содржи наредбите за управување со тастер. Во овој случај можеме директно да користиме тастери во програмата, без префиксот gpiozero.

Пример 4.41.

```
1 from gpiozero import Button
2 button = Button(2)
```

Постои можност наместо со поединечни тастери да работиме со цели тастатури. Во тој случај треба да се повика класа од библиотеката gpiozero позната под името ButtonBoard, што во превод значи плоча со тастери.

Пример 4.42.

```
1 from gpiozero import ButtonBoard
```

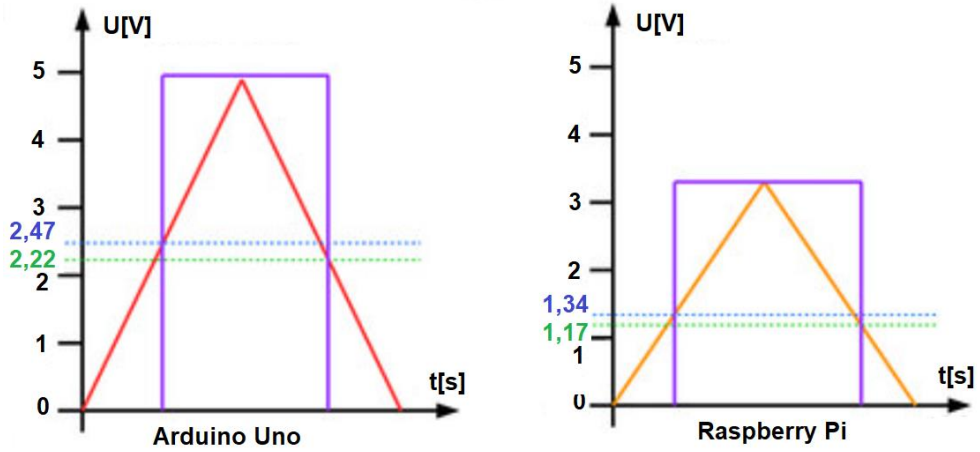
Веќе спомнавме дека **библиотеката GPIO е поделена на класи, во зависност од видот на влезно-излезните уред**и. Подолу се претставени класите и уредите на коишто тие се однесуваат.

Влезни уреди	→	Тастер, инфрацрвен сензор, сензор за движење, оптички сензор, сензор за растојание
Излезни уреди	→	Лед-диода (RGB или контролирана со импулсно-ширински импулси), зујалици, мотори, серво-мотори
Сериски периферни уреди	→	Аналогно-дигитален претворувач
Платформи и додатоци	→	LED-дисплеи, тастатури, роботи, контролери за домашна употреба преку далечинско управување
Внатрешни компоненти	→	Нагодување на време, температура на процесор, пресметка на средни вредности, искористеност на дискот и друго
Генератори на тонови	→	Генерирање музички тонови, прости или сложени

4.6. Класи на влезни уреди од библиотеката GPIO Zero

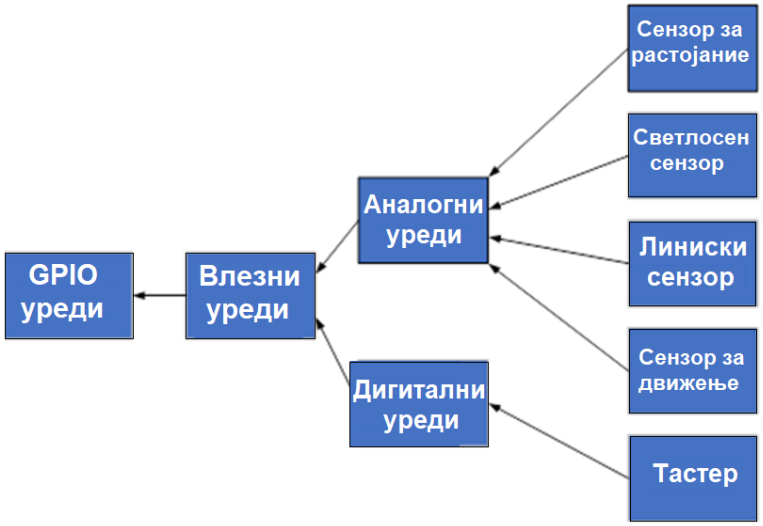
Дигиталните сигнали имаат само две вредности, еден или нула, LOW или HIGH. Аналогните сигнали можат да примаат најразлични вредности во опсегот од 0 до максимално дозволениот напон, вклучувајќи и децимални броеви. Raspberry Pi нема вграден аналогно-дигитален претворувач како Arduino платформата. Доколку сакаме да отчитуваме аналогни вредности од влезовите на Raspberry Pi тогаш треба да поврземе екстерен аналогно-дигитален претворувач како што е на пример интегрираното коло MCP3008. За работа со влезни аналогни уреди може да се искористи таканаречениот напон на праг (анг. threshold). Ако влезниот напон е поголем од напонот на праг тогаш таа вредност ќе се смета за логичка единица (HIGH), а ако влезниот напон е помал од напонот на праг тогаш таа вредност ќе се смета за логичка нула (LOW). На слика 4.16. споредбено се прикажани напоните на праг за Arduino Uno и Raspberry Pi. Напонот на влезните пинови на Raspberry Pi не смее да биде поголем од 3,3V. На истата слика може да видиме дека напонот на праг се разликува во зависност

од тоа дали влезниот напон се намалува или зголемува. Кога влезниот напон се зголемува од нула кон максимално дозволената вредност напонот на праг изнесува 1.34V за Raspberry Pi и споредбено 2,47V за Arduino Uno. Кога влезниот напон се намалува од максимално дозволената вредност кон нула тогаш напонот на праг изнесува 1.17V за Raspberry Pi и споредбено 2,22V за Arduino Uno.



Слика 4.16. Споредба на напони на праг за Arduino Uno и Raspberry Pi

На сликата 4.17 е прикажана поделбата на влезните уреди според видот на сигналите што ги генерираат.

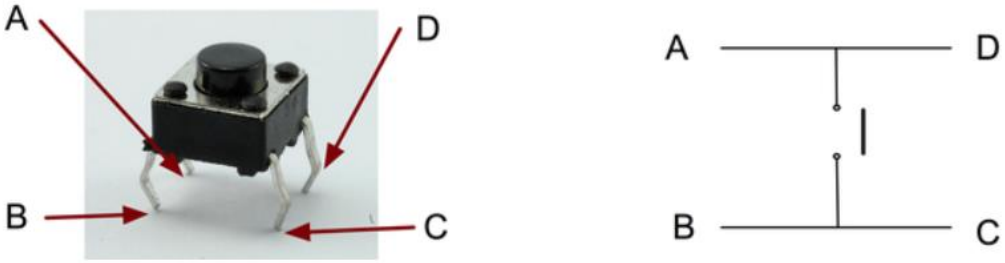


Слика 4.17. Влезни уреди за Raspberry Pi

Сите класи во составот на библиотеката GPIOZERO покажуваат вакво разгранување. Ваквиот пристап во голема мера го поедноставува програмирањето на Raspberry Pi.

4.6.1. Тастер (анг. Button)

Да се потсетиме дека приклучоците на тастерот кои лежат на иста линија, но спротивна страна се кусо поврзани. Ова е прикажано на слика 4.18.



Слика 4.18. Внатрешна поврзаност на приклучоците на тастер

Примерот 4.43. претставува програмски код за прикажување на една линија текст под услов тастерот да биде притиснат.

Пример 4.43.

```
1  from gpiozero import Button
2  button = Button(4)
3  button.wait_for_press()
4  print("The button was pressed!")
```

Во овој програмски код, употребени се две инструкции за работа со тастер: Button() и wait_for_press () . Постојат уште 12 други инструкции за работа со тастер. Ќе ги објасниме поважните инструкции и нивните параметри, кои се пишуваат во малата заграда.

Button() ⟶ Во примерот 4.43, бројот 4 претставува **број на GPIO-пин.** Овој параметар е задолжителен. Во спротивно, конзолата за дебагирање ќе пријави грешка. Ќе го споменеме и параметарот pull_up. Овој параметар може да има две состојби, true или false. Стандардната вредност true значи високо ниво (HIGH) кога тастерот е притиснат. Во овој случај, едниот пин од тастерот се поврзува на заземјување, а другиот на GPIO-пинот. Ако параметарот pull_up е false, тогаш се добива ниско ниво (LOW) кога тастерот е притиснат и првиот пин наместо на маса се поврзува со напојувањето од 3,3 V.

wait_for_press () ⟶ Извршувањето на **програмскиот код се прекинува** додека не помине означеното време или тастерот не биде притиснат.

wait_for_release() ⟶ Извршувањето на програмскиот код се прекинува додека не помине означеното време или тастерот не биде ослободен.

held_time ⟶ Ако тастерот го држиме притиснат подолго време, оваа инструкција го прикажува времето на притискање изразено во секунди.

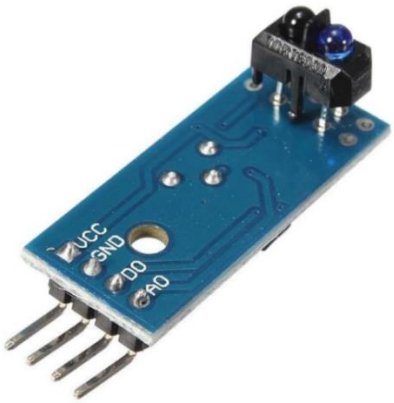
hold_time ⟶ Оваа инструкција определува **колку секунди да се чека** по притискањето на тастерот.

is_held ⟶ По истекот на времето определено со инструкцијата hold_time вредноста станува вистинита (true).

is_pressed ⟶ Резултатот од оваа инструкција е булова вредност, true или false. Вредноста е вистинита (true) само кога тастерот е притиснат.

4.6.2. Инфрацрвен рефлектирачки модул за следење (TRCT5000)

Инфрацрвениот рефлектирачки модул за следење е прикажан на слика 4.19. Тој може да се искористи за детекција на полна линија**.** Модулот има четири приклучоци, но за поврзување со Raspberry Pi користиме три: Vcc , GND и DO (анг.Data Out). Vcc-пинот се поврзува со напојувањето од 3,3 V, GND е заземјување. DO-пинот е пин за сигнализација и се поврзува со еден од GPIO-пиновите.



Слика 4.19. Инфрацрвен рефлектирачки модул за следење

Примерот 4.44. претставува програмски код за прикажување на една линија текст, во зависност од вредноста што ќе ја прочита сензорот.

Пример 4.44.

```
1  from gpiozero import LineSensor
2  from signal import pause
3  sensor = LineSensor(4)
4  sensor.when_line = lambda: print('Line detected')
5  sensor.when_no_line = lambda: print('No line detected')
6  pause()
```

LineSensor(4) ⟶ Со оваа инструкција **го конфигурираме пинот** за поврзување на сензорот

when_line ⟶ Инструкцијата се извршува кога сензорот е активен.

when_no_line ⟶ Инструкцијата се извршува кога сензорот не е активен

lambda ⟶ Ова е стандардна Python инструкција којашто се користи како подинструкција на друга инструкција, за да означи некакво дејство. Не се користи како самостојна инструкција.

4.6.3. Сензор за растојание (HC-SR04)

Сензорот за растојание е ултразвучен сензор што испраќа насочен бран, кој се рефлектира од објектот поставен пред сензосетрот и се враќа до сензорот. Се мери времето од моментот кога предавателот ќе го испрати бранот до моментот кога приемникот ќе го прими рефлектираниот бран. Според овој параметар се пресметува растојанието помеѓу сензорот и објектот. **Испратениот бран се нарекува тригер (активатор), а рефлектираниот бран се вика ехо.**

Слика 4.20. Сензор за растојание (HC SR04)

Примерот 4.45. претставува програмски код за HC SR04 кој дава информација за растојанието изразено во cm. Пинот за тригерирање е поврзан со пинот GPIO17.

Пример 4.45.

```
1  from gpiozero import DistanceSensor
2  from time import sleep
3  sensor = DistanceSensor(echo=18, trigger=17)
4  while True:
5      print('Distance: ', sensor.distance / 100)
6  sleep(1)
```

DistanceSensor() ⟶ **Задолжителни се два параметри. Прв е бројот на GPIO-пинот за ехото, а потоа за тригерот**. Тие се пишуваат во малата заграда.

wait_for_in_range ⟶ Извршувањето на програмскиот код се прекинува додека растојанието е над растојанието на праг (анг. threshold_distance) или не помине времето наведено во малата заграда. Растојанието на праг вообичаено

изнесува 0.3m и како параметар може да се нагоди со инструкцијата DistanceSensor().

wait_for_out_of_range → Извршувањето на програмскиот код се прекинува додека растојанието е под растојанието на праг или не помине наведеното време.

distance → Резултат од оваа инструкција е растојанието што го измерил сензорот и ова растојание е дадено во метри.

when_in_range → По оваа инструкција, следува знакот еднакво и друга помошна инструкција која се извршува доколку условот е исполнет .

when_out_of_range → Инструкцијата дефинира дејство доколку растојанието е над растојанието на праг.

4.6.4. Оптички сензор или фотоотпорник (анг. LDR-Light Dependet Resistor)

Фотоотпорникот е најчесто користен сензор за детекција на светлина. Неговата отпорност се менува од 1KΩ при светло и 100KΩ при мрак односно истата се намалува кога интензитетот на светлината се зголемува. Подоцна ќе реализираме практична вежба со негова примена, а сега ќе се запознаеме со неколку инструкции за работа со овој сензор.

Пример 4.46.

```
1  from gpiozero import LightSensor
2  ldr = LightSensor(18)
3  ldr.wait_for_light()
4  print("Light detected!")
```

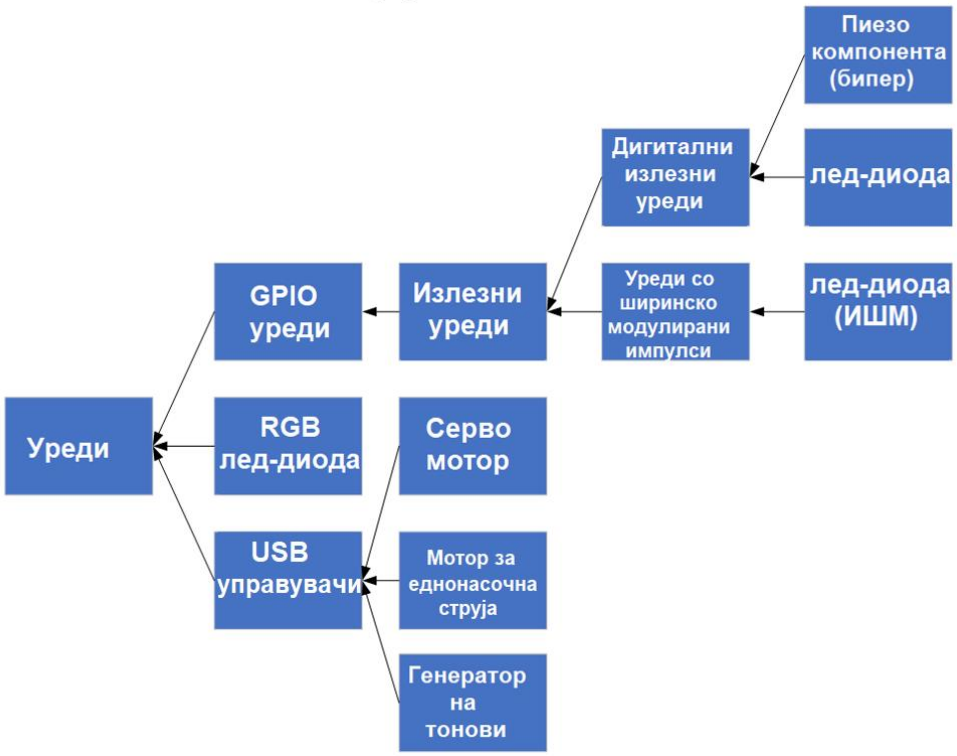
LightSensor( ) → Во малата заграда се наведува бројот на GPIO-пинот. Во примерот 4.46. , ldr е симболично име на сензор и името го задава програмерот.

wait_for_light() → Со оваа инструкција, програмскиот код се стопира сè додека не се активира сензорот. Во малата заграда може да се наведе и времето што треба да помине за да сензорот се активира.

wait_for_dark () → Со оваа инструкција се стопира програмскиот код сè додека не се појави мрак.

4.7. Класи на излезни уреди од библиотеката GPIO Zero

На сликата 4.21. е прикажана поделбата на излезните уреди според видот на сигналите што ги генерираат.



Слика 4.21. Излезни уреди за Raspberry Pi

Стеблото со излезни уреди е разгрането уште повеќе. Според видот на приклучокот, извршена е уште една поделба, излезни уреди со GPIO-приклучок и уреди со USB-приклучок. Уредите со GPIO-приклучок се поделени на дигитални излезни уреди и уреди со импулсно-ширинска контрола (PWM).

За вклучување и исклучување на дигиталните уреди, потребни се две вредности, LOW или HIGH. Со импулсно-ширинската контрола вршиме промена на интензитетот на светлина на лед-диодата. Исто како и кај влезните уреди, така и кај излезните уреди, библиотеката gpiozero содржи специјално множество од инструкции за секој уред посебно.

4.7.1. Лед-диода

При поврзувањето на LED-диодата со Raspberry Pi, треба да внимаваме **подолгиот пин (анодата) да се поврзе со GPIO-пинот, а пократкиот пин (катодата) со заземјувањето.** Исто така треба да се употреби отпорник заради ограничување на струјата.

Програмскиот код во примерот 4.47. врши вклучување на LED-диодата.

Пример 4.47.

```
1  from gpiozero import LED
2  led = LED(17)
3  led.on()
```

LED() ⟶ Во малата заграда се наведува бројот на GPIO-пинот, активната и почетната вредност. Ако активната вредност е HIGH, тогаш диодата ќе свети како во горниот пример. Ако active_high=false, тогаш за да светне лед-диодата ни треба инструкција off(). led е симболично име на диодата што го задава програмерот.

blink (on_time=1**,** off_time=1**)** ⟶ Со оваа инструкција, диодата почнува да трепка. Можеме да го менуваме **времето на вклучување и исклучување. Стандардно, ова време изнесува една секунда.**

off() ⟶ Оваа инструкција ја исклучува лед-диодата

on() ⟶ Оваа инструкција ја вклучува лед-диодата

toggle() ⟶ Оваа инструкција **ја менува состојбата на лед-диодата**, ако била вклучена ја исклучува и обратно.

4.7.2. Лед-диода контролирана со ширински модулирани импулси (PWMLED)

Ако сакаме да го менуваме интензитетот на светлината на лед-диодата тогаш е потребно да ја повикаме библиотеката GPOIZero класа PWMLED. Светлината е поголема кога импулсите од периодичната поворка имаат поголема ширина. Ширината на импулсите може да ја менуваме со помош на потенциометар.

Слика 4.22. Зависност на интензитетот на светлина од ширината на импулсите

Доколку сакаме го менуваме интензитетот на светлина во програмата треба да се вклучи класата PWMLED од gpiozero библиотеката. Во инструкцијата PWMLED(), во малата заграда се наведува бројот на GPIO-пинот за поврзување,

фреквенцијата на периодичната поворка и почетната вредност. Доколку не се наведе фреквенција тогаш ќе важи стандардната фреквенција од 100 Hz. За управување со интензитетот на светлина се користат вредности од нула до еден.

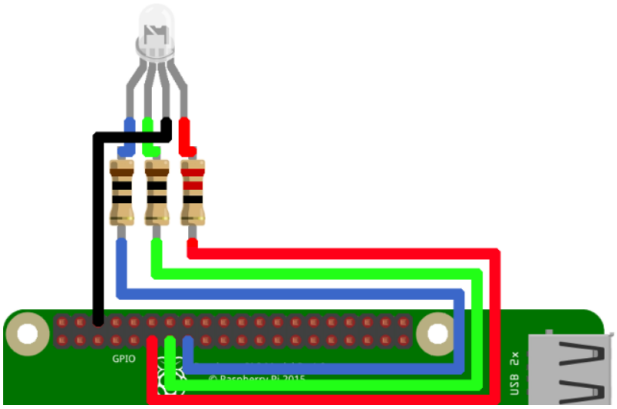
Пример 4.48.

```
1  from gpiozero import PWMLED
2  from time import sleep
3  led = PWMLED(17)
4  while True:
5      led.value = 0
6      sleep(1)
7      led.value = 0.5
8      sleep(1)
9      led.value = 1
10     sleep(1)
```

4.7.3. Лед-диода со променлива боја (RGB-лед)

Кратенката RGB доаѓа од зборовите red, green, blue, што во превод значи црвена, зелена и сина. Со мешање на овие три основни бои се добиваат сите останати. Постојат 256 нијанси од секоја основна боја и со мешање на овие нијанси можат да се добијат 16 777 216 различни бои. Секоја основна боја претставува променлива чија вредност е децимален број со вредност од 0 до 1. **Останатите бои се добиваат како комбинација од основните.** На пример, комбинацијата (1, 0, 0) ја претставува црвената боја, комбинацијата (0, 1, 0) ја претставува зелената боја, комбинацијата (1, 1, 0) ќе биде жолта боја и комбинацијата (1, 0.5, 0) ќе биде портокалова боја. За полесна употреба на боите може да се користи библиотеката color, која е стандардна библиотека на програмскиот јазик Python.

RGB лед-диодата има четири приклучоци, три аноди и една катода која е подолга. Катодата се поврзува со заземјувањето, а анодите со три GPIO пинови, по еден за секоја од трите основни бои. Со програмскиот код во пример 4.49. лед-диодата ќе свети жолто. Пример 4.50. е програмски код за употреба на библиотеката color.



Слика 4.23. Поврзување на RGB-диода со Raspberry Pi

Пример 4.49.

```
1  from gpiozero import RGBLED
2  led = RGBLED(2, 3, 4)
3  led.color = (1, 1, 0)
```

Пример 4.50.

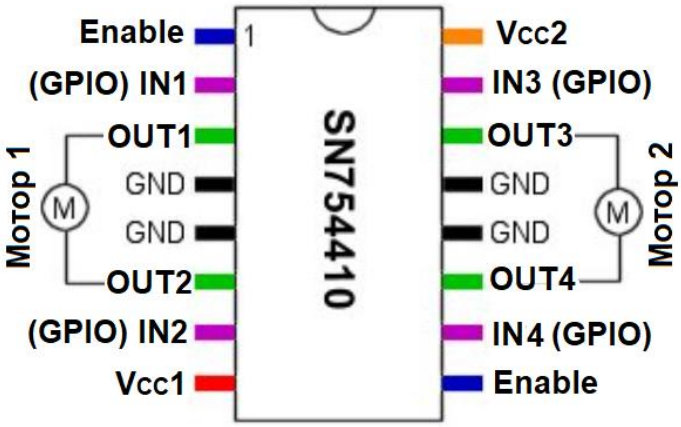
```
1  from gpiozero import RGBLED
2  from colorzero import Color
3  led = RGBLED(2, 3, 4)
4  led.color = Color('yellow')
```

За RGB-диодата важат истите инструкции како и за обичната лед-диода, со тоа што треба да се конфигурираат три GPIO-пинови и да се дефинира бојата на лед-диодата со инструкцијата color(). Светлината на RGB-диодата може да се менува според интензитет ако во инструкција RGBLED() се вклучат ширински модулираните импулси со употреба на параметарот pwm=True.

4.7.4. Мотор на еднонасочна струја

Ако се промени насоката на струјата, ќе се промени и насоката на вртење на моторот на еднонасочна струја. Тој **се поврзува со Raspberry Pi преку H-мост.** Принципот на работа на H-мостот е прикажан на слика 4.24 .

Слика 4.24. Принцип на работа на H-мост



Слика 4.25. Пин-дијаграм на H-мост

Тој е насочувач и може да го менува поларитетот на напонот. Ако прекинувачите S1 и S4 се затворени, а прекинувачите S2 и S3 се отворени тогаш моторот врти во насока на стрелката на часовникот. Ако S1 и S4 се отворени, а прекинувачите S2 и S3 се затворени тогаш моторот врти во обратна насока. За работа со Raspberry Pi како H-мост најчесто се користат интегрираните кола

L293D или SN754410. На слика 4.25. е прикажан пин-дијаграмот на интегрираното коло SN754410 и истото може да се користи за управување со два мотори за еднонасочна струја. Ако со моторите се поврзат тркала тогаш овој склоп може да се искористи за движење на робот. Излезите на H-мостот се поврзуваат со моторот на еднонасочна струја, а влезовите со два GPIO-пинови.

Во практичната вежба 4.8.2.7. Промена на насока вртење на мотор на еднонасочна струја е употребено интегрираното коло SN754410 и во истата е објаснет начинот на поврзување на моторот, H-мостот, Raspberry Pi и протоплочката.

Подолу се дадени неколку инструкции за работа со DC-мотор.

Motor() ⟶ Во малата заграда се запишуваат броевите на GPIO-пиновите за поврзување на H-мостот.

backward*(speed=1)* ⟶ Моторот за еднонасочна струја се врти во насока спротивна од насоката на стрелката на часовникот**. Вредноста на брзината на вртење (speed) може да се менува во опсегот од 0 до 1,** под услов да се користат ширински модулирани импулси. Еден е максимална вредност на брзината на вртење.

forward*(speed=1)* ⟶ DC-моторот се врти во насока на стрелката на часовникот.

reverse() ⟶ Се менува насоката на вртење на моторот.

stop() ⟶ Моторот престанува да се врти.

Програмскиот код во примерот 4.51. го врти моторот нанапред, во насока на стрелките на часовникот.

Пример 4.51.:

```
1  from gpiozero import Motor
2  motor = Motor(17, 18)
3  motor.forward()
```

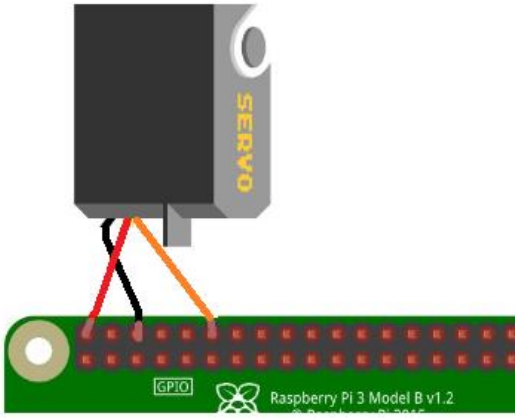
4.7.5. Серво-мотор

Да се потсетиме, серво-моторот не се врти во круг, туку тој се поместува за одреден агол, кон лево или десно од својата средна позиција, во зависност од големината на управувачкиот сигнал. Начинот на работа на серво-моторот е прикажан на слика 4.11.

На слика 4.26. е прикажан надворешниот изглед на серво моторот, а на слика 4.27. е прикажан начинот на неговото поврзување со GPIO пиновите на

Raspberry Pi. Серво-моторот има три приклучоци: црвен, црн и портокалов. Црвениот кабел се поврзува со напојувањето од 5 волти, а црниот со заземјување. Портокаловиот кабел се поврзува со еден од GPIO-пиновите и преку него Raspberry Pi комуницира со моторот, односно го контролира аголот на вртење.

Слика 4.26. Надворешен изглед на серво-мотор



Слика 4.27. Поврзување на серво=мотор со Raspberry Pi

Со програмскиот код во примерот 4.52, серво-моторот ќе се заврти за 45 степени.

Пример 4.52.

```
1  from gpiozero import Servo
2  servo = Servo(17)
3  servo.value = 0.5
```

Ќе се задржиме само на инструкцијата servo.value(), со која се дефинира вредноста на аголот на вртење. Ова вредност може да се менува во опсегот меѓу -1 (минимален агол од 0 степени) и 1 (максимален агол од 180 степени). Ако вредноста изнесува нула, тогаш серво-моторот е во својата средна позиција односно 90 степени.

4.8. Практични вежби за модуларна единица: Програмирање на микрокомпјутери

4.8.1. Практични вежби за програмирање на Arduino Uno R3 во програмскиот јазик C/C++

Сите вежби се едноставни и наменети за почетници, со цел практична примена на теоретските знаења од програмирање на Arduino развојна платформа. За успешно програмирање на Arduino Uno R3 потребно е познавање на нејзиниот хардвер и софтвер. Пред почетокот на практичните вежби треба да се повторат следните наставни единици:

- 2.4. Електронски компоненти за Arduino Uno R3 и нивно поврзување
- 2.8.3.1. Мерки за заштита и безбедност при работа со Arduino-базирана микроконтролерска платформа
- 2.8.3.2. Упатство за работа со протоплочка
- 2.8.3.3. Компјутерска симулација за Arduino Uno R3 платформа
- 3.2.6.2. Мени и алатки на развојна средина и впишување на програма во Arduino Uno R3 платформа

4.8.1.1. Практична вежба: Вклучување на лед-диода преку тастер

1. **Цел на вежбата**

Цел на вежбата е поврзување на лед-диода и тастер, како влезна и излезна компонента, со Arduino Uno R3, според дадена функционалната и монтажна шема и употреба на протоплочка. Учениците треба да напишат програма во развојната средина, да ја внесат програмата во меморијата на Arduino Uno R3 и да ја испитаат нејзината работа.

2. **Потребни компоненти** за реализација на оваа вежба се:
    - Arduino Uno R3 платформа
    - протоплочка
    - тастер за печатена плочка

- лед диода
- отпорници со отпорности R1=220Ω и R2=1KΩ
- жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

Тастерот испраќа вредности (HIGH или LOW) до Arduino Uno R3, а потоа платформата управува со лед-диодата. Отпорникот R1 е потребен за да се нагоди работната струја на диодата, со цел јачината на светлината да биде оптимална, без штетно загревање. Отпорникот R2 врши заштита на пинот на кој е поврзан тастерот од брзи промени на неговиот напон.

3. Подготовка за вежбата

Учениците треба да се потсетат на:

- пин-дијаграмот на Arduino Uno R3
- начинот на поврзување на лед-диода и тастер на протоплочка. Истиот е објаснет во упатството за работа со протоплочка.
- влезно-излезните инструкции за Arduino Uno R3, задолжителните структури (setup и loop) и if…else структурата за избор на можности.

4. Опис на електричната шема и начин на поврзување

На слика 4.28. се прикажани функционалната и монтажната шема за реализација на оваа вежба.

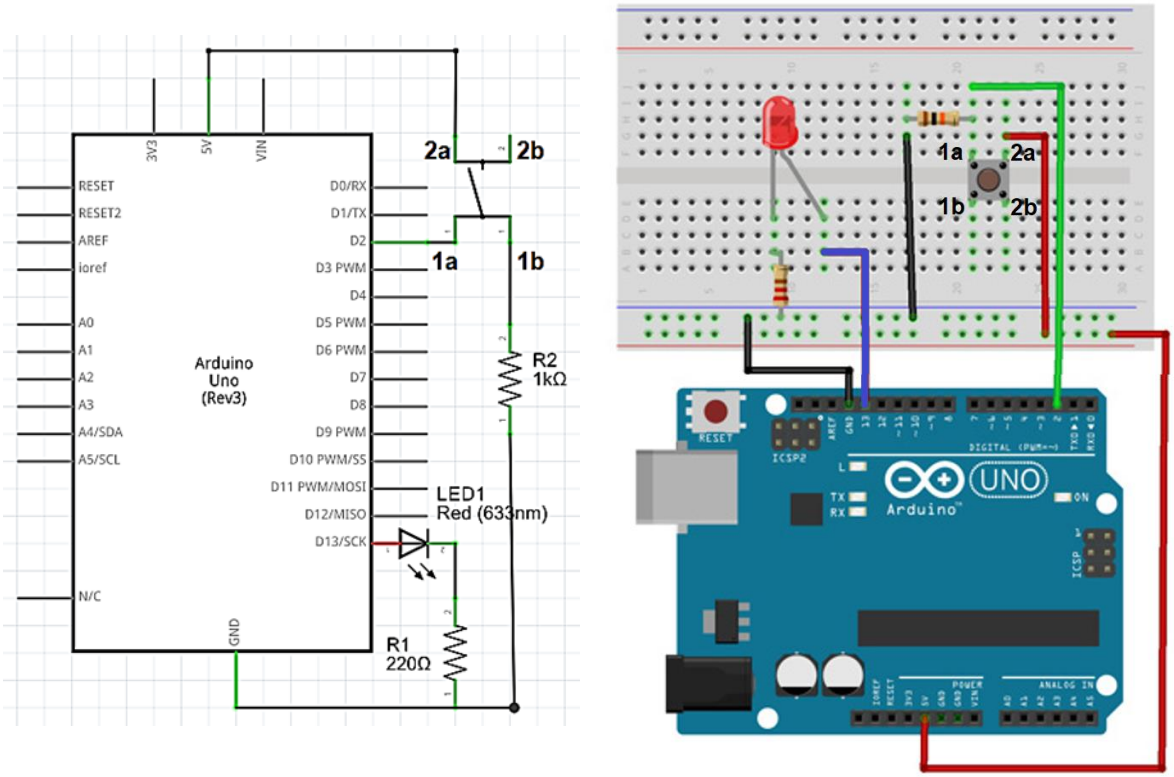
Кога тастерот не е притиснат вториот пин на Arduino Uno R3 е поврзан со заземјувањето што одговара на ниско логичко ниво (анг. LOW) и напонот на вториот пин изнесува 0V. Кога тастерот е притиснат вториот пин на Arduino Uno R3 е поврзан со напојувањето што одговара на високо логичко ниво (анг. HIGH) и напонот на вториот пин е различен од нула. Лед-диодата свети кога 13-от пин е на високо логичко ниво односно кога излезниот напон на пинот изнесува 5V.

Поврзувањето на компонентите се реализира на следниот начин:

(1) Го поставуваме тастерот на средината од протоплочката. За поврзување на тастерот се користат три краткоспојници. Со црвена краткоспојница ги поврзуваме приклучоците 2a-2b со лентата за напојување (обележана со + или со црвена боја). Со црната краткоспојница и преку отпорникот R2 ги поврзуваме приклучоците 1a-1b на тастерот со лента за заземјување (обележана со минус или сина боја). Истовремено, со зелената краткоспојница приклучоците 1a-1b ги поврзуваме со вториот дигитален пин на Arduino Uno R3. Зелената краткоспојница е еден вид сигнална жица и преку неа тастерот ги испраќа вредностите HIGH или LOW до вториот пин.

(2) Ја поставуваме лед-диодата на протоплочката при што треба да се внимаваме анодата да биде поврзана на контактна точка со повисок потенцијал од онаа на катодата. Со црна краткоспојница и преку отпорникот R1 ја поврзуваме катодата на лед-диодата со лентата за

заземјување, а со сина краткоспојница ја поврзуваме анодата со 13-от пин на Arduino Uno R3.



Слика 4.28. Поврзување на лед-диода и тастер со Arduino Uno R3

(3) Со црвена краткоспојница го поврзуваме 5V пинот на Arduino Uno R3 со лента за напојување на протоплочката, а со црната жица го поврзуваме GND пинот со лентата за заземјување.

5. Пишување и внесување на програма за Arduino Uno R3

Arduino Uno R3 ја проверува состојбата на тастерот со употреба на инструкцијата digitalRead(). Лед-диодата се вклучува и исклучува софтверски со употреба на инструкцијата digitalWrite(). Програмата содржи if...else структура за избор на една од две можности (лед-диодата свети или лед-диодата не свети). Состојбата на тастерот е услов за извршување на if...else структура

Подолу е дадена програмата со којашто се врши контрола на горното коло.

```
// Се врши декларирање на константите и променливите пред да се напишат
//структурите setup и loop
// Константите не се менуваат и се користат за избор на пинови за поврзување
1  const int buttonPin = 2;     // Тастерот е поврзан на дигиталниот
                                // пин број 2.
2  const int ledPin =  13;      // Лед-диодата е поврзана со
                                // дигиталниот пин број 13.
// Променливите чуваат податоци што постојано се менуваат.
```

```
3  int buttonState = 0;                    // buttonState е име за вредноста што се
                                           // добива со читање на пинот број 2, каде
                                           // е приклучен тастерот.
4  void setup() {                          // Почеток на структурата setup.
5    pinMode(ledPin, OUTPUT);              // Пинот на кој е поврзана лед-диодата
                                           // се конфигурира како излезен пин, што
                                           // значи Arduino ќе испраќа податоци.
6    pinMode(buttonPin, INPUT);            // Пинот на кој е поврзан тастерот се
                                           // конфигурира како влезен пин, што
                                           // значи Arduino ќе прима податоци.
7  }                                       // Крај на структурата setup.
8  void loop() {                           //  Почеток на структурата loop.
9     buttonState=digitalRead(buttonPin);  // Се чита вредноста на пинот број 2 и
                                           // таа се запишува во променливата
                                           // buttonState.
10   if (buttonState == HIGH) {            // Се проверува дали тастерот е
                                           // притиснат
11     digitalWrite(ledPin, HIGH);         // Ако е притиснат, тогаш лед-диодата
                                           // свети.
12   }                                     // Крај на исказот if.
13     else {                              // Почеток на исказот else.
14     digitalWrite(ledPin, LOW);          // Ако не е притиснат тастерот, диодата
                                           // не свети.
15   }                                     // Крај на исказот else.
16 }                                       // Крај на структурата loop.
```

Постапката за внесување на програма во меморијата на Arduino Uno R3 е иста за сите програми и истата е објаснета, чекор по чекор, во наставната единица 3.2.6.2. Мени и алатки на развојна средина и впишување на програма во Arduino Uno R3 платформа.

6. Резултати од реализацијата на вежбата

Веднаш после внесувањето на програмата треба да се провери дали Arduino Uno R3 правилно работи. Кога ќе го притиснеме тастерот лед-диодата треба да свети. Ако го отпуштиме тастерот тогаш лед-диодата престанува да свети.

Коментар:_____

7. Направи промена!

- Што ќе се случи ако двете digitalWrite инструкции во програмата си ги заменат местата?

  Коментар:_____
- Што ќе се случи ако внесеме време на доцнење од 5s после секоја од двете digitalWrite инструкции? Да се употреби инструкцијата delay().

  Коментар:_____

4.8.1.2. Практична вежба: Контрола на лед-диода со потенциометар

1. Цел на вежбата

Оваа вежба претставува практична примена на аналогните влезови на Arduino Uno R3. Учениците ќе научат како големината на аналогните влезни сигнали влијаат на времетраењето на сигналите на излез од дигиталните пинови на Arduino Uno R3. Со помош на потенциометар ќе ја менуваме брзината на трепкање на лед-диода.

2. Потребни компоненти за реализација на оваа вежба се:

- Arduino Uno R3 платформа
- протоплочка
- тример-потенциометар со отпорност од 10 ĶΩ
- лед диода
- отпорник со отпорност R1=220Ω
- жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

Во оваа практична вежба потенциометарот е главна компонента која е поврзана со еден од аналогните влезови на Arduino Uno R3. Потенциометарот е со променлива отпорност и со него се генерира променлив напон со максимална вредност од 5 V и минимална вредност од 0 V. Аналогно-дигиталниот претворувач вграден во самата платформа ги претвора аналогните вредности во цели броеви од 0 до 1023. Колку е поголема вредноста на излез од аналогно-дигиталниот претворувач, толку поретко диодата ќе се вклучува и исклучува, односно лед-диодата ќе трепка со помала брзина.

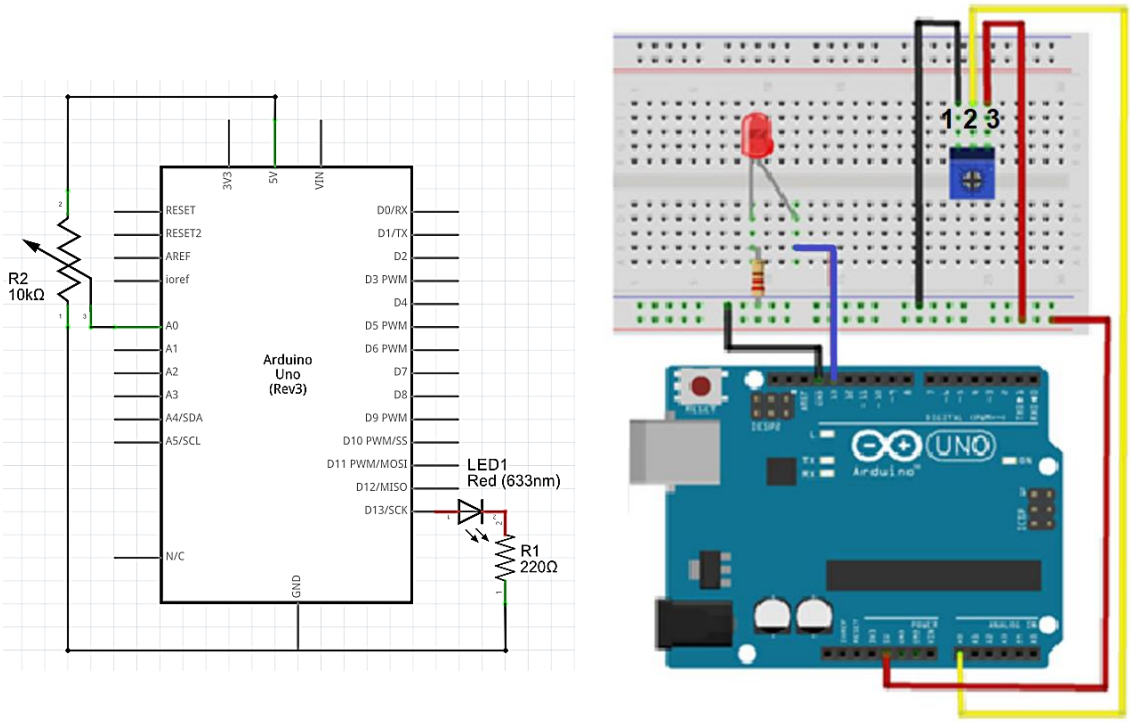
3. Подготовка за вежбата

Учениците треба да се потсетат на:

- импулсно ширинската модулација и нејзиното значење за аналогните влезови на Arduino Uno R3
- инструкциите за работа со аналогните влезни пинови на Arduino Uno R3
- значењето на инструкцијата delay() и изборот на време на доцнење

4. Опис на електричната шема и начин на поврзување

На слика 4.29. се прикажани функционалната и монтажната шема за реализација на оваа вежба. Потенциометарот има три приклучоци од кои два приклучоци се напојување и заземјување. **Средниот пин на потенциометарот** е контролен или податочен пин и него **го поврзуваме со аналогниот пин A_0** на Arduino Uno R3.



Слика 4.29. Поврзување на лед-диода и тример-потенциометар со Arduino Uno R3

Начинот на поврзување е следен:

(1) Го поставуваме потенциометарот на протоплочката при што треба да внимаваме неговите приклучоци да ги поставиме во контактни точки кои припаѓаат на различни проводници. Приклучоците на потенциометарот се нумерирани од 1 до 3. Приклучоците со број 1 и 3 се поврзуваат со лентите за напојување и заземјување. Пинот со број 2 , преку жолтата краткоспојница, директно се поврзува со аналогниот пин A0.

(2) Ја поставуваме лед-диодата на протоплочката исто како што беше објаснето во претходната вежба, катодата ја поврзуваме со лентата за заземјување, а анодата со 13-от пин.

(3) Пиновите означени со GND и 5V на Arduino Uno R3 ги поврзуваме со лентите за заземјување и напојување на протоплочката.

5. Пишување и внесување на програма за Arduino Uno R3

Arduino Uno R3 ја проверува вредноста на сигналот на аналогниот влезен пин A0 со употреба на инструкцијата analogRead(). Прочитаната вредност претставува променлива со симболично име sensorValue. Оваа променлива се запишува како време на доцнење во двете инструкции delay() кои следат после двете инструкции digitalWrite(), со кои се менува состојбата на лед-диодите.

Подолу е дадена програмата за контрола на фреквенцијата на трепкање на лед-диодата со потенциометар.

```
1  int sensorPin = A0;          // Избор на влезен пин за
                                // потенциометарот.
2  int ledPin = 13;             // Избор на пин за поврзување на лед-
                                // диода
3  int sensorValue = 0;         // Декларираме променлива за
                                // складирање на вредноста добиена од
                                // потенциометарот.
4  void setup() {               //
5    pinMode(ledPin, OUTPUT);   // Пинот ledPin се конфигурира како
                                // излезен
6  }                            // Крај на структурата setup
7  void loop() {                // Почеток на циклусот.
8    sensorValue =analogRead(sensorPin);
//Прочитаната вредност од потенциометарот се сместува во променливата со
//симболично име „sensorValue“
9    digitalWrite(ledPin, HIGH);  // Лед-диодата се вклучува
10   delay(sensorValue);          // Диодата свети онолку милисекунди
                                  // колку што изнесува вредноста добиена
                                  // од потенциометарот
11   digitalWrite(ledPin, LOW);   // Лед-диодата се исклучува
12   delay(sensorValue);          // Диодата нема да свети онолку
                                  // милисекунди колку што изнесува
                                  // вредноста добиена од потенциометарот.
                                  // Потоа повторно се враќаме на почетокот
                                  // од циклусот.
                                  //
13 }                             // Крај на циклусот.
```

После пишувањето на програмата истата се внесува во меморијата на Arduino Uno R3 како што беше претходно објаснето.

6. Резултати од реализација на вежбата

Кога ќе се вклучи напојувањето за Arduino Uno R3, со помош на шрафцигер се нагодува отпорноста на потенциометарот од минимум до максимум. Кога отпорноста е најголема тогаш вредноста на влезниот сигнал и времето на доцнење се најмали и лед-диодата ќе трепка со најголема брзина . Доколку времето на доцнење е премногу мало, можно е да не се забележи трепкањето на диодата. Да нагласиме дека максималното време на доцнење изнесува 1023.

Коментар:_____

7. Направи промена!

- Што ќе се случи доколку во инструкциите delay() променливата sensorValue се помножи со пет?

  Коментар:_____

- Што ќе се случи ако после читањето на вредноста на аналогниот влезен сигнал во програмскиот код се вметне нова инструкција procent=map(val,0,1023,0,100) како што е прикажано во пример 4.53?

Пример 4.53.

```
7   void loop() {
8     sensorValue =analogRead(sensorPin);
9     procent = map(val,0,1023,0,100);
10    digitalWrite(ledPin, HIGH);
11    delay(procent);
12    digitalWrite(ledPin, LOW);
13    delay(100-procent);
14    }
```

Коментар:_____

4.8.1.3. Практична вежба: Регулација на интензитетот на светлина на лед-диода

1. **Цел на вежбата**

Учениците ќе користат еден од аналогните излезни пинови на Arduino Uno R3 и софтверски ќе ја менуваат големината на излезниот напон. Промените на напонот ќе се следат преку употреба на лед-диода при што интензитетот на светлината постепено ќе се менува од минимум до максимум и обратно.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Arduino Uno R3 платформа
- протоплочка
- лед диода
- отпорник со отпорност R1=220Ω
- две жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

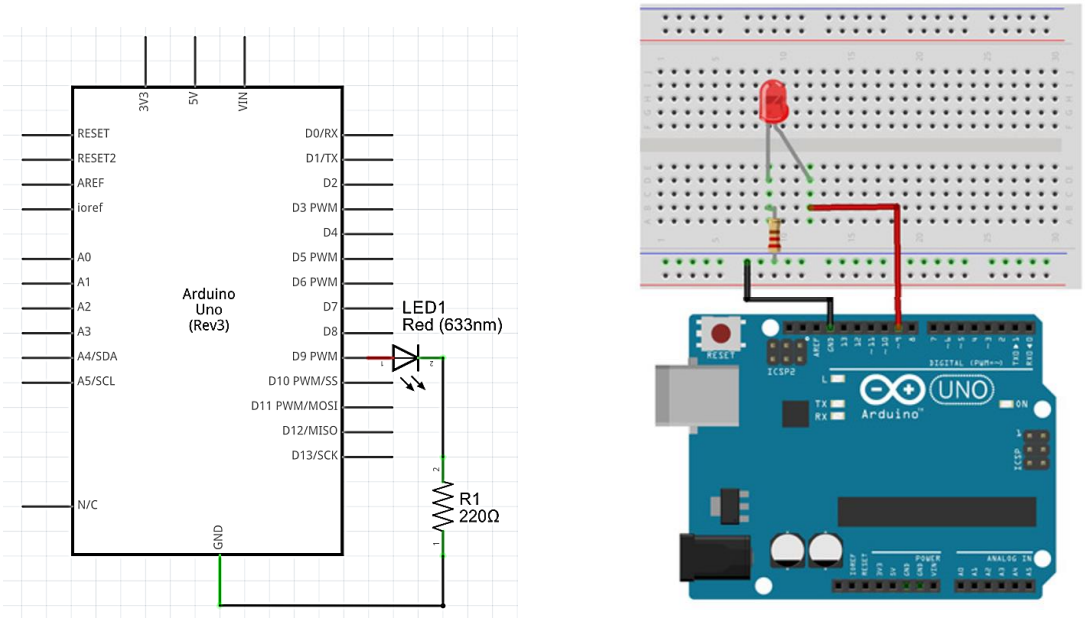
3. **Подготовка за вежбата**

Учениците треба да се потсетат на:

- импулсно ширинската модулација, нејзиното значење за аналогните излези на Arduino платформата и бројот на можни вредности
- инструкциите за работа со аналогните излезни пинови на Arduino платформата
- for структурата за повторување на циклуси

4. **Опис на електричната шема и начин на поврзување**

На слика 4.30. се прикажани функционалната и монтажната шема за реализација на оваа вежба.



Слика 4.30. Поврзување на лед-диода на PWM-пин од Arduino Uno R3

Колото е многу едноставно. Катодата на лед диодата, преку отпорникот R1, треба да се поврзе со лентата за заземјување, а анодата со деветтиот пин Arduino Uno R3. Во оваа вежба не можеме да го употребиме тринаесеттиот пин како во претходните вежби, бидејќи на него не можат да се запишуваат аналогни вредности.

5. Пишување и внесување на програма за Arduino Uno R3

Излезниот напон, со кој се контролира интензитетот на светлината на лед-диодата, го задаваме преку употреба на инструкцијата analogWrite() при што во средната заграда се внесува аналогната вредност. Минималната аналогна вредност изнесува 0, а максималната 255.

Програмата за регулација на интензитетот на светлината користи две „for“ структури за повторување на циклус. Во првата for структура, променливата ја зголемуваме за пет и циклусот се повторува сè додека не добиеме вредност 255. Во втората for структура, променливата се намалува за пет сè додека не се постигне вредност нула.

```
1  int ledPin = 9;                          //  Лед-диодата е поврзана на пин
                                            //  број 9.
2  void setup()  {                              Нема конфигурација на
3  }                                        //  пиновите
4  void loop()  {                           //  Почеток на циклусот
5    for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5)
   {
// За for структурата декларираме променллива со симболично име  „fadeValue“ и
// почетна вредност нула. Неа ја зголемуваме за 5, сè додека не добиеме број
// еднаков на 255. По секое зголемување или намалување се чека 30 милисекунди.

6      analogWrite(ledPin, fadeValue);      //
```

```
7     delay(30);                                    //   Го зголемуваме интензитетот
8    }                                              //   на светлината
9    for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5)
     {
     .
//Го намалуваме интензитетот на светлината. Променливата ја намалуваме за пет
//додека не добиеме вредност нула. Потоа се враќаме на почетокот од циклусот
10    analogWrite(ledPin, fadeValue);
11    delay(30);
12   }
13  }
```

Како и претходните програми така и оваа програма ја внесуваме во Arduino Uno R3 со следнава постапка:

(1) Ја поврзуваме Arduino Uno R3 со компјутер преку USB кабел
(2) Во главното мени ја избираме опцијата Tools и избираме вид на платформа и сериска порта
(3) Ја преведуваме програмата на машински јазик преку употреба на алатката Verify/Compile
(4) И на крај ја внесуваме програмата со притискање на алатката Upload

6. Резултати од реализација на вежбата

Веднаш после внесувањето на програмата електричното коло ќе почне да работи односно интензитетот на светлината на лед-диодата ќе почне наизменично да се намалува и зголемува.
Коментар:_____

7. Направи промена!

- Што ќе се случи ако го промениме чекорот на зголемување или намалување на аналогната вредност од пет на некоја друга вредност?
  Коментар:_____
- Што ќе се случи ако го зголемиме времето на доцнење во инструкцијата delay() ?

Коментар:_____

4.8.1.4. Практична вежба:Фотоотпорник за контрола на интензитет на светлина

1. Цел на вежбата

Во оваа вежба ќе го менуваме интензитетот на светлината на лед-диода со употреба на фотоотпорник. Имено лед-диодата ќе послужи како покажувач за осветленоста на околината во која е поставена Arduino Uno R3 заедно со

фотоотпорникот. Исто така во оваа вежба учениците ќе научат што значи и како се врши калибрација на сензор.

2. Потребни компоненти за реализација на оваа вежба се:

- Arduino Uno R3 платформа
- протоплочка
- лед-диода
- фотоотпорник
- отпорници со отпорност R1=220Ω и R2=10KΩ
- жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

Фотоотпорникот е полупроводнички елемент чиј електричен отпор се менува под влијание на светлината која паѓа врз неговата површина. Заради оваа особина наоѓа примена како електронски сензор. Пред да го вградиме фотоотпорникот во колото, со употреба на мултиметар, можеме да ја измериме неговата отпорност со тоа што ќе го менуваме интензитетот на светлината. Кога нема светлина неговата отпорност е најголема. Колку повеќе го доближуваме фотоотпорникот до изворот на светлина толку повеќе се намалува неговата отпорност.

3. Подготовка за вежбата

Учениците треба да се потсетат на:

- основните карактеристики на фотоотпорниците
- пин-дијаграмот на Arduino Uno R3
- инструкциите за работа со аналогни пинови и опсегот на можни вредности
- значењето на условната if структура
- примената на инструкцијата map() за промена на опсегот на вредности.

4. Опис на електричната шема и начин на поврзување

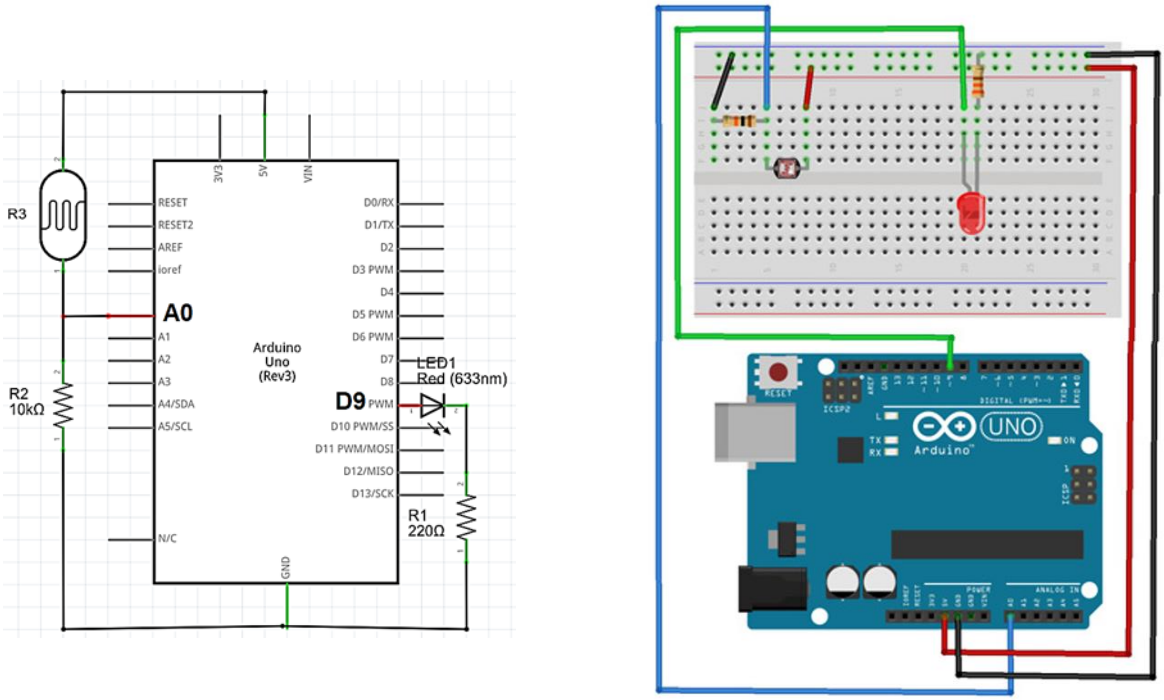
На слика 4.31. се прикажани функционалната и монтажната шема за реализација на оваа вежба.

(1) Го поставуваме фотоотпорникот на протоплочката при што треба да внимаваме неговите приклучоци да ги поставиме во контактни точки кои припаѓаат на различни проводници. Со црвена краткоспојница едниот крај на фотоотпорникот го поврзуваме со лентата за напојување. Другиот крај на фотоотпорникот, преку отпорникот R2 и црна краткоспојница го поврзуваме со лентата за заземјување. Всушност фотоотпорникот и отпорникот R2 формираат напонски делител. Со сината краткоспојница контактната точка меѓу фотоотпорникот и отпорникот R2 се поврзува со аналогниот влез A0.

(2) Како во претходната вежба, катодата на лед-диодата, преку отпорникот R1 и црната краткоспојница, ја поврзуваме со лентата за заземјување.

Со зелената краткоспојница ја поврзуваме анодата со деветтиот пин со ознака PWM, што значи дека е аналоген излез.

(3) Пиновите GND и 5V на Arduino Uno R3 ги поврзуваме со лентите за напојување и заземјување.



Слика 4.31. Поврзување на лед-диода и фотоотпорник со Arduino Uno R3

Arduino Uno R3 го мери напонот добиен од напонскиот делител и измерените вредности ги претвора правопропорционално во целобројни вредности во опсег од 0 до 1023. Целобројната вредност 1023 одговара на напон од 5V, а нулата на напон од 0V. Но, поради падот на напонот на отпорникот R2 влезниот напон на пинот A0 никогаш нема да достигне вредност 5V, ниту пак влезниот сигнал ќе добие максимална вредност. Исто така, фотоотпорникот е дизајниран да детектира промени во интензитетот на светлината во многу поширок опсег него што се промените на светлината во услови на затворен простор. Од овие причини потребно е да се изврши калибрација на фотоотпорникот. Калибрацијата е неопходна за сензорот да работи прецизно. Во мерната техника под поимот калибрација на мерен инструмент подразбираме утврдување на отстапувањето на вредноста која се отчитува од мерниот инструмент во однос на вистинската вредност (вредноста на еталонот). Во однос на Arduino Uno R3 и фотоотпорникот отстапувањето претставува разлика помеѓу максималната или минималната вредност на влезниот сигнал што ја дава самата платформа и максималната или минималната вредност што може да ја детектира самиот фотоотпорник. Крајните вредности во опсегот од 0 до 1023 никогаш нема да се постигнат што автоматски значи дека лед-диодата секогаш ќе свети, но никогаш нема да го достигне својот максимум или минимум. За да

се елиминира ваквото отстапување **треба да се нагодат максималната и минималната вредност на сензорот** уште во првите неколку секунди по пуштањето во работа на електричното коло. Тоа се постигнува со движење на раката кон и од сензорот, при што платформата софтверски ги одбира опсегот на максимална и минимална вредност.

5. Пишување и внесување на програма за Arduino Uno R3

Калибрацијата се извршува само еднаш, таа не се повторува и затоа истата се пишува како програмски код во составот на setup задолжителната структура. Поради потребата од калибрација се воведени две нови променливи со симболични имиња lightLow и lightHigh. Променливата lightHigh постојано се зголемува од нула до максималната вредност што може да ја детектира фотоотпорникот во дадена физичка средина. Променливата lightLow постојано се намалува од 1023 до минималната вредност на фотоотпорникот. Инструкцијата prilagodilightLevel = map(lightLevel, lightHigh, lightLow, 0, 255) врши претворање на влезниот опсег со калибрирани вредности на минимумот и максимумот во излезен опсег на вредности, од 0 до 255.

Подолу е дадена програмата со коментари за нејзините инструкции и структури.

```
//Вообичаено се започнува со избор на пинови за поврзување на фотоотпорникот и
//лед-диодата.
1   const  int sensorPin = 0;        //   Сензорот е поврзан на нултиот аналоген пин.
                                     //
2   const  int ledPin = 9;           //   Лед-диодата е поврзана на деветтиот пин на
                                     //   чиј излез има ширински модулирани импулси.
                                     //
3   int lightLevel;                  //   Променливата ги чува вредностите добиени
                                     //   од фотоотпорникот.
4   int prilagodilightLevel;             Променливата со симболично име
                                         prilagodilightLevel се користи при промена на
                                         опсегот од влез за излез
5   int lightHigh = 0;               //   Променливата lightHigh претставува
                                     //   максимална вредност после извршената
                                     //   калибрација.
6   int lightLow = 1023;             //   Променливата lightHigh претставува
                                     //   минимална вредност после извршената
                                     //   калибрација.
7   void setup()
8   {
9     pinMode(ledPin, OUTPUT);       //   Конфигурација на излезен пин за лед-
                                     //   диодата.
//Калибрацијата се извршува во првите пет секунди од почетокот на извршувањето
// на програмата.
10    while (millis() < 5000) {      //   Со инструкцијата millis() се мери времето
                                     //
```

```
// Се прочитува вредноста на напонот на пинот sensorPin (фотоотпорникот) и
започнува калибрацијата.
11        lightLevel = analogRead(sensorPin);
// Променливата lightLow добива вредност што одговара на минималниот интензитет
// на светлината во дадената средина.
12        if (lightLevel < lightLow) {
13          lightLow = lightLevel;
14        }
//Променливата lightHigh добива вредност што одговара на максималниот интензитет
// на светлината во дадената средина
15        if (lightLevel > lightHigh) {
16          lightHigh = lightLevel;
17        }                                  //    Крај на if структура.
18      }                                    //    Крај на while структура.
19    }                                      //    Крај на setup структура.
20    void loop()
21    {                                      //    Почеток на структурата loop.
//  Се прочитува вредноста на напонот на пинот sensorPin (фотоотпорникот).
22        lightLevel = analogRead(sensorPin);
// Преминот од еден во друг опсег е линеарен.
23        prilagodilightLevel = map(lightLevel, lightHigh, lightLow, 0, 255);
24      analogWrite(ledPin, prilagodilightLevel);
// Сигналот што одговара на осветленоста на фотоотпорникот ја побудува
// лед-диодата.
25    }                                      //    Крај на структурата loop
```

Откако ќе ја напишеме програмата истата се внесува во Arduino Uno R3 според веќе објаснетата постапка во претходните примери.

6. Резултати од реализацијата на вежбата

Кога електричното коло ќе го поврземе со напојувањето, веднаш во првите пет секунди, вршиме калибрација така што ја движиме раката кон и од сензорот. Потоа проверуваме дали лед-диодата посилно свети кога врз фотоотпорникот паѓа повеќе светлина отколку кога фотоотпорникот е покриен со рака или друг предмет.

Коментар:_____

7. Направи промена!

- Како потврда на сè што е кажано за калибрацијата, можеме да ги прикажеме добиените резултати на екранот од серискиот монитор. За таа цел треба да ја дефинираме брзината на пренос во setup структурата со употреба на инструкцијата Serial.begin(9600) и да ги внесеме инструкциите Serial.println(lightLow) и Serial.println(lightHigh) после извршената калибрација и Serial.println(lightLevel) пред мапирањето односно пред промената на влезниот опсег вредности во излезен опсег.

  Коментар:_____

- Наместо фотоотпорник, можеме да поставиме каков било сензор, и неговите вредности ќе бидат пресликани во светлината што ја емитува лед-диодата. Исто така, наместо лед-диода можеме да поставиме пиезо-сензор кој ќе испушта тонови или мотор чија брзина ќе зависи од осветленоста на фотоотпорникот. Ваквите промени во програмскиот код бараат добро познавање на инструкциското множество на Arduino платформата.
  Коментар:_____

4.8.1.5. Практична вежба:Поврзување на Arduino Uno R3 со серво-мотор

1. Цел на вежбата

Целта на оваа практична вежба е придвижување на серво-мотор за 180 степени. За програмирање на серво-моторот и негова контрола учениците треба да ја повикаат библиотека Servo.h., со која се запознавме во наставната единица 4.3.7. Инструкции за работа со библиотеки. Ова е прва практична вежба за програмирање на Arduino Uno R3, од досега реализираните, која користи библиотека.

2. Потребни компоненти за реализација на оваа вежба се:

- Arduino Uno R3 платформа
- протоплочка
- серво-мотор
- жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

Принципот на работа на серво моторот е прикажана на слика 4.11.

3. Подготовка за вежбата

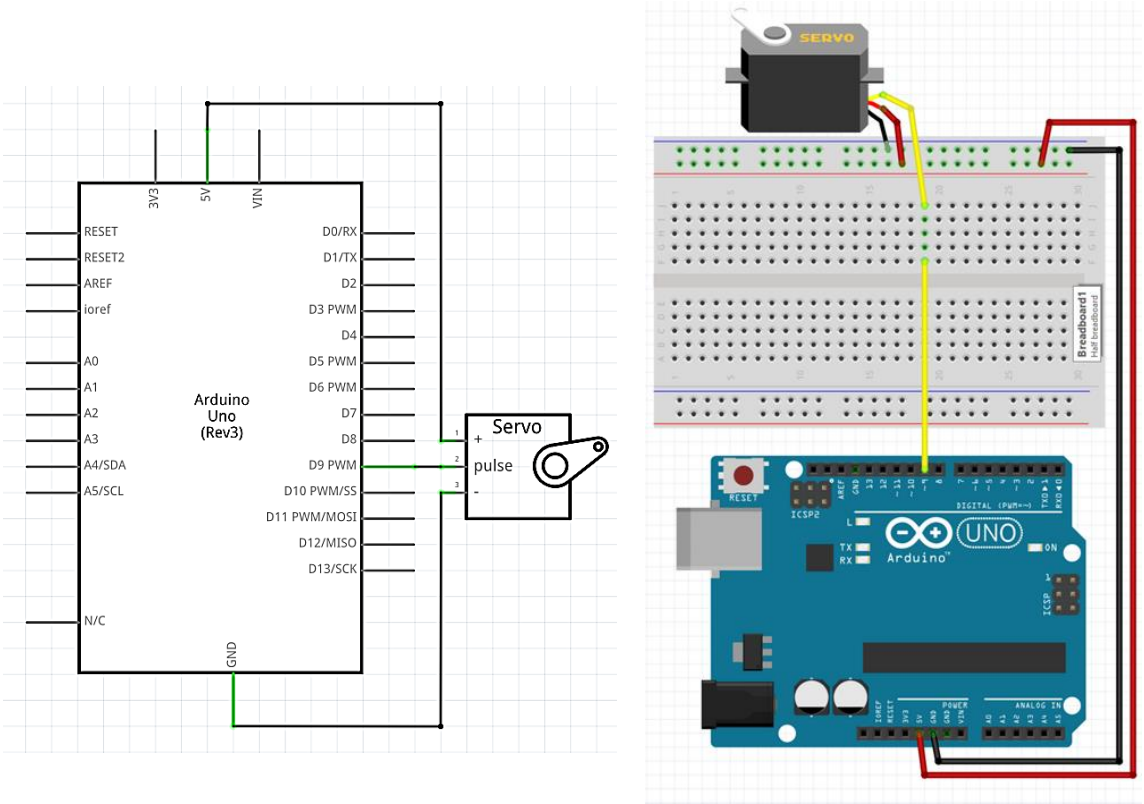
Учениците треба да се потсетат на:

- пин-дијаграмот на Arduino Uno R3
- принципот на работа на серво-моторот
- инструкциите кои се дел од библиотеката Servo.h
- значењето на for структура за повторување на циклуси

4. Опис на електричната шема и начин на поврзување

На слика 4.32. се прикажани функционалната и монтажната шема за реализација на оваа вежба. Монтажната шема е многу едноставна. Серво-моторот има три жици. Црната жица е за заземјување, црвената е за напојување и жолтата жица е за контрола на моторот и преку неа серво-моторот е поврзан со 9-тиот пин од Arduino Uno R3. Да се потсетиме, само пиновите 9 и 10 имаат поддршка за поврзување на серво-мотор. Со поставување кондензатор со капацитивност 100 μF на пиновите за заземјување и напојување, можат да се избегнат брзите

промени на напонот. Серво-моторот е излезен уред. Во оваа вежба нема влезен уред и серво-моторот ќе го контролираме чисто софтверски, само со инструкции.



Слика 4.32. Поврзување на серво-мотор со Arduino Uno R3

5. Пишување и внесување на програма за Arduino Uno R3

Подолу е дадена програма со којашто серво-моторот ќе се заврти за 180 степени, од лево кон десно, со чекор од два степена при секое повторување на структурата for.

```
1  #include <Servo.h>          //  Вклучуваме библиотека за контрола на серво-
                               //  мотор.
2  Servo myServo;              //  Дефинираме серво-објект со симболично
                               //  име myServo. На овој објект ќе се однесуваат
                               //  наредбите од библиотеката Servo.
3   int position;              //  Променливата position ја содржи вредноста
                               //  на аголот за кој треба да се заврти серво-
                               //  моторот.
4  void setup()
5  {                           //  Почеток на функцијата setup.
6    myServo.attach(9);        //  Деветтиот пин го конфигурираме како пин на
                               //  којшто ќе биде поврзан серво-моторот со
7  }                           //  Крај на функцијата setup.
8  void loop()
9  {                                Почеток на функцијата loop.
   //Со структурата for, аголот на вртење постепено го зголемуваме за два
   //степена,почнувајќи од нула до максималната вредност 180 степени.
```

```
10   for(position = 0; position < 180; position += 2) {
11         myServo.write(position);
     //При секое повторување на for структурата, серво-моторот се
     // поместува за два степени во десно.
12         delay(20);          //  Краткото време на доцнење служи за
                               //  стабилизација на положбата пред новото
                               //  завртување.
13    }                        //  Крај на структурата for.
14   }                         //  Крај на функцијата loop.
```

Библиотеката ја вклучуваме со притискање на главното мени Sketch→Import Library, по што се отвора паѓачко мени. Библиотеката Servo.h е стандардна библиотека во составот на развојната средина на Arduino платформата. Програмата ја внесуваме со постапката: поврзување на Arduino платформата со компјутер → избор на платформа и сериска порта → верификација → внесување.

6. Резултати од реализацијата на вежбата

Веднаш после внесувањето на програмата електричното коло ќе почне да работи така што серво-моторот ќе почне да се врти од лево кон десно, за пола круг. Кога ќе се заврти за 180 степени серво-моторот застанува.

Коментар:_____

7. Направи промена!

- Во горната програма може да се додаде уште една for структура со којашто аголот на завртување ќе се врати назад, од 180 степени до нула. Во тој случај од вредноста на аголот треба да се одземе два. На тој начин, серво-моторот ќе се врти напред-назад.

```
14   for(position = 0; position < 180; position - = 2) {
15         myServo.write(position);
16         delay(20);
17   }
```

Коментар:_____

- За контрола на аголот на завртување на серво-моторот може да се користи и потенциометар, како влезен уред приклучен на еден од аналогните пинови. Се разбира, во тој случај треба да се направат поголеми промени во горенаведената програма. Во задолжителната loop структура ќе треба да се употребат инструкциите: analogRead() за читање на вредноста на потенциометарот, map() за промена на влезниот опсег во излезен (од 0-1023 кон 0-255) и myServo.write() за придвижување на серво-моторот.

Коментар:_____

4.8.1.6. Практична вежба: Поврзување на Arduino Uno R3 со LCD-екран

1. **Цел на вежбата**

Целта на оваа вежба е да го испишеме текстот „Hello world“на од LCD-екран и изминатото време изразено во милисекунди.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Arduino Uno R3 платформа
- протоплочка
- LCD-екран, модел GDM1602K
- Тример-потенциометар со отпорност 10KΩ
- жици за поврзување (краткоспојници)
- компјутер со инсталирана развојна средина за Arduino платформа

На слика 4.10 е прикажан пин-дијаграмот на LCD-екранот. Во наставната единица 4.3.7. Инструкции за работа со библиотеки беа опишани неколку инструкциите во состав на LiquidCrystal библиотеката. При изборот на LCD-екран треба да провериме дали контролерот во составот на екранот е компатибилен со LiquidCrystal библиотеката, за што ќе ни послужи техничко-технолошката документација на екранот. На пример, стандардни контролери компатибилни со оваа библиотека се HD44780 или ST7066U . Тример-потенциометарот се користи за нагодување на контрастот.

3. **Подготовка за вежбата**

Учениците треба да се потсетат на:

- пин-дијаграмот на Arduino Uno R3
- пин-дијаграмот на LCD-екранот, модел GDM1602K
- инструкциите за работа со LCD-екран

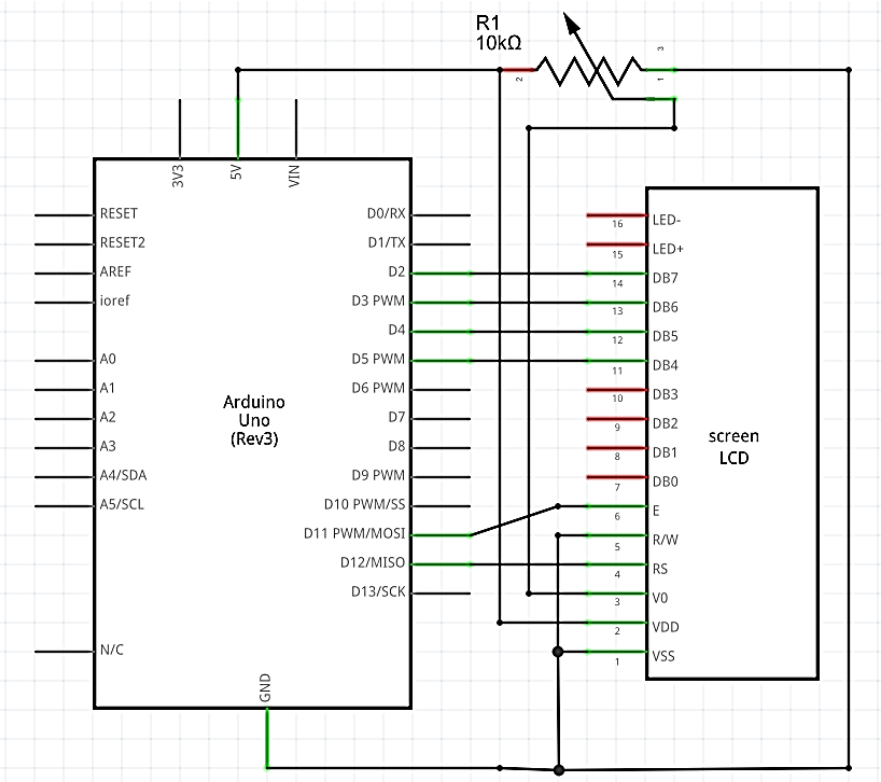
4. **Опис на електричната шема и начин на поврзување**

На слика 4.33. е прикажан начинот на поврзување на LCD-дисплејот со Arduino Uno R3.

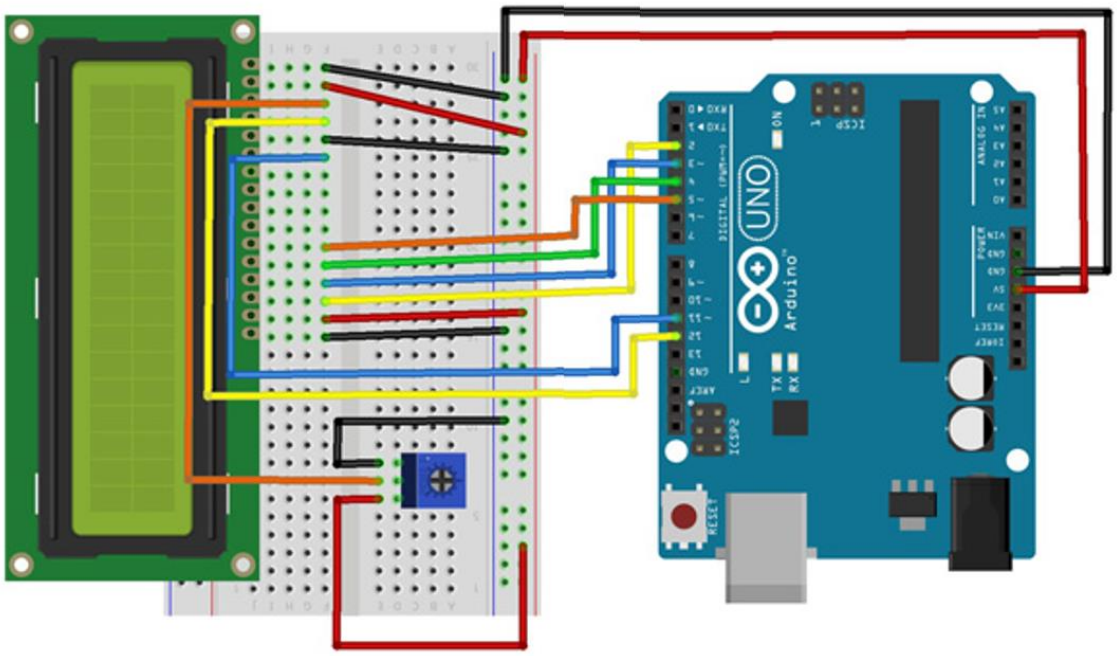
Пиновите на LCD-дисплејот треба да се поврзат со пиновите на Arduino Uno R3 на следниот начин:

	Приклучоци на LCD-екранот		Пинови на Arduino R3
(1)	RS	→	дигиталниот пин 12
(2)	Enable пинот со	→	дигиталниот пин 11
(3)	D4 пинот со	→	дигиталниот пин 5
(4)	D5 пинот со	→	дигиталниот пин 4
(5)	D6 пинот со	→	дигиталниот пин 3
(6)	D7 пинот со	→	дигиталниот пин 2

(7) R/W → пинот со заземјување
(8) V_{SS} → пинот со заземјување
(9) V_{CC} → пинот со напојувањето од 5 V



Слика 4.33. Електричната шема за поврзување на LCD-екран со Arduino Uno R3



Слика 4.34. Монтажна шема за поврзување на LCD-екран со Arduino Uno R3

Пинот VEE е поврзан со вториот приклучок на потенциометарот преку портокаловата краткоспојница. Првиот и третиот приклучок на потенциометарот се поврзани со лентите за напојување и заземјување.

5. **Пишување и внесување на програма за Arduino Uno R3**

Следува изработката на софтверот во развојната средина.

```
1  #include <LiquidCrystal.h>      //  Се вклучува библиотеката за работа со LCD-
                                   //  дисплеј, која е стандардна библиотека
                                   //
2  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
   // Ги декларираме пиновите на Arduino Uno R3 за поврзување со дисплејот.
3  void setup() {                  //
4    lcd.begin(16, 2);             //  Започнува комуникацијата со LCD-дисплејот
                                   //  преку поставување на бројот на редици и
                                   //  колони.
5    lcd.print("hello world!");    //  На екранот се печати текстот.
6  }
7  void loop() {
8    lcd.setCursor(0, 1);          //  Се избира активното поле од вкупно 32,
                                   //  преку избор на колона и редица. Да
                                   //  нагласиме, редиците и колоните се бројат
                                   //  почнувајќи од нула
9    lcd.print(millis()/1000);     //  На екранот се печати времето изразено во
                                   //  секунди, изминато од последното
                                   //  ресетирање на Arduino Uno R3
10 }
```

Откако ќе ја напишеме програмата, истата ја внесуваме во меморијата на Arduino Uno платформата со постапката: поврзување на Arduino Uno R3 со компјутер → избор на платформа и сериска порта → верификација → внесување.

6. **Резултати од реализацијата на вежбата**

Веднаш после внесувањето на програмата електричното коло ќе почне да работи така што на екранот ќе се појави текстот „hello world” и измереното време.
Коментар:_____

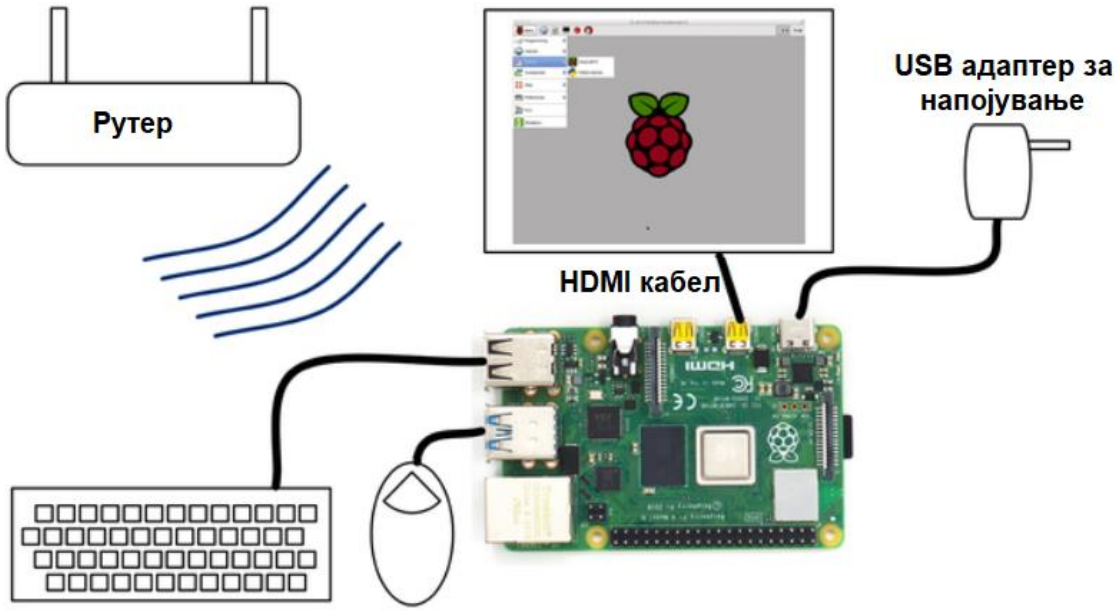
7. **Направи промена!**

Во самата развојна средина постојат голем број готови програми за LCD-екран. Тие може да се отворат и да се анализираат со притискање на File→Example→LiquidCrystal. Ќе наброиме неколку од нив: автоматско поместување на текстот, трепкање на текстот, печатење текстуални форми, избор на насока на текстот и др.
Коментар:_____

4.8.2. Практични вежби за програмирање на Raspberry Pi 3B+ во програмскиот јазик Python

За успешна реализација на практичните вежби, потребни се следните предзнаења: инсталација на оперативен систем Raspbian, програмирање во програмски јазик Python, користење на библиотеката GPIOZERO и поврзување и карактеристики на основни влезно-излезни уреди.

При изведувањето на вежбите, Raspberry Pi 3B+ треба да биде поврзан со тастатура за внесување на текстот на програмскиот код и со монитор за прикажување на текстуалните резултати добиени по извршувањето на програмата. На слика 4.35. е прикажан еден комплетен Raspberry Pi систем. Да се потсетиме, за поврзување со тастатура се користи USB-приклучокот, а за поврзување со монитор HDMI-приклучокот. Исто така, не постои ограничување во изборот на модел на Raspberry Pi, бидејќи дадените кодови се компатибилни со сите модели.



Слика 4.35. Raspberry Pi систем

Во секоја практична вежба, дадена е шемата за поврзување на Raspberry Pi 3B+ со влезно-излезните уреди, преку употреба на протоплочка. Потоа следува програмскиот код со соодветни коментари. Со програмските кодови може да се експериментира преку менување на вредностите на некои параметри: брзина на вртење на моторот, фреквенција на трепкање на диодата, боја на RGBLED-диода, прикажан текст итн.

4.8.2.1. Мерки за заштита и безбедност при работа со Raspberry Pi

За успешна и безбедна реализација на практичните вежби мора да се почитуваат општите правила наведени во наставната единица 2.8.1. Мерки за заштита и безбедност при работа. Но, исто така треба да се познаваат и специфичните карактеристики на електронската опрема кои најчесто се содржат во нејзината техничко-технолошката документација. Најчести причини за оштетување на Raspberry Pi се несоодветно напојување и непознавањето на карактеристиките на поврзаните влезно-излезни компоненти. Најважни мерки за заштита при работа со Raspberry Pi се:

- Изборот на изворот за напојување е мошне важен. Се користи USB адаптер со 5V максимален напон за напојување. Да нагласиме дека Raspberry Pi 4 модел B користи адаптер со USB-C приклучок, а останатите модели микро USB.
- Струјата која треба да ја обезбеди изворот за напојување зависи од моделот на Raspberry Pi и од потрошувачката на електрична енергија на поврзаните влезно-излезни компоненти. Максимално дозволената струја изнесува од 1,5A до 2A. Поради поголемата потрошувачка на електрична енергија на Raspberry Pi 4 модел B му е потребна струја со јачина од 3A.
- GPIO подножјето содржи 4 пинови за напојување и осум пинови за заземјување. Два од пиновите за напојување имаат константен напон од 5V, а другите два 3,3V.
- Максимално дозволениот напон за GPIO влезните пинови изнесува 3,3V.
- Пред почетокот на секоја монтажа, Raspberry Pi треба да се исклучи од неговиот извор за напојување.
- Бидејќи Raspberry Pi е осетлив на статички електрицитет, пред почетокот на монтажата потребно е да се допре некаква метална површина. На таков начин доаѓа до празнење на човековото тело.
- Пред да се вклучи изворот за напојување задолжително да се провери дали сите електронски компоненти се поврзани како што е наведено во функционалната или монтажната шема.
- Пиновите никогаш не смеат да се поврзуваат меѓусебно бидејќи може да дојде до нивно оштетување.
- За нагодување на работната струја да се користат отпорници. На пример, ако лед-диодата ја поврземе директно на пинот за напојување од 5V ќе дојде до загревање и оштетување на истата.
- За поврзување на мотори или други индуктивни потрошувачи да се користат H-мостови (анг. motor driver)

4.8.2.2. Практична вежба: Вклучување на лед-диода со тастер

1. **Цел на вежбата**

Цел на вежбата е поврзување на лед-диода и тастер со Raspberry Pi 3B+ со употреба на протоплочка, пишување на програма во Python програмскиот јазик, внесување на програмата во Raspberry Pi преку алатките на Thonny Python интегрираната развојна средина и испитување на работата на електричното коло.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем. Системот е составен од Raspberry Pi 3B+ со адаптер за напојување, тастатура, глувче и монитор.
- протоплочка
- тастер за печатена плочка
- лед-диода
- отпорник со отпорност R1=220Ω
- жици за поврзување (краткоспојници)

3. **Подготовка за вежбата**

Учениците треба да се потсетат на:

- пин-дијаграм на GPIO подножјето
- начинот на поврзување на лед-диода и тастер на протоплочка. Истиот е објаснет во упатството за работа со протоплочка.
- инструкциите од GPIO библиотеката класи Led и Button за работа со тастер и лед-диода.
- практичната вежба 4.8.1.1. Вклучување на лед-диода со тастер со примена на Arduino Uno R3 и да се направи споредба на електричната шема и програмскиот код

4. **Опис на електричната шема и начин на поврзување**

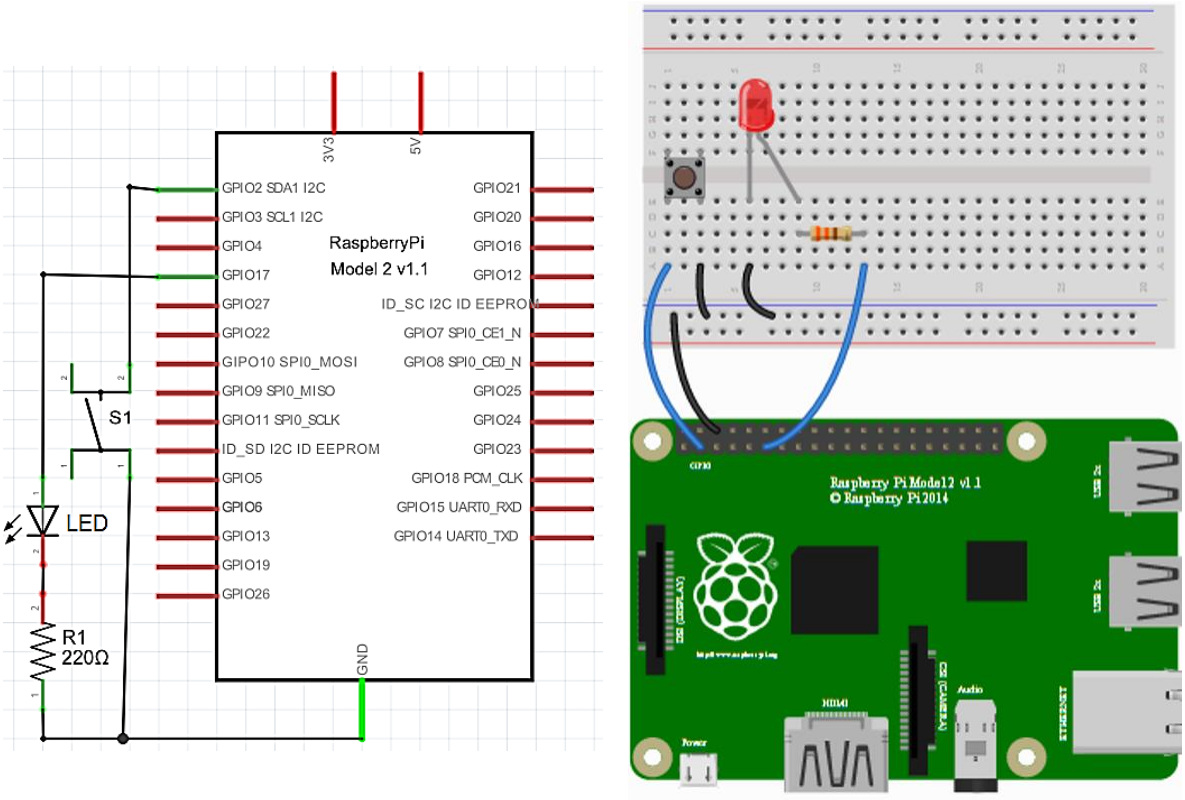
На слика 4.36. се прикажани функционалната и монтажната шема за реализација на оваа вежба.

(1) Тастерот се поставува на средината од протоплочката. Десниот приклучок на тастерот се поврзува со лентата за заземјување, а левиот приклучок со GPIO-пинот број 2.

(2) Катодата на лед-диодата се поврзува со лентата за заземјување, а анодата со GPIO-пинот број 17.

(3) Лентата за заземјување на протоплочката се поврзува со GND пинот на Raspberry Pi 3B+

Електричните шеми на слика 4.28. и слика 4.36. се различни. Кај Raspberry Pi тастерот не е поврзан со лентата за напојување на протоплочката. Имено кога тастерот не е притиснат тогаш GPIO-пинот број 2 не е поврзан, колото е отворено. Кога тастерот е притиснат GPIO-пинот број 2 се поврзува со лентата за заземјување. Ниското логичко ниво (анг. High) на GPIO-пинот број 2 е показател дека тастерот бил притиснат. Лед-диодата свети кога GPIO-пинот број 2 е на високо логичко ниво.



Слика 4.36. Поврзување на лед-диода и тастер со Raspberry Pi 3B+

5. Пишување и извршување на програма за Raspberry Pi 3B+

На слика 4.14. е прикажан изгледот на интегрираната развојна средина Tonny Python IDE која ќе ја користиме за пишување на програмата и нејзино извршување. Отвораме нов проект со алатката New, го пишуваме програмскиот код во уредувачот на текст и откако ќе ја напишеме ја зачувуваме (анг. Save). Кога ќе ја зачуваме називот на програмата се менува од untitled во името што сме го задале. Делот именуван како Shell служи за прикажување на текстуални пораки со употреба на инструкцијата print(). Во малата заграда се пишува текстот кој треба да се прикаже, напишан меѓу наводници. Подолу е даден програмскиот код за реализација на оваа вежба.

```
1  from gpiozero import LED, Button    # Вклучуваме два дела од библиотеката
                                       # GPIOZERO за работа на лед-диода и
                                       # тастер.
```

```
2  from signal import pause          # Вклучуваме дел од библиотеката signal за
                                     # внесување време на доцнење.

3  led = LED(17)                     # Конфигурираме пин за поврзување на
                                     # диодата.
4  button = Button(2)                # Конфигурираме пин за поврзување на
                                     # тастерот.
5  button.when_pressed = led.on      # Ја вклучуваме лед-диодата кога тастерот е
                                     # притиснат.
6  button.when_released = led.off    # Ја исклучуваме лед-диодата кога тастерот
                                     # е отпуштен.
7  pause()                           # Внесуваме време на доцнење. Стандардно
                                     # време на доцнење е една секунда. Времето
                                     # го менуваме со внесување вредност во
                                     # малата заграда.
```

За да се изврши напишаната програма потребно е да ја притиснеме иконата Run. Споредбено со Arduino платформата, Raspberry Pi е микрокомпјутер со свој оперативен систем. Интерпретерот ја преведува програмата инструкција по инструкција дури во текот на самото пишување на програмата и не се потребни алатките Verify и Upload.

6. Резултати од реализацијата на вежбата

Веднаш после стартот на програмата треба да го притиснеме тастерот и лед-диодата ќе светне. Кога тастерот ќе го отпуштиме лед-диодата ќе престане да свети

Коментар:_____

7. Направи промена!

- Сакаме лед-диодата да свети определено време и после отпуштањето на тастерот. Која инструкција ќе се употреби и каде ќе се вметне истата?

  Коментар_____
- Сакаме лед-диодата да свети кога тастерот е отпуштен и да не свети кога тастерот е притиснат. Кои промени треба да се направат во програмскиот код?

  Коментар_____

4.8.2.3. Практична вежба:Семафор

1. Цел на вежбата

Светлата на семафорот се вклучуваат и исклучуваат според точно определен тајминг и циклусот е бесконечен односно непрекинато се повторува.

Цел на оваа практична вежба е управување со светлата на семафорот преку употреба на Raspberry Pi 3B+. Во gpiozero библиотеката постои посебна класа токму за работа со семафор (анг. TrafficLights) и истата значајно го поедноставува пишувањето на програмскиот код.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем
- протоплочка
- три лед-диоди
- три отпорници со отпорност R=220Ω
- жици за поврзување (краткоспојници)

3. **Подготовка за вежбата**

Од хардверот на Raspberry Pi 3B+ учениците треба да се потсетат на пин-дијаграмот на GPIO подножјето и начинот на поврзување на лед-диоди на протоплочка. Истиот е објаснет во упатството за работа со протоплочка.

Исто така, пред почетокот на вежбата учениците треба да се запознаат со класата TrafficLights од библиотеката gpiozero и инструкцијата sleep од библиотеката time.

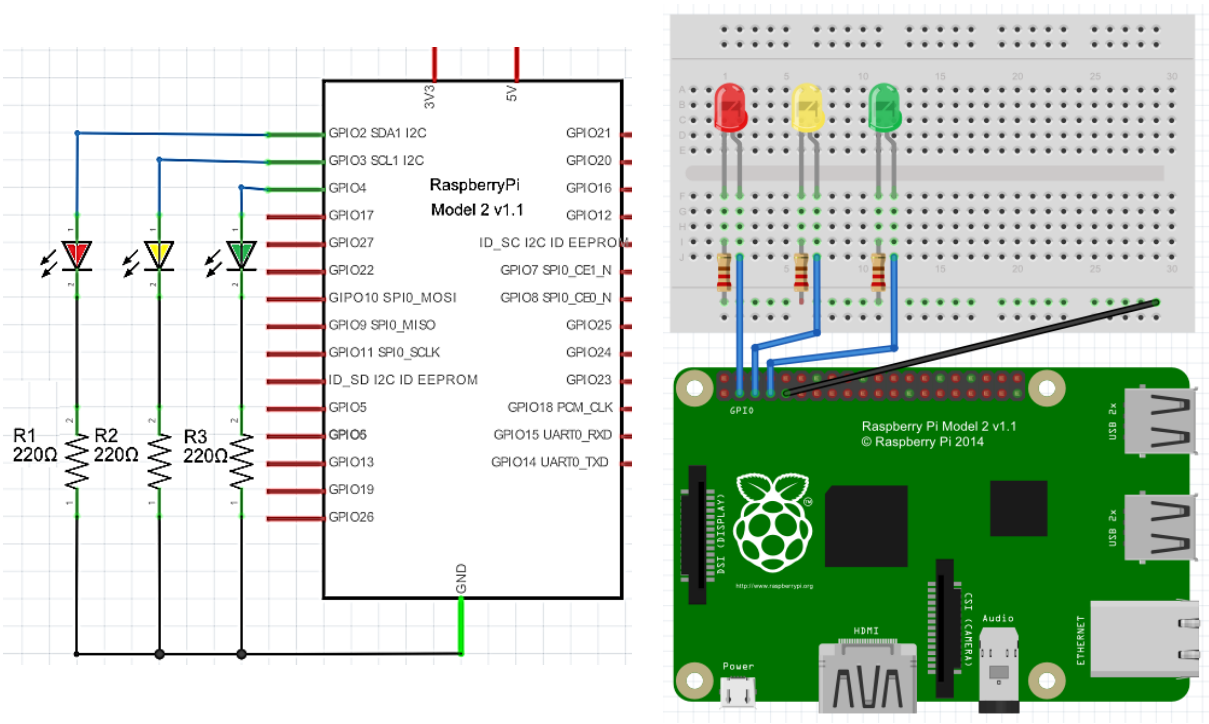
Пример 4.54.

```
1 semafor=TrafficLights(25, 8, 7)
```

Пример 4.54. е пример за декларација на објект кој ќе биде носител на класата TrafficLights(). Во програмата на оваа практична вежба деклариран е еден објект од оваа класа зошто постои само еден семафор. Доколку сакаме да го проширime системот со уште еден семафор, кој ќе управува со сообраќајот по втората улица од раскрсницата тогаш ќе треба да декларираме уште еден објект од класата TrafficLights(). Броевите 25, 8 и 7 во малата зграда се броеви на GPIO пиновите на кои се поврзани трите светла односно трите диоди. Од библиотеката за контрола на време се користи само инструкцијата sleep(). Во малата заграда се наведува времето изразено во секунди за кое системот ќе биде во мирување.

4. **Опис на електричната шема и начин на поврзување**

Поврзувањето на лед-диодите со Raspberry Pi 3B+ е многу едноставно. При поставувањето на лед-диодите на протоплочката треба да внимаваме нивните приклучоци да ги поставиме во контактни точки кои припаѓаат на различни проводници. Катодата на секоја лед-диода е поврзана со лентата за заземјување преку отпорник од 220Ω. Анодата на црвената лед-диода е поврзана со GPIO-пинот број 2, анодата на жолтата лед-диода со GPIO-пинот број 3 и анодата на зелената лед-диода со GPIO-пинот број 4. Да не заборавиме пинот GND на Raspberry Pi 3B+ да го поврземе со лентата за заземјување преку црна краткоспојница.



Слика 4.37. Поврзување на семафор со Raspberry Pi 3B+

5. Пишување и внесување на програма за Raspberry Pi 3B+

Постапката за пишување и внесување на програмата е иста како во претходната вежба. Прво отвораме нов проект, ја пишуваме во уредникот за текст, ја сочувуваме и именуваме и на крај ја извршуваме со притискање на алатката Run. Подоле е даден програмскиот код за оваа практична вежба.

```
1  from gpiozero import TrafficLights      # Се вклучува делот од библиотеката
                                           # GPIOZRO за работа со семафор.
2  from time import sleep                  # Со инструкцијата sleep го внесуваме
                                           # уредот во режим на мирување.
3  lights = TrafficLights(2, 3, 4)         # Конфигурација на пиновите за
                                           # црвената, жолтата и за зелената лед-
                                           # диода.
4  lights.green.on()                       # Се вклучува зелената лед-диода.
5  while True:                             # Започнуваме циклус што бесконечно
                                           # се повторува.
6      sleep(10)                           # Зелената лед-диода свети 10s .
7      lights.green.off()                  # Ја исклучуваме зелената лед-диода.
8      lights.amber.on()                   # Ја вклучуваме жолтата.
9      sleep(1)                            #
10     lights.amber.off()                  # Ја исклучуваме жолтата.
11     lights.red.on()                     # Вклучуваме црвена лед-диода.
12     sleep(10)                           # Таа свети десет секунди.
13     lights.amber.on()                   # Се вклучува жолтата лед-диода.
14     sleep(1)
15     lights.red.off()
```

```
16      lights.amber.off()
17      lights.green.on()
        sleep(2)
```

6. **Резултати од реализацијата на вежбата**

Веднаш после притискањето на алатката Run во лентата со алатки во Thonny Python интегрираната развојна средина зелената лед-диода ќе почне да свети 10 секунди, после што светнува жолтата лед-диода за една секунда, па свети црвената лед-диода со времетраење 10 секунди. На крајот светнуваат црвената и жолтата лед-диода заедно со времетраење од две секунди и повторно се враќаме на зелената лед-диода. Циклусот се повторува бесконечно.
Коментар_____

7. **Направи промена!**

Учениците може да го менуваат времето на светење на лед-диодите. Исто така може да додадат уште еден семафор кој ќе работи инверзно во однос на првиот и за таа цел треба да го променат програмскиот код.
Коментар_____

4.8.2.4. Практична вежба: Времето на реакција

1. **Цел на вежбата**

Целта на оваа вежба е да се утврди кој од два тастери ќе биде прв притиснат. Лед-диодата светнува и го означува почетокот на „трката“. Резултатот се прикажува во делот именуван како Shell во облик на текстуална порака. [3]

2. **Потребни компоненти за реализација на оваа вежба се:**

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем
- протоплочка
- една лед-диода
- еден отпорник со отпорност R=220Ω
- два тастери
- жици за поврзување (краткоспојници)

3. **Подготовка за вежбата**

Слично како во претходната вежба учениците треба да се потсетат на пин-дијаграмот на GPIO подножјето и начинот на поврзување на лед-диоди на протоплочка.

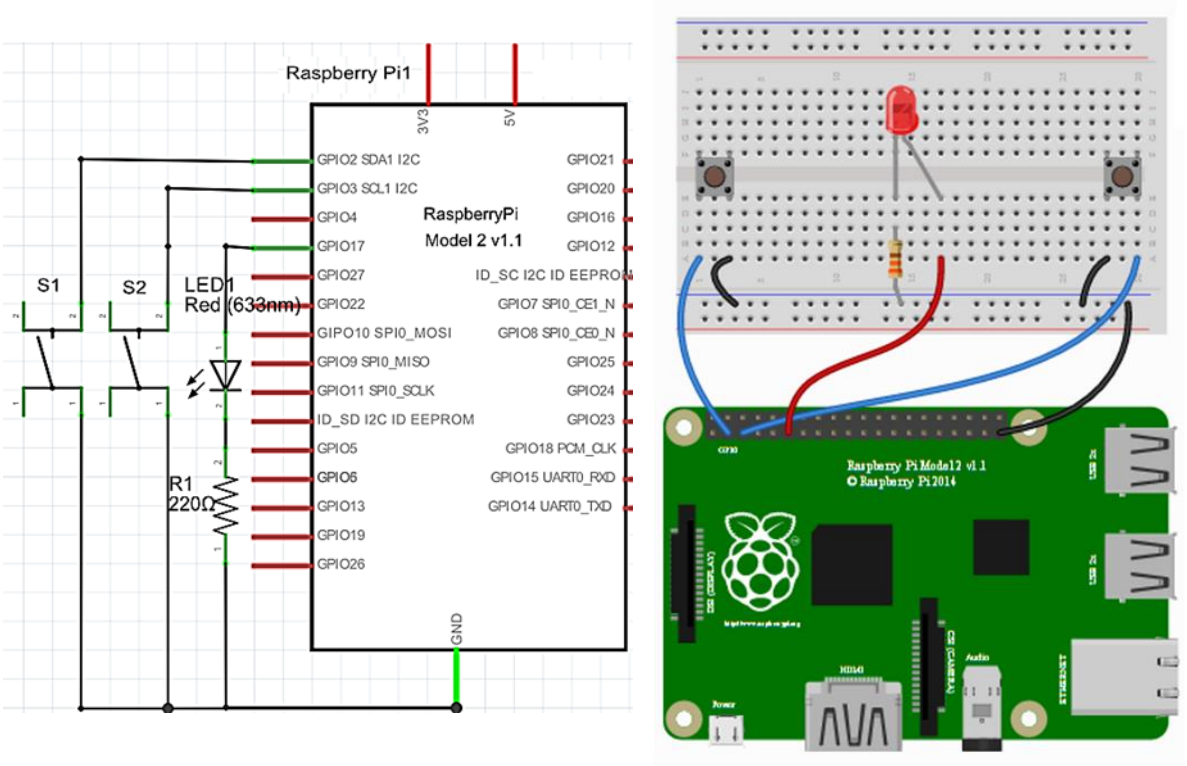
Во оваа вежба за првпат ќе биде употребена инструкцијата uniform од библиотеката random. Оваа инструкција служи за случаен избор (анг. random) на децимален број во опсегот од 0 до 5 и истата е искористена за избор на време (анг. time) во инструкцијата sleep(time). Промената на времето се користи како

елемент на изненадување за двајцата играчи бидејќи нема да се знае кога точно ќе светне диодата.

Исто така, за првпат ќе биде употребен серискиот монитор (анг. Shell) и инструкцијата print() за прикажување на текстуална порака со информација за победникот, прв или втор играч.

4. Опис на електричната шема и начин на поврзување

На слика 4.38. се прикажани функционалната и монтажната шема за реализација на оваа вежба. Поврзувањето е многу едноставно. Тастерите се поставуваат на средината од протоплочката и десните приклучоци се поврзани со лентата за заземјување , а левите приклучоци со GPIO пиновите број 2 и 3 со сини краткоспојници. Приклучоците на лед-диодата се поставуваат во контактни точки кои припаѓаат на две различни проводници од протоплочката и катодата преку отпорникот се поврзува со лентата за заземјување, а анодата директно со GPIO пинот број 17 со црвена краткоспојница. Да не заборавиме лентата за заземјување да ја поврземе со еден од GND пиновите на Raspberry Pi.



Слика 4.38. Поврзување на два тастери и лед-диода со Raspberry Pi 3B+

5. Пишување и внесување на програма за Raspberry Pi 3B+

Ја пишуваме програмата во развојната средина Thonny Python IDE. Да се потсетиме истата е дел од оперативниот систем Raspbian и за нејзино отворање притискаме на опцијата Programming→Tools во менито на самиот оперативен систем.

```
1  from gpiozero import Button,
   LED
```

```
2  from time import sleep
3  import random                    #  Библиотеката random дава можност
                                    #  за избор на случајни вредности.
4  led = LED(17)
5  player_1 = Button(2)             #  Конфигурација на пиновите за
                                    #  поврзување на тастери и истовремено
6  player_2 = Button(3)             #  задавање симболични имиња на
                                    #  двата тастера.
7  time = random.uniform(5, 10)     #  Дефинираме променлива со
                                    #  симболично име time и случајна
                                    #  вредност во опсегот од 5 до 10.
8  sleep(time)                      #  Системот е во мирување за време од
                                    #  5 до10 секунди.
9  led.on()                         #  По истекот на тоа време се вклучува
                                    #  диодата.
10 while True:                      #  Започнува циклус.
11    if player_1.is_pressed:       #  Се прикажува текстуална порака
                                    #  доколку тастерот поврзан со пинот
12       print("Player 1 wins!")    #  број 2 е прв притиснат.
13       Break                      #  Ако е исполнет горниот услов,
                                    #  излегуваме од циклусот.
14    if player_2.is_pressed:       #  Се прикажува текстуална порака
                                    #  доколку тастерот поврзан со пинот
15       print("Player 2 wins!")    #  број 3 е прв притиснат.
16       Break                      #  Излегуваме од циклусот.
17 led.off()                        #  Лед-диодата се исклучува и се
                                    #  враќаме на почетокот на програмата.
```

Со самото пишување на програмата истата се внесува во меморијата на Raspberry Pi.

6. Резултати од реализацијата на вежбата

За да ја провериме исправноста на вежбата треба само да притиснеме Run во летата со алатки на развојната средина и двајцата играчи да ги притиснат тастерите откако ќе светне лед диодата. Во мониторот на развојната средина треба да се појави текстуална порака.Меѓу две притискања на тастерите е воведено време со случајна вредност во опсегот од 5 до 10 секунди. Бидејќи се работи за бесконечен циклус while во второто повторување на циклусот може да се добие поинаков резултат односно поинаква текстуална порака.

Коментар:_____

7. Направи промена!

Во вежбата двајцата играчи можат да го внесат своето име и истото да стои во тесктуалната порака. За таа цел треба да се направи промена во програмскиот код. После програмскиот ред 6 односно после конфигурацијата на пиновите за поврзување на тастерите треба да се додадат следните инструкции.

```
lev_igrac=input (‘Vnesi go imeto’)
desen_igrac=input (‘Vnesi go imeto’)
```

Промена треба да се направи и во инструкциите print() и наместо print("Player 1 wins!") и print("Player 1 wins!") да се напише print("lev_igrac wins!") и print("desen_igrac wins!")
Коментар:_____

4.8.2.5. Практична вежба: Вклучување и исклучување на лед-диода со фотоотпорник

1. **Цел на вежбата**

Оваа практична вежба е слична со вежбата 4.8.1.4. Фотоотпорник за контрола на интензитетот на светлина преку Arduino Uno R3. Но, Raspberry Pi нема вграден аналогно-дигитален претворувач како што има Arduino Uno. Поради тоа не можеме да го менуваме интензитетот на светлината на лед-диодата како што правевме во практичната вежба со Arduino Uno. Наместо да се менува интензитетот на светлината на лед-диода таа само ќе се вклучува и исклучува, во зависност од тоа дали е светло или мрак, дали е ден или ноќ.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем
- протоплочка
- една лед-диода
- еден отпорник со отпорност R=220Ω
- фотоотпорник
- кондензатор со капацитивност 1μF
- жици за поврзување (краткоспојници)

3. **Подготовка за вежбата**

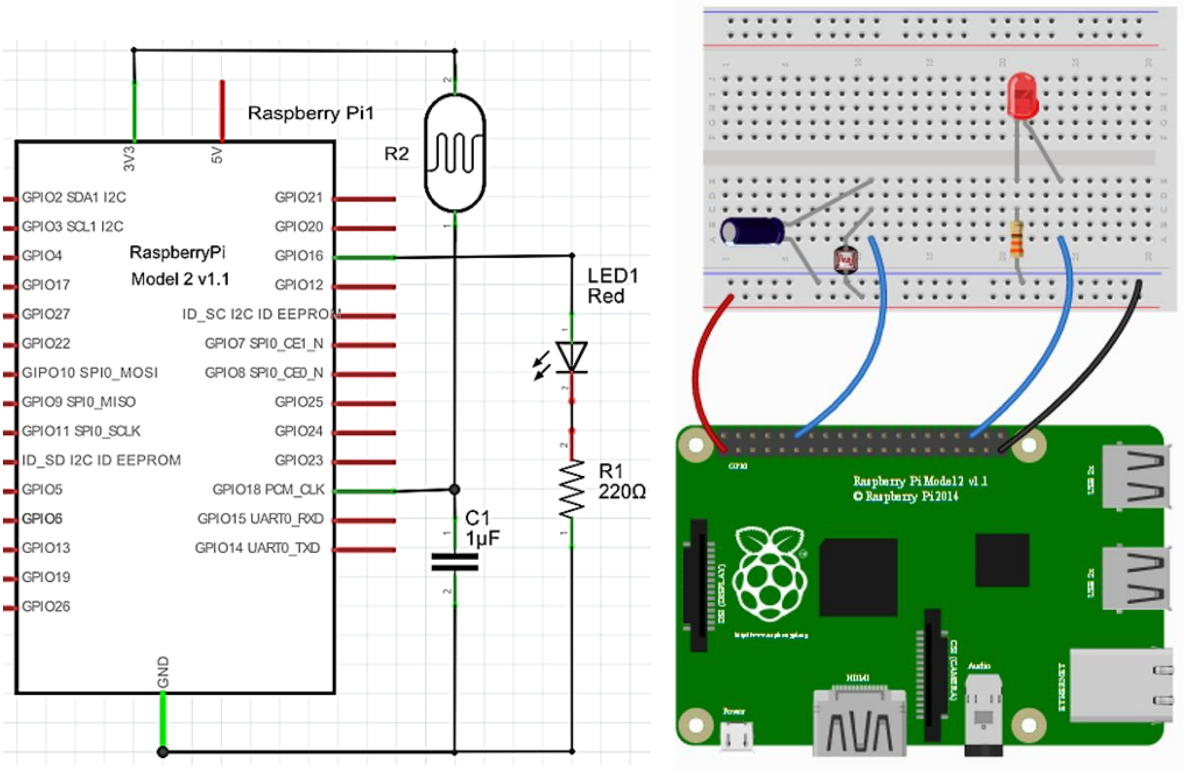
Учениците треба да се потсетат на:

- работа на редно RC коло (наставен предмет Електротехника)
- пин-дијаграм на GPIO подножје и начин на поврзување на лед-диода на протоплочка.
- промената на отпорноста на фотоотпорникот во зависност од интензитетот на светлината и инструкциите од класата LightSensor од

библиотеката gpio. (наставна единица 4.6.4. Оптички сензор или фотоотпорник)

4. Опис на електричната шема и начин на поврзување

Функционалната и монтажна шема за реализација на оваа вежба е дадена на слика 4.39..



Слика 4.39. Поврзување на лед-диода и фотоотпорник со Raspberry Pi 3B+

Во вежбата 4.8.1.4. Фотоотпорник за контрола на интензитетот на светлина фотоотпорникот и отпорникот од 10KΩ формираат напонски делител. Во оваа практична вежба наместо напонски делител ќе користиме редно RC коло составено од еден отпорник и кондензатор, редно поврзани со напојувањето од 3,3V.

Едниот приклучок на фотоотпорникот се поврзува со лентата за напојување, а другиот приклучок се поврзува со GPIO пинот број 18. На истиот GPIO пин се поврзува кондензатор со капацитет од 1 μF, а другиот пин на кондензаторот се поврзува со лентата за заземјување. Ако фотоотпорникот има мала отпорност тогаш напонот на кондензаторот побрзо се менува и обратно. Raspberry Pi 3B+ го мери времето потребно напонот на кондензаторот да се зголеми од нула до вредност 1,34V што претставува напон на праг (види слика 4.16.) и според измереното време детектира мрак или светло.

Поврзувањето на лед-диодата е идентично како во претходните практични вежби со таа разлика што сега е употребен GPIO пинот број 16.

Лентата за напојување ја поврзуваме со пинот 3,3V на Raspberry Pi, а лентата за заземјување со пинот GND.

5. **Пишување и внесување на програма за Raspberry Pi 3B+**

Програмата ја пишуваме во развојната средина Thonny Python IDE. Од библиотеката gpio ја преземаме класата со симболично име LightSensor која содржи инструкции за работа со фотоотпорник. Ќе ги употребиме инструкциите sensor.when_dark и sensor.when_light. Инструкцијата sensor.when_dark се извршува кога Raspberry Pi 3B+ детектира мрак (бавна промена на напонот на кондензаторот), а инструкцијата sensor.when_light се извршува кога Raspberry Pi 3B+ детектира светло (брза промена на напонот на кондензаторот),

```
1  from gpiozero import LightSensor, LED   # Од библиотеката GPIOZERO ги
                                           # повикуваме потребните класи.
2  from signal import pause
3  sensor = LightSensor(18)                # Ги конфигурираме пиновите за
4  led = LED(16)                           # поврзување на фотоотпорникот и лед-
                                           # диодата.
5  sensor.when_dark = led.on               # Лед-диодата се вклучува кога сензорот
                                           # ќе детектира мрак .
6  sensor.when_light = led.off             # Лед-диодата се исклучува кога
                                           # сензорот ќе детектира мрак .
7  pause()                                 # Се внесува време на доцнење.
```

Откако ќе ја напишеме програмата во развојната средина притискаме Run во менито со алатки.

6. **Резултати од реализацијата на вежбата**

Кога фотоотпорникот е изложен на светлина лед-диодата не свети. Кога ќе го покриеме фотоотпорникот со некаков предмет лед-диодата почнува да свети.

Коментар:_____

4.8.2.6. Практична вежба:Проверка на растојание до најблизок објект

1. **Цел на вежбата**

Цел на оваа вежба е примена на сензор за растојание (HC-SR04), негово поврзување со Raspberry Pi 3B+ и пишување на програма која измереното растојание ќе го прикаже во облик на текстуална порака на Shell мониторот во развојната средина Thonny Python IDE.

2. **Потребни компоненти за реализација на оваа вежба се:**

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем
- протоплочка
- сензор за растојание (HC-SR04)
- жици за поврзување (краткоспојници)

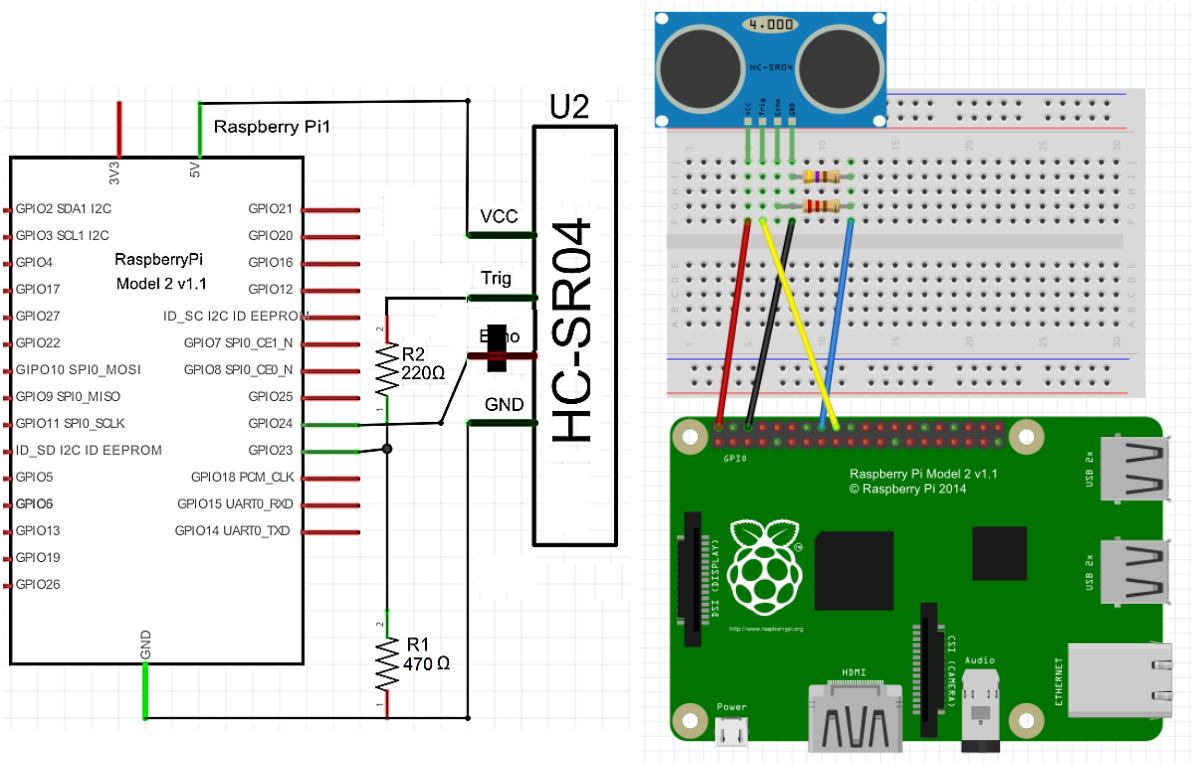
3. Подготовка за вежбата

Учениците треба да се потсетат на:

- наставната единица 2.8.3.2. Упатство за работа со протоплочка
- пин-дијаграм на GPIO подножје (слика 4.15.)
- наставната единица 4.6.3.Сензор за растојание (HC-SR04)

4. Опис на електричната шема и начин на поврзување

Функционалната и монтажна шема за реализација на оваа вежба е дадена на слика 4.40.



Слика 4.40. Поврзување на сензор за растојание со Raspberry Pi 3B+

За поврзување на сензорот за растојание со Raspberry Pi 3B+ потребни се два GPIO пинови, еден за тригерот и еден за ехото.

(1) Приклучокот за заземјување на сензорот го поврзуваме со GND пинот на Raspberry Pi

(2) Приклучокот Trig на сензорот го поврзуваме со GPIO пинот број 24 на Raspberry Pi.

(3) Едниот крај на отпорникот со отпорност 220Ω се поврзува со Echo приклучокот на сензорот

(4) Едниот крај на отпорникот со отпорност 470Ω се поврзува со GND приклучокот.

(5) Другите два краја на двата отпорници заедно се поврзани со GPIO пинот на Raspberry Pi. На ваков начин се формира напонски делите кој го намалува напонот на излез од приклучокот Echo. Овој напон

изнесува 5V, а знаеме дека максималниот влезен напон за Raspberry Pi 3B+ изнесува 3,3V. Во спротивно може да дојде до оштетување на Raspberry Pi.

(6) Приклучокот VCC на сензорот се поврзува со пинот за напојување 5V на Raspberry Pi.

5. Пишување и внесување на програма за Raspberry Pi 3B+

Програмата ја пишуваме во развојната средина Thonny Python IDE по веќе познатата постапка: отворање на нов проект→пишување во уредникот за текст→ сочувување на програмата под симболично име избрано од корисникот. Од библиотеката gpio ја преземаме класата со симболично име DistanceSensor која содржи инструкции за работа токму со сензорот за растојание (HC-SR04).

```
1  from gpiozero import DistanceSensor   # Ја повикуваме класата со инструкции за
                                         # работа со сензорот за растојание
                                         # (HC-SR04).
2  from time import sleep
3
4  sensor = DistanceSensor(23, 24)       # Конфигурација на пинови
5
6  while True:                           # Започнува бесконечен циклус
7      print('Rastojanieto do najbliskiot predmet iznesuva', sensor.distance, 'm')
   #Прикажување на измереното растојание
8      sleep(1)                          # Системот е во мирување 1 секунда.
```

Програмата е многу едноставна и малечка. Тоа се должи пред сè на употребата на библиотеката gpio и класата DistanceSensor. Корисникот не мора да користи инструкции за читање и пишување како што тоа беше случај со Arduino Uno платформата. Со самото повикување на класата DistanceSensor ова автоматски се врши. Сите хардверски карактеристики на сензорот се земени предвид и креиран е софтвер и истиот е содржан во библиотеката gpio односно класата DistanceSensor.

6. Резултати од реализацијата на вежбата

Пред сензорот за растојание поставуваме одреден предмет. Максималното растојание изнесува 1 метар. Откако ќе ја напишеме и сочуваме програмата притискаме Run во менито со алатки и веднаш во полето на серискиот монитор Shell се појавува текстуалната порака со информација за растојанието од сензорот до поставениот објект. Објектот може да се поместува и да ја провериме веродостојноста на добиените податоци.

Коментар:_____

4.8.2.7. Практична вежба: Промена на насока на мотор на еднонасочна струја

1. Цел на вежбата

Оваа вежба е уште еден доказ колку е лесно да се програмира Raspberry Pi 3B+ благодарение на библиотеките и нивното инструкциско множество, поради кои програмирањето станува многу едноставно и разбирливо. Целта на оваа вежба е да се менува насоката на вртење на DC-моторот. Да се потсетиме моторот за еднонасочна струја не се поврзува директно со Raspberry Pi туку се користи H-мост како интерфејс-компонента.

2. Потребни компоненти за реализација на оваа вежба се:

- Raspberry Pi 3B+ систем со инсталиран Raspbian оперативен систем
- протоплочка
- мотор на еднонасочна струја со напојување 3V
- интегрирано коло SN754410
- жици за поврзување (краткоспојници)

3. Подготовка за вежбата

Учениците треба да се потсетат на:

- пин-дијаграм на GPIO подножје на Raspberry Pi 3B+ (слика 4.15.)
- наставната единица 4.7.4. Мотор на еднонасочна струја
- пин-дијаграмот на интегрираното коло SN754410 (слика 4.25.)

4. Опис на електричната шема и начин на поврзување

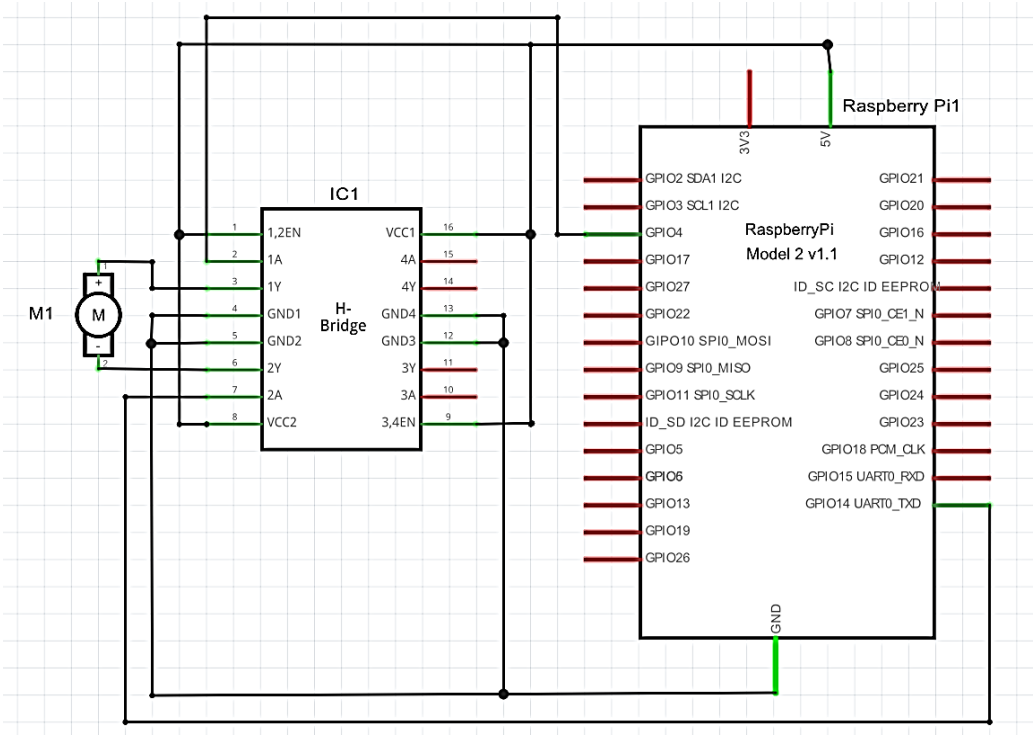
На слика 4.41. и 4.42. е прикажана функционалната и монтажната шема за реализација на оваа вежба.

(1) Интегрираното коло SN754410 го поставуваме на средината на протоплочката внимавајќи да не ги извиткаме неговите пинови. Со црвени краткоспојници ги поврзуваме пиновите за напојување и оспособување (анг. Vcc, Enable) со лентата за напојување, а со црни краткоспојници ги поврзуваме четирите GND пинови со лентата за заземјување.

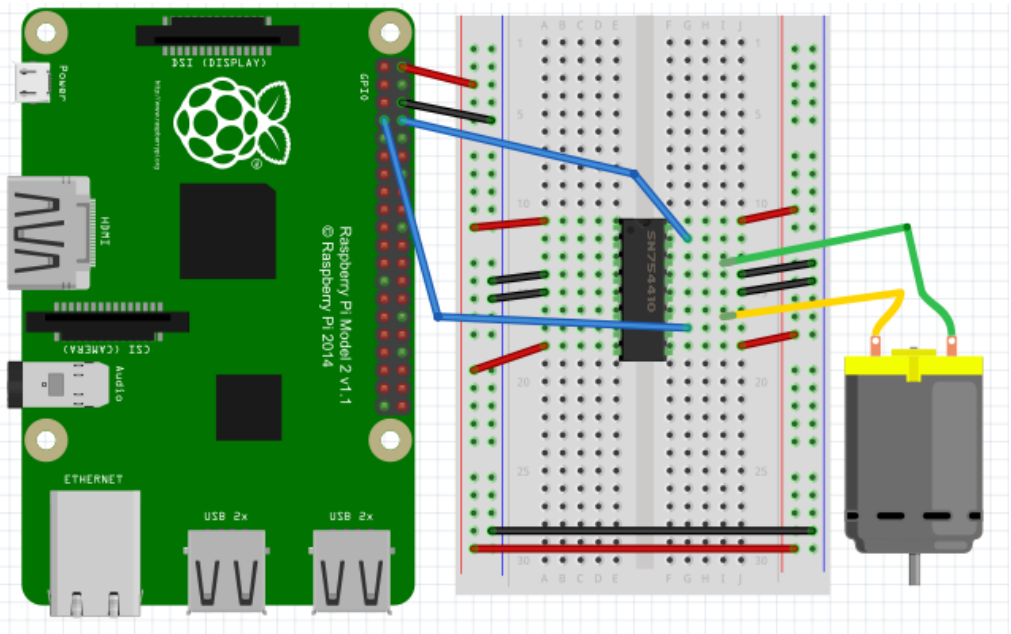
(2) Преку протоплочката, со сините краткоспојници ги поврзуваме GPIO пиновите број 4 и 14 со влезните пинови (анг. In1,In2) на интегрирано коло SN754410.

(3) Преку протоплочката, со зелената и жолтата краткоспојница ги поврзуваме приклучоците на моторот за еднонасочна струја со излезните пинови (анг. Out1, Out2) на интегрирано коло SN754410.

(4) На крај, горната и долната лента за напојување ја поврзуваме со пинот 5V на Raspberry Pi, лентата за напојување со GND пинот.



Слика 4.41. Функционална шема за поврзување на мотор на еднонасочна струја со Raspberry Pi 3B+



Слика 4.42. Монтажна шема за поврзување на мотор на еднонасочна струја со Raspberry Pi 3B+

5. Пишување и внесување на програма за Raspberry Pi 3B+

Ја пишуваме програмата во развојната средина Thonny Python IDE во состав на Raspbian оперативниот систем. Ја повикуваме класата инструкции Motor од библиотеката gpio. Ги користиме инструкциите motor.forward() и motor.backward() до кои веќе се запознавме.

```
1  from gpiozero import Motor   #  Го внесуваме делот од библиотеката gpio
                                #  за работа со мотор.
2  from time import sleep
3  motor = Motor(forward=4, backward=14)
4  #4 и 14 се броеви на GPIO-пиновите.
5  while True:                  #  Започнува бесконечен циклус.
6     motor.forward()           #  Моторот врти во насока на стрелката #на
                                #  часовникот. Бидејќи малата заграда е
                                #  празна, тоа значи дека моторот ќе се врти
                                #  со максимална брзина. Ако во малата
                                #  заграда беше бројот 0,5 тоа ќе значеше
                                #  дека моторот ќе врти со половина од
                                #  својата максимална брзина.
7     sleep(5)                  #  Моторот врти 5 секунди во една насока.
8     motor.backward()          #  Се менува насоката на вртење.
9     sleep(5)                  #  Моторот врти пет секунди во насока
                                #  обратна од стрелката на часовникот.
```

6. **Резултати од реализацијата на вежбата**

Резултатот од реализацијата на оваа практична вежба ќе го видиме ако притиснеме Run во менито со алатки на развојната средина Thonny Python IDE. Очекувани резултати се моторот да почне да врти со промена на својата насока на секои 5 секунди.

Коментар:_____

Заклучоци

Основни градбени елементи на програмскиот јазик C/C++ се : променливите, инструкциите и структурите.

Синтакса во програмирање значи множество на правила за подредување на ознаките, симболичните имиња, операторите, интерпункциските знаци, коментарите, со цел да се добие исказ разбирлив за компјутерот.

Променливите ги чуваат податоците кои се обработуваат во компјутерот. Освен името и видот на податок, променливите имаат и свои вредности. Кога декларираме (најавуваме) променливи за Arduino платформа прво се пишува видот на променлива, па симболичното име, операторот за доделување и вредноста на променливата..

Основните видови на податоци се: цели броеви, децимални броеви, карактери, низи и логички податоци.

Операторите се знаци со кои програмерот според точно определени правила гради искази, инструкции. Постојат повеќе видови оператори: математички, логички, споредбени, оператори за доделување.

Инструкциите за влезно- излезни пинови служат за конфигурирање на пиновите (влез или излез) и за запишување или читање на нивните вредности. Во оваа група спаѓаат инструкциите: digitalRead(pin), digitalWrite(pin,vrednost), pinMode(pin,mode), analogRead(pin) и analogWrite(pin,value).

Аналогно-дигиталниот конвертор, во составот на Arduino Uno R3, ги претвора аналогните вредности во цели броеви од 0 до 1023. Целиот број потоа се претставува како бинарен код од 10 битови.

Инструкцијата delay (ms) се користи за внесување на време на доцнење изразено во милисекунди.

Инструкцијата map(vrednost, fromLow,fromHihg,toLow, toHigh) го менува опсегот на вредности на променливата. fromLow и fromHihg се минималната и максималната вредност на стариот опсег, а toLow и toHigh се минималната и максималната вредност на новиот опсег.

Инструкцијата bitClear (x,n) го ресетира (поставува на нула) битот со реден број n во променливата x. Инструкцијата bitSet (x,n) го сетира (поставува на високо ниво) битот со реден број n во променливата x. .

Со инструкцијата Serial.begin(brzina) се поставува пропусниот опсег односно брзината на сериски пренос на податоци чија единица мерка е број на битови во една секунда. Со инструкцијата Serial.print (x) на екранот од серискиот монитор се печати вредноста на променливата.

Структурите за избор на можности се составени од исказите:if, if...else и else. If исказот го проверува условот и ако истиот е исполнет (вистинит) тогаш ги извршува инструкциите.

Библиотеките се потпрограми кои овозможуваат комуникација меѓу периферните уреди и Arduino платформата. Arduino развојната средина содржи неколку стандардни библиотеки и истите може да ги видиме со притискање (click) на Sketch > Import Library во самото мени.

Опсегот на вредности кои може да ги даде сензорот е многу поголема од опсегот на вредности кои може да се измерат во реалната околина. Поради тоа треба да се нагодат максималната и минималната вредност на сензорот уште во првите неколку секунди по пуштањето во работа на сензорот. Оваа постапка се вика калибрација.

Кодовите во програмскиот јазик Python се пократки зошто не постои декларирање на променливи. Кога пишуваме блок на инструкции во рамките на една структура, како што се if….else или while, не се користат големи загради туку инструкциите се вовлекуваат за четири празни места навнатре.

Инструкции за работа со тастер се: Button(), wait_for_press (),wait_for_release(), held_time, hold_time, is_held и is_pressed.

Инструкции за работа со лед-диода се: LED(),blink(on_time=1, off_time=1), off(), on() и toggle().

Прашања за повторување

1. Кои се основните градбени елементи на програмскиот јазик C++?
2. Што подразбираме под поимот синтакса во програмирањето?
3. Набројте ги основните видови на податоци што се користат во програмскиот јазик C++?
4. Што претставува име, вредност и вид на променливата
   bool isCodingFun = true?
5. Кои се ознаките за текстуален податок и за низа?
6. Колку вредности можат да имаат логичките податоци?
7. Што претставуваат операторите во програмскиот јазик C++?
8. Кој е оператор за доделување на вредност на променлива?
9. Каков резултат ќе се добие по извршувањето на инструкцијата x=13%5;?
10. Напишете го скратениот исказ за аритметичката инструкција x=x-1?
11. Кој е условот за извршување на блокот инструкции под структурата

if (x!=y) {…} ?

12. Кои се параметри на инструкцијата digitalWrite () ?

13. За што служи аналогно-дигиталниот претворувач во составот на Arduino Uno развојната платформа?

Искоментирајте го следниот код:

```
void loop() {
   val = analogRead(analogPin);
   analogWrite(ledPin, val / 4);
 }
```

14. За што служи инструкцијата tone (pin, frequency) ?

15. Со која инструкција се мери времето од почетокот на извршувањето на програмата во Arduino Uno развојната платформа?

16. Објаснете ја инструкцијата constrain (x, a, b)?

17. Која инструкција се користи за ресетирање на одреден бит во дадена променлива?

18. Кои се задолжителни структури во програмите за микрокрокомпјутерот Arduino Uno и која е нивната функција?

19. Што се запишува во малата и во средната заграда на структурата if () {…}?

20. Именувајте ја структурата за избор на можности!

21. Која е разликата меѓу структурите while…do и do….while?

Колкупати ќе се повтори циклусот во следниот код:

```
int x = 0;
do {
  delay(50);
  x = readSensors();
} while (x < 100);
```

22. Кои три искази се запишуваат во малата загради на структурата for (…) {…}?

23. Што претставува секој параметар во инструкцијата

LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)?

24. Искоментирајте ја инструкцијата servo.attach (pin, min, max); ?

Искоментирајте го следниот код:

```
void loop() {
   int val = analogRead(0);
   val = map(val, 0, 1023, 0, 255);
   analogWrite(9, val);
 }
```

25. Кои се предностите и недостатоците на програмскиот јазик Python?

26. По што се разликуваат системските програми компајлер и интерпретер?

27. Која инструкција од програмскиот јазик Python се користи за креирање нови функции?

28. Која библиотека од програмскиот јазик Python ја користиме за работа со влезно-излезни уреди?

29. Како се обележани GPIO-пиновите со физички броеви 1, 2, 3 и 5? Објаснете ја нивната функција!

30. Објаснете по што се разликуваат следниве инструкции: import gpiozero, from gpiozero import Button, from gpiozero import ButtonBoard!

31. Набројте неколку аналогни влезни уреди за микрокомпјутер Raspberry Pi!

32. Објаснете како микрокомпјутерот Raspberry Pi одлучува дали некој аналоген влезен уред е активен или не!

33. Направи споредба меѓу програмските јазици C/C++ и Python? Кој од нив располага со поголем број на библиотеки и која е добивката од нивната употреба?

34. Кои параметри се карактеристични за инструкцијата Butoon()?

35. Каде се сретнува и за што служи инструкцијата wait_to_release?

36. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import Button
from signal import pause
def say_hello():
    print("Hello!")
```

```
button = Button(2)
 button.when_pressed = say_hello
pause()
```

37. Зошто е потребен кондензатор при поврзување на фотоотпорник со Raspberry Pi 3B+?

38. За што служат двата податочни приклучоци кај сензорот за растојание?

39. Која библиотека и класа се користи за уредот да се постави во режим на мирување?

40. Набројте неколку излезни уреди што се поврзани со Raspberry Pi преку USB-приклучок?

41. Во која класа се користи и каков резултат дава инструкцијата distance?

42. Која инструкција се користи за промена на состојбата на лед-диодата?

43. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import LED
from signal import pause
red = LED(17)
red.blink()
pause()
```

44. Како ширината на ширинско модулираните импулси влијае врз интензитетот на светлината на PWMLED-диодата?

45. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import PWMLED
from time import sleep
led = PWMLED(17)
while True:
    led.value = 0
    sleep(1)
    led.value = 0.5
    sleep(1)
    led.value = 1
```

```
    sleep(1)
```

46. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import LEDBoard
from signal import pause
leds = LEDBoard(5, 6, 13, 19, 26, pwm=True)
leds.value = (0.2, 0.4, 0.6, 0.8, 1.0)
pause()
```

47. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import TrafficLights
from time import sleep
from signal import pause
lights = TrafficLights(2, 3, 4)
def traffic_light_sequence():
    while True:
        yield (0, 0, 1)
          sleep(10)
        yield (0, 1, 0)
        sleep(1)
        yield (1, 0, 0)
        sleep(10)
        yield (1, 1, 0)
        sleep(1)
lights.source = traffic_light_sequence()
pause()
```

48. Кои се трите основни бои и какви вредности треба да имаат тие за да се добие бела и црна боја?

49. Со која инструкција може да се менува брзината на вртење на DC-мотор?

50. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import MotionSensor, LED
from signal import pause
pir = MotionSensor(4)
led = LED(16)
pir.when_motion = led.on
pir.when_no_motion = led.off
pause()
```

51. Кој параметар ја одредува брзината на вртење на DC при негово поврзување со УRaspberry Pi?

52. За што служи инструкцијата value() при работа со серво-мотор?

53. Искоментирајте ги инструкциите и функцијата на кодот!

```
from gpiozero import Servo
from time import sleep
servo = Servo(17)
while True:
    servo.min()
    sleep(2)
    servo.mid()
    sleep(2)
    servo.max()
    sleep(2)
```

Прилог: Симулатори за Raspberry

Во наставната единица 2.8.3.3. Компјутерска симулација за Arduino Uno R3 се запознавме со извонредните можности кои ги нуди платформата Tinkercad. За жал оваа платформа не поддржува симулација на Raspberry Pi. Бесплатен симулатор за Raspberry Pi во Python програмскиот јазик во моментов не постои. Сепак како прилог на последната модуларна единица, накратко ќе се запознаеме со четири симулатори за Raspberry кои се достапни сега.

Доста популарен е симулаторот на Microsoft компанијата во рамките на Azure IoT платформата. Azure е облак технологија која нуди софтверска и хардверска поддршка за дигитални трансформации. Во првата тема истакнавме дека Интернет на нештата е платформа која нуди вмрежување на сензори. Корисникот не мора да се грижи за изборот на комуникациски протоколи и нивно нагодување бидејќи тоа го врши самата платформа. Таа е замислена да биде од отворен тип односно нејзините корисници да можат да споделуваат програмски кодови. Да напомнеме дека овој симулатор е во изработка и се користи пробна верзија. За жал истата не е бесплатна иако Microsoft дозволува бесплатна употреба за едукативни цели во времетраење од една година. Корисникот мора да има свој профил на Azure.

Wyliodrin STUDIO е бесплатен симулатор за Raspberry Pi. Апликацијата може да се преземе од официјалната веб-страница со следниот линк https://wyliodrin.studio/, но постои и веб-верзија за што не е потребна инсталација туку само најава со корисничкиот профил на Google. Слично како Azure ѝ Wyliodrin платформата нуди вмрежување на сензори. При креирањето на проект потребно е да се состави електричната шема во апликацијата Fritzing, а потоа сочуваните документи ги презема самата Wyliodrin платформа. И покрај големиот број предности овој симулатор не е погоден за наставните цели на овој учебник бидејќи за програмирање на Raspberry Pi се користи програмскиот јазик JavaScript, а не Python.

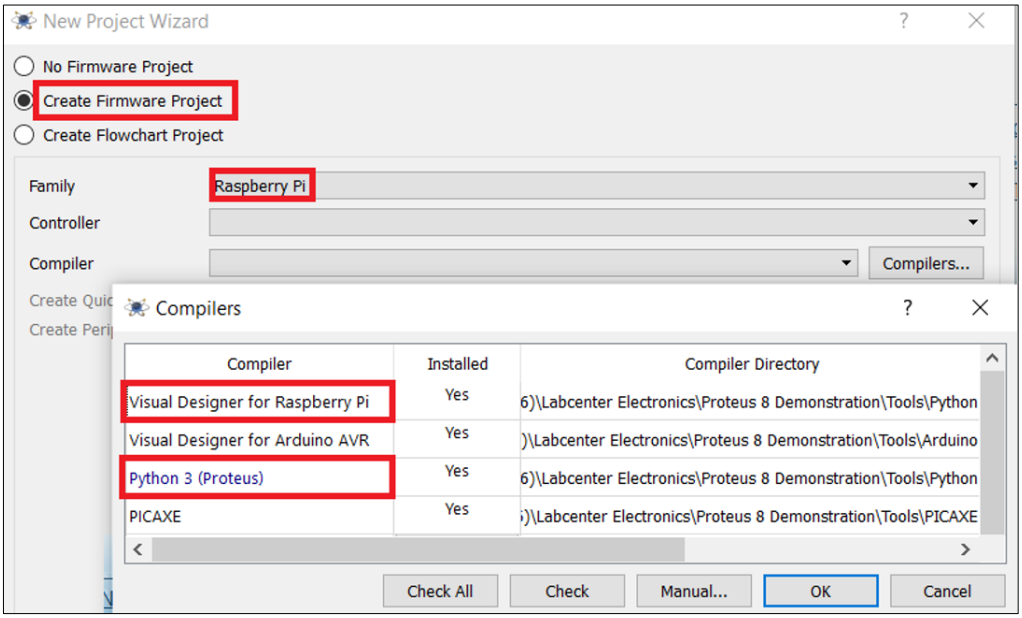
Wokwi е уште еден бесплатен симулатор, но истиот не го поддржува Raspberry Pi 3B+ туку Raspberry Pico. Тој не е микрокомпјутер туку микроконтролер, без оперативен систем, и не можат да се користат инструкциите од библиотеката GPIO.

Proteus е веќе добро позната апликација за симулација на електрични кола, која датира уште од 1989 година. Поновите верзии на оваа апликација поддржуваат симулација на Arduino и Raspberry Pi. За програмирање на Raspberry Pi може да се користи програмскиот јазик Python 3, но најчесто се користи Visual Designer едиторот за составување на програмски блок-дијаграми..

Овој симулатор не е бесплатен. Доколку сакаме да се запознаеме со неговите можности може да ја преземеме бесплатната пробна верзија на симулаторот од официјалната страна на Proteus на линкот https://www.labcenter.com/simulation/. Пробната верзија нема временско

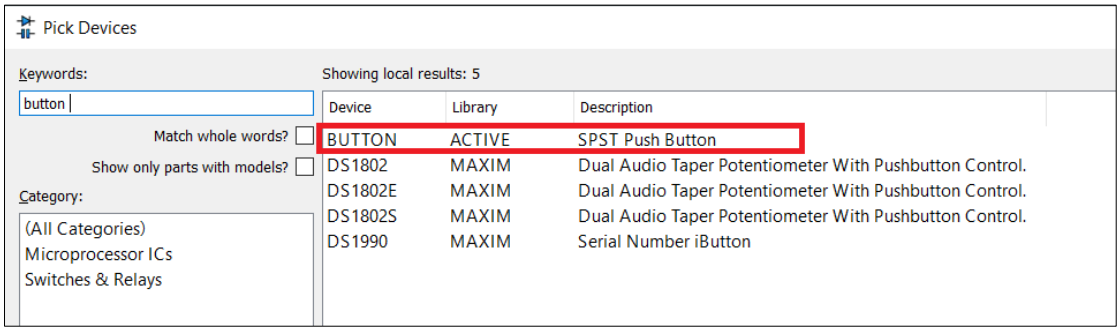
ограничување, но не дозволува да се сочуваат новите проекти и за програмирање на Raspberry Pi го дозволува само Visual Designer едиторот. Накратко ќе се запознаеме со постапката за креирање на проект.

Најнапред треба да се отвори нов проект (анг. New Project), истиот да се именува и да се сочува во избраната датотека. Потоа се отвораат неколку прозорци при што избираме вообичаен модел на дизајн (анг. Create a schematic design from the selected template-DEFAULT), НЕ избираме дизајн за електрична плочка (анг. Do not create a PCB layout) и избираме софтвер за вградливи системи (анг. Create Firmware Project), после што го избираме контролерот и видот на компајлер. Ова е прикажано на слика 4.43.



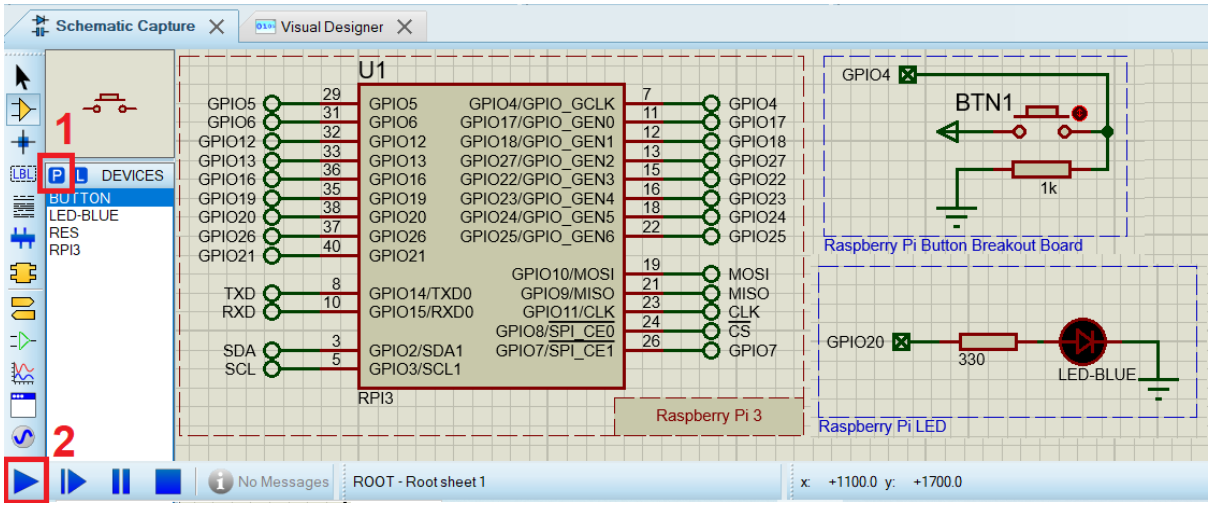
Слика 4.43. Избор на компајлер за Raspberry Pi 3B+

Со притискање на копчето Finish во следниот прозорец се отвора работната површина во чиј состав има два прозорци, прозорец за составување на електрично коло (анг. Schematic Capture) и прозорец за составување на програма (анг. VSM Studio или Source Code). Во горниот лев агол од прозорецот за составување на електрично коло се наоѓа копче означено со буквата P (анг. Pick Devices) (слика 4.45. копче број 1) и со негово притискање се отвора нов прозорец за избор на електрични компоненти. Истите ги наоѓаме со пребарување по клучен збор (слика 4.44).



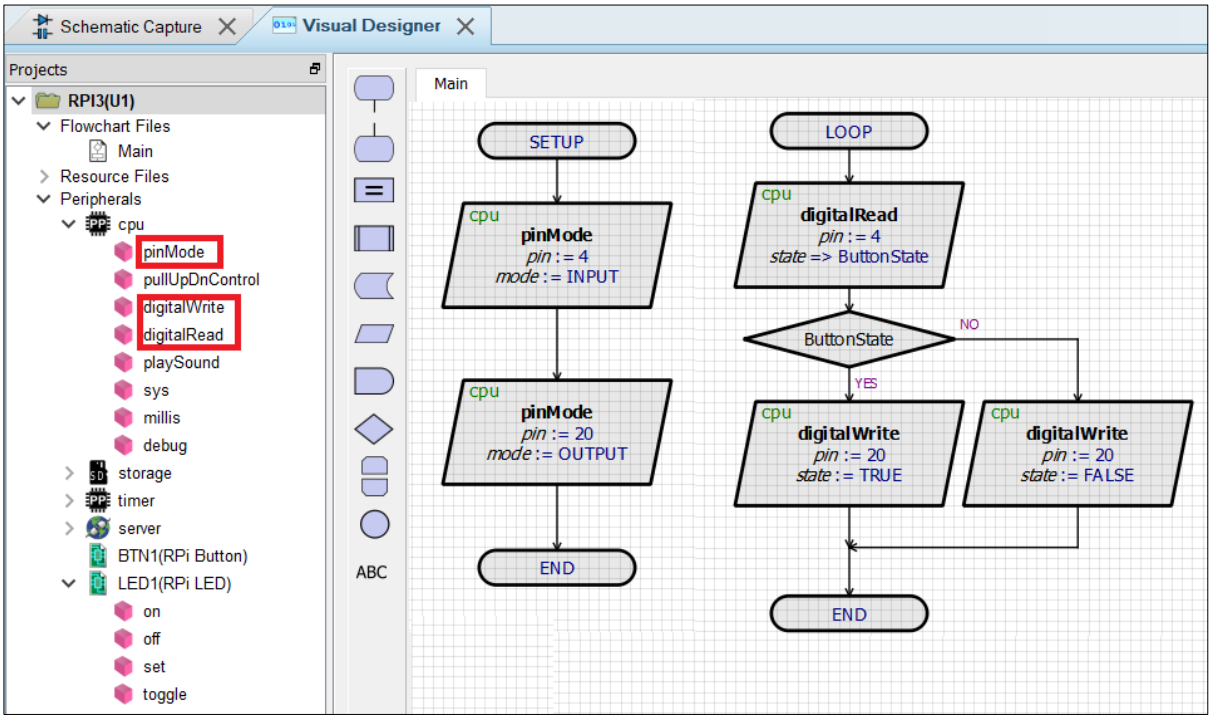
Слика 4.44. Избор на хардверски компоненти за електрично коло

На слика 4.45. е прикажано електрично коло составено од тастер, отпорник, лед-диода и Raspberry Pi 3B+. Во симулаторот компонентите со GPIO пиновите на Raspberry Pi 3B+ не се поврзуваат со водови туку со исто именување на истите.



Слика 4.45. Изглед на работна површина за составување на електрично коло

Откако ќе го составиме електричното коло, преминување кон составување на програмата со Visual Designer едиторот или во Python програмскиот јазик. На слика 4.46. е прикажан програмскиот блок-дијаграм за горното електрично коло составен во Visual Designer едиторот. Програмирањето е многу едноставно.



Слика 4.46. Изглед на работна површина за Visual Designer едиторот

Во посебен прозорец од левата страна на работната површина, со назив Project, наведени се сите употребени хардверски компоненти и со нивно

притискање се отвораат паѓачки менија со сите инструкции кои се однесуваат на конкретната компонента. Со нивна селекција и методот „Повлечи и пушти“ блоковите ги внесуваме во дијаграмот и составуваме програма. Со притискање на копчето „Run“ (слика 4.45. копче број 2) започнува симулацијата.

Употребата на симулатори може да го олесни учењето и да ги збогати наставните содржини. Симулаторите дозволуваат лесна промена и експериментирање со дизајнот на електричните кола, менување на програмските кодови и откривање на евентуалните грешки и што е најважно за сè ова не е потребна набавка на хардверски компоненти.

Користена литература:

[1] M. Rafiquzzaman, "INTRODUCTION TO MICROPROCESSORS AND MICROCOMPUTER-BASED APPLICATIONS," in *Microprocessors and Microcomputer-Based System Design, 2nd Edition*, CRC Press, 2021.

[2] [Online]. Available: https://www.arduino.cc/en/hardware.

[3] G. Halfacree, Raspberry Pi Beginner's Guide How to use your new computer, Cambridge, CB1 2JH: Raspberry Pi Trading Ltd, 2018.

[4] И. Ј. Ј. Д. Љубица Маркудова, Дигитална електроника и микропроцесори-III година, Министерство за образование и наука, 2010.

[5] M. S. Scott Fitzgerald, The Arduino Project Book, Torino, Italy, 2012.

[6] B. J. N. W. Michael Margolis, Arduino Cookbook, 3rd Edition, O'Reilly Media, 2020.

[7] S. Monk, Raspberry Pi Cookbook, 3rd Edition, O'Reilly Media, 2019.

[8] [Online]. Available: https://www.alza.co.uk/how-to-build-your-own-PC.

[9] D. Wilcher, Basic Arduino Projects, Make:Community, 2014.

[10] Г. Силбершац, Концепти на оперативните системи, 2016: Арс Ламина.

[11] A. Warren, Writer, *Windows 10 Exam 70-698: Installing and Configuring Windows 10 LiveLessons.* [Performance]. Pearson IT Certification, 2018.

[12] "Arduino Reference," [Online]. Available: https://www.arduino.cc/reference/en/.

[13] "gpiozero-GPIO Zero 1.6.2. Documentation," [Online]. Available: https://gpiozero.readthedocs.io/en/stable/.