УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ

Катарина Андонов

Програмирање микробит уређаја у Мејк Коду и Микропајтону

МАСТЕР РАД

Београд, 2022.

Ментор:

др Мирослав Марић, редовни професор

Универзитет у Београду, Математички факултет

Чланови комисије:

др Александар Картељ, доцент

Универзитет у Београду, Математички факултет

др Стефан Мишковић, доцент

Универзитет у Београду, Математички факултет

Датум одбране: 14.07.2022.

Резиме: Информатика и рачунарство имају све већи значај у образовању ученика од најранијих узраста. Све више институција и организација учествује помагању школама и наставном особљу да на што креативнији и иновативнији начин приближе ученицима појам и суштину програмирања као веома значајног дела њиховог даљег образовања и стицања информатичке писмености. Програмирање микробит уређаја (енгл. micro bit) у Мејк Коду (енгл. MakeCode) и Микропајтону (енгл. MicroPython) је део светског пројекта „Школе за 21. век“ који има за циљ да подстакне нове генерације да користе нове технологије на забаван и креативан начин. Коришћењем технологија на овај начин ученици основних школа се подстичу на критичко мишљење и заједничко решавање проблема.

Специфични циљ овог рада је упознавање са карактеристикама и програмирањем микробит уређаја. Микробит уређај је нови едукативни уређај у настави информатике и рачунарства у основним школама, који подстиче развој логичког мишљења и вештине решавања проблема. Циљ овог рада јесте да на што бољи начин, прикаже разноврсне могућности микробита и његову применљивост у решавању проблема из свакодневног живота. Основни појмови програмирања микробит уређаја ће бити уведени коришћењем блоковског окружења Мејк Код, као и програмског окружења Микропајтон. У лекцијама ће бити обрађени основни концепти програмирања као што су: одлучивање у програмима, понављање, низови и креирање и употреба функција. Након тога биће представљени примери који приказују да је коришћењем микробит уређаја могуће направити сопствени уређај на којем се приказују реални подаци који се могу користити за анализу и решавање реалних проблема.

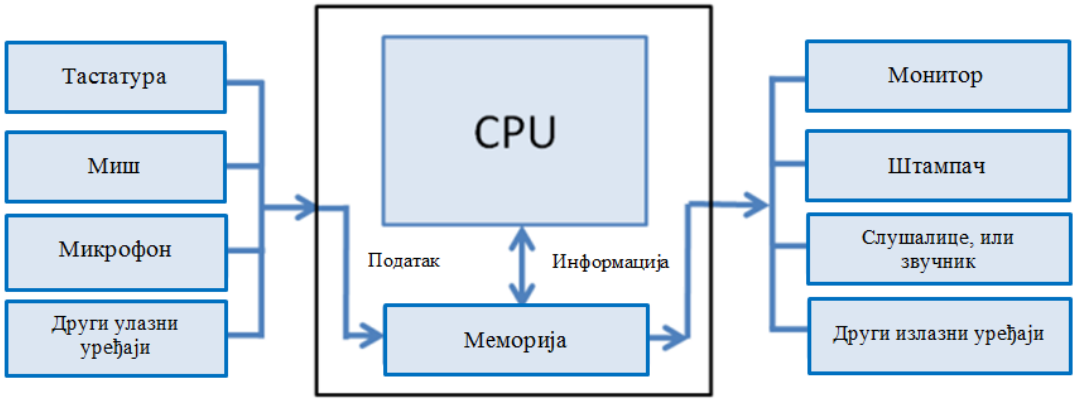
Кључне речи: микробит, програмирање, критичко мишљење, решавање проблема

Садржај

1 Увод

У свету технологије много пажње се посвећује рачунарима са једном плочом (енгл. single-board computers — SBC). Први међу њима је 2012. године развила фондација Raspberry Pi, по којој је мини рачунар и добио назив. До сада, неколико генерација ових уређаја је изашло на тржиште. Сваки од модела има *Broadcom* чип који садржи *APM* процесор, док је количина рам меморија у интервалу од 256MB до 1GB. У зависности од модела, цена ових уређаја се креће од неколико долара до неколико стотина долара. Приликом развијања оваквих уређаја идеја компанија је да ови уређаји могу да извршавају једноставније задатке, али и да буду повезани са Интернет стварима (енгл. Internet of Things, скраћено IOT). Термин Интернет ствари је увео британски инжењер Кевин Ештон 1999. године. Овај појам се користи да опише повезивање паметних физичких уређаја са електроником, софтвером или сензорима. У почетку, процесори су били превелики и користили су се само за сервере, рачунаре и лаптопове. Међутим, фирме које су се бавиле производњом процесора су покушавале да смање чипове како би могли да их ставе и у мобилне телефоне. Експанзија развоја малих чипова, довела је до тога да компаније увиде да је могуће користити ове чипове за повезивање готово свега путем мреже. Тако је настао термин Интернет ствари. Програмери и технолошки стручњаци широм света деле мишљење да ће овај термин променити свет и олакшати живот. Компаније Гугл и Амазон су већ развиле дигиталне кућне помоћнике, док се интернет конекција појављује и у паметним фрижидерима и веш машинама. Главни циљ развијања овог концепта јесте да ствари, односно физички уређаји могу на дневном нивоу да „комуницирају“ са људима и да им на тај начин олакшају дневне обавезе. У ову групу спада и нови уређај: Би-Би-Си Микробит (енгл. BBC Micro:bit). Микробит је ручни, програмибилни микро-рачунар развијен као део иницијативе „Make it Digital“ из 2015. године, коју је покренуо Би-Би-Си. Креиран је од стране Би-Би-Сија, компаније Мајкрософт, као и многих других компанија у циљу промовисања програмирања међу ученицима од 10 до 15 година. Микробит уређај је развијен са циљем да инспирише ученике да постану креативни и да на занимљив начин науче основне концепте програмирања. Може се користити за различите интересантне пројекте, од музичких инструмената до робота. Програмирање микробит уређаја могуће је на рачунарима, али и на паметним мобилним телефонима и таблетима. Основна идеја развијања микробит уређаја јесте да покаже ученицима да је програмирање забавно, али и то да је повезано са другим предметима.

Са обзиром да је микробит микро-рачунар, потребно је подсетити се основних компонента рачунара које су приказане на слици 1.1. Рачунар се састоји из четири главне компоненте: процесор (енгл. Central Processing Unit — CPU), меморија, улазни и излазни уређаји.



Слика 1.1: Главне компоненте рачунара

Процесор (енгл. Central Processing Unit — CPU) јесте мали чип у рачунару који обрађује и трансформише податке. Процесор се може сматрати као мозак рачунара.

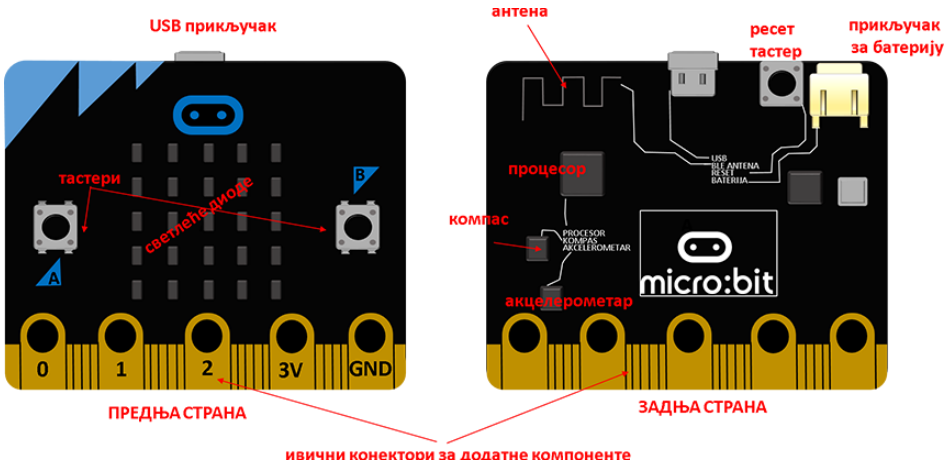
Меморија је уређај у рачунару у коме се чувају подаци. Како се подаци могу чувати тренутно или трајно, постоје две врсте меморије: рам меморија и хард диск. *RAM* (Random Access Memory) је краткотрајна меморија рачунара, док је хард диск дугорочна меморија рачунара у којој се могу чувати информације и када је рачунар искључен.

Улазни уређаји помажу рачунару да добија информације. Улазни уређаји могу бити: миш, тастатура, камера, микрофон, скенер итд. Излазни уређаји помажу да се информације преносе или прикажу. Излазни уређаји могу бити: монитор, штампач, слушалице, звучници итд.

1.1 О микробит уређају

Микробит је микро-рачунар који се састоји од 32 битног ARM процесора, тастера и екрана. Осим тога, микробит се састоји из многобројних сензора који нам омогућавају различите примене овог микро-рачунара. Микробит се напаја са две AAA батерије. Физичке компоненте микробит уређаја (слика 1.1.1) су:

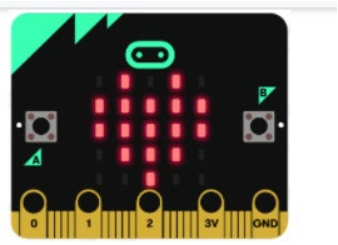
- светлеће диоде;
- тастери A и B који се могу програмирати и тастер за ресетовање;
- пинови тј. ивични конектори за додатне компоненте;
- сензори за температуру, светло, сензори покрета;
- тастер за ресетовање уређаја;
- УСБ прикључак.



Слика 1.1.1: Физичке компоненте микробит уређаја

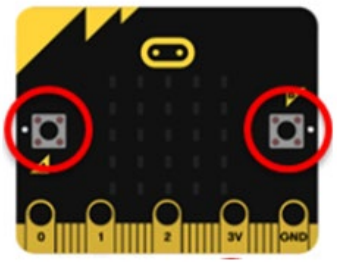
Предња страна табле микробит уређаја дизајнирана је за интеракцију са корисником и састоји се од следећих компоненти: екран, тастери, пинови и лого.

Екран микробит уређаја састоји се од 25 светлећих лед лампица које су поређане у пет колона и пет врста. Лед је скраћеница од Light Emitting Diode, што значи светлећа диода тј. диода која емитује светлост.



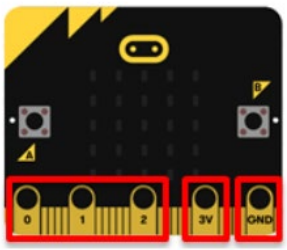
Слика 1.1.2: Екран микробит уређаја

Са леве и десне стране екрана уређаја Микробит, налази се по један тастер: тастер А и тастер Б. Микробит може да региструје који тастер је притиснут. Такође, тастери су програмибилни и омогућавају кориснику интеракцију са њима. Микробит може и да реагује на притисак неког од тастера, тако што ће извршити одређену радњу, у зависности од тога на који начин ће бити програмиран.



Слика 1.1.3: Тастери микробит уређаја

Микробит садржи и 25 пинова – пет великих и двадесет малих, помоћу којих је на микробит могуће повезати разне додатне сензоре, али и управљати различитим уређајима. Велики пинови су означени са: 0, 1, 2, 3V и GND. Пинови 0, 1 и 2 су улазно-излазни пинови, док пинови 3V и GND служе за напајање.



Слика 1.1.4: Пинови микробит уређаја

Лого микробит уређаја је осетљив на додир и може бити искоришћен као улазни уређај. Лого препознаје додир уочавајући мале промене у електричном пољу – слично као мобилни телефон или таблет.

Задњи део табле микробит уређаја састоји се од хардвера и електронских компоненти. Најважније компоненте које се могу наћи на задњем делу табле су: акцелерометар, компас, сензор за температуру, компонента за радио, блутут антена, УСБ улаз, тастер за ресетовање програма, лампица која сигнализира да је микробит повезан са рачунаром и прикључак за спољну батерију.

Уграђени акцелерометар је један од многих корисних сензора микробит уређаја, који свакако може допринети да програми постану занимљивији. Акцелерометар служи за препознавање различитих типова покрета: нагиб, дрхтај или слободан пад. Уз помоћ компаса,

који је уграђен у микробит, могуће је одредити јачину магнетног поља у близини, али и тренутну орјентацију уређаја у односу на стране света. Овај компас мора бити калибрисан први пут када се користи, како би очитавања била тачна и прецизна. Када се покрене калибрација на екрану ће се приказати порука „Tilt to fill screen“ („Нагни да попуниш екран“) и након тога треба померати микробит уређај тако да цео екран буде испуњен тачкама. Важно је знати да не постоји посебан сензор за температуру који је уграђен у микробит, али је могуће користити сензор који мери температуру процесора микробит уређаја у степенима Целзијуса. Компонента за радио је један од начина на који микробит уређаји могу да комуницирају међусобно слањем и примањем порука. BLE (Bluetooth Low Energy) антена уз помоћ које микробит уређај може бежично да комуницира са различитим уређајима. Комуникација са уређајима је двосмерна – микробит може да прима податке, али и да их прослеђује. На задњем делу микробит уређаја налази се и УСБ улаз који омогућава повезивање микробита са рачунаром помоћу УСБ кабла.

Слика 1.1.5: Задња страна микробит уређаја

1.2 Пакети микробит уређаја

„Школе за 21. век[1]“ је едукативни програм, који спроводи британска Влада у циљу развоја вештина критичког мишљења, решавања проблема и програмирања. У оквиру овог пројекта, свака школа је добила одређени број микробит уређаја који ученици могу да користе за програмирање, као и решавање проблема из разних области.

Основни пакет који су, у оквиру поменутог пројекта, добиле школе у Србији, омогућава све што је потребно за повезивање микробит уређаја са рачунаром као и за напајање батеријама. Основни пакет, који је приказан на слици 1.2.1, садржи: један BBC микробит уређај, један УСБ кабл, један држач батерија и две AAA батерије.

[1] Програм „Школе за 21. век на Западном Балкану“ има за циљ пружање подршке школама из региона Западног Балкана које учествују у програму кроз обуку и подршку школским лидерима, обуку и подршку за наставнике као и кроз доступност ресурса за развој дигиталних вештина ученика и умрежавање школа.

Слика 1.2.1: Основни пакет микробит уређаја

Поред основног пакета, постоји мноштво додатака за микробит уређај, који омогућавају да буду реализовани напредни пројекти. Велики број задатака и пројеката могуће је реализовати без коришћења додатне опреме. Међутим, употребом додатних компоненти задаци постају занимљивији и јасније приказују колико је једноставна примена програмирања и савремених технологија у решавању проблема из свакодневног живота. Неки од додатака су: светлеће диоде, крокодилке, звучници и слушалице, потенциометар, фото-отпорник, серво мотори итд. Начин на који се додатне компоненте могу повезати на микробит уређај, као и њихова употреба, биће приказани у наредним главама овог рада.

2 Мејк Код окружење

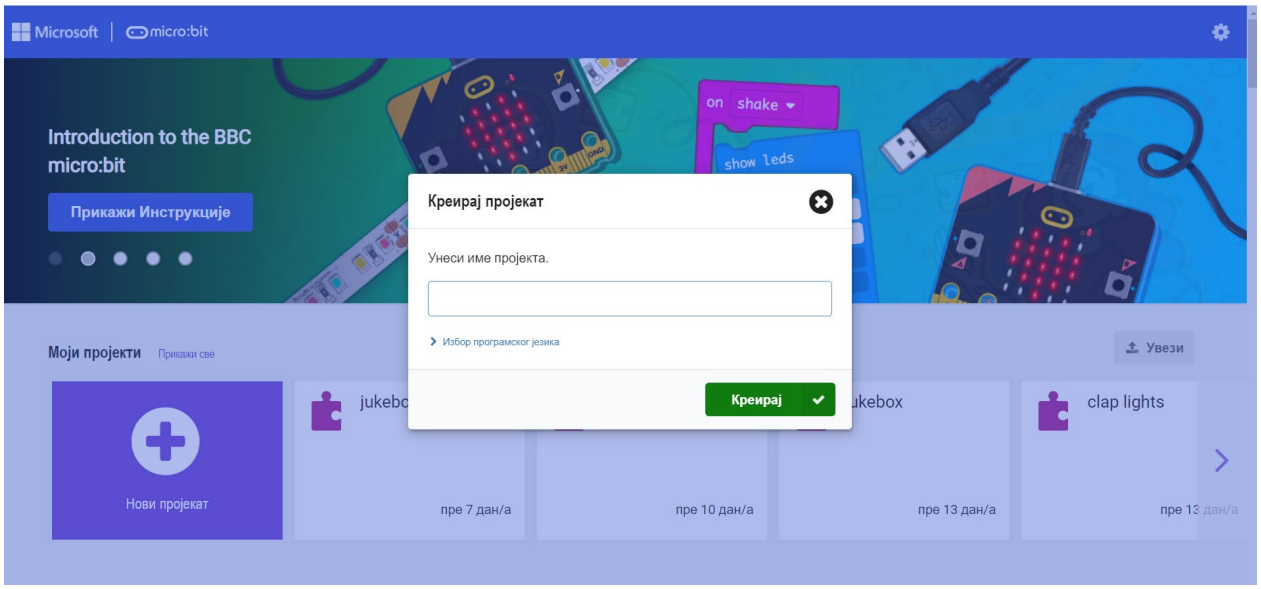
У циљу једноставнијег програмирања микробит уређаја, постоји пет различитих начина за програмирање микробита. Ученици могу да бирају: *Touch Develop*, *Code Kingdoms' JavaScript editor*, Мајкрософтов *Block Editor*, *MicroPython* или *PXT*.

У настави информатике и рачунарства у петом разреду основне школе за програмирање микробит уређаја користи се окружење за блоковски програмски језик Мејк Код. С обзиром да се из области рачунарства у петом разреду основне школе обрађује програм за визуелно програмирање Скреч (енгл. Scratch), ученици су упознати са превлачењем блокова у циљу низања наредби. У овој глави биће приказан интерфејс Мејк Код окружења, као и основне функције које пружа ово окружење.

Мајкрософт Мејк Код је окружење које се користи за програмирање уређаја као што је микробит. Софтверска апликација Мејк Код садржи све што је потребно за писање, компилацију, покретање, тестирање и отклањање грешака у програму. За коришћење овог окружења потребно је користити било који веб прегледач и није потребна инсталација на рачунар. Као у Скречу и у Мејк Коду програми се могу креирати превлачењем блокова који су различитих боја, који нам указују на врсту наредбе.

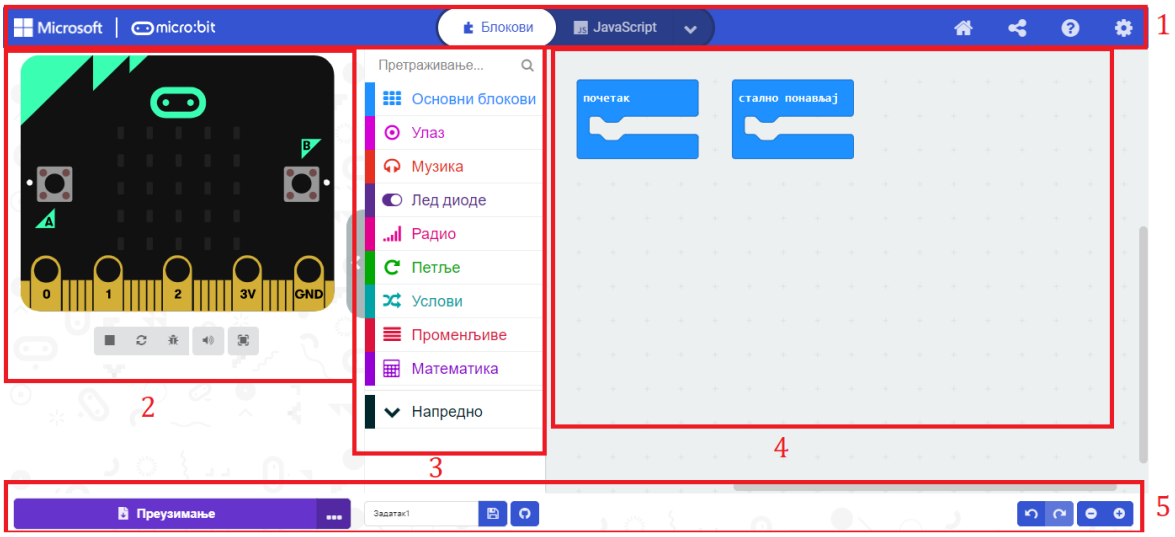
2.1 Интерфејс Мејк Код програмског окружења

За приступ Мејк Код окружењу потребно је отворити интернет страницу: https://makecode.microbit.org/. Мејк Код окружење подржава више језика на којима може бити приказан. Међу тим језицима је и српски језик. Уколико је потребно, могуће је одмах променити језик на српски кликом на икону за подешавања која се налази у десном углу, а затим одабиром српског језика из понуђене листе. Након тога, потребно је левим тастером миша кликнути на дугме *Нови пројекат*, а затим је потребно унети назив пројекта. Да би процес креирања пројекта био готов, потребно је левим тастером миша кликнути на дугме *Креирај пројекат* као што је приказано на слици 2.1.1.



Слика 2.1.1: Креирање новог пројекта у Мејк Код окружењу

Тада, нови пројекат је направљен и приказује се интерфејс Мејк Код окружења као на слици 2.1.2.



Слика 2.1.2: Интерфејс Мејк Код окружења

На самом врху екрана налази се трака менија (1) која садржи више функција. Једна од основних функција ове траке јесу два дугмета која омогућавају прелазак из блоковског у текстуални програмски језик и обрнуто. Такође, постоји функција „Share“ помоћу које је могуће објавити пројекат и поделити га са другима путем линка. Мени трака садржи и опцију за подешавања у оквиру које је могуће променити језик. Мејк Код окружење подржава и српски језик. Са леве стране екрана налази се виртуелни микробит симулатор (2), који се покреће аутоматски када год се направи измена у програму. Уз помоћ овог симулатора може се видети како ће изгледати програм када се преузме на прави микробит уређај. Симулатор је јако важан јер, пре свега, омогућује брзо откривање и отклањање грешака. Испод симулатора налазе се тастери уз помоћ којих је могуће зауставити или покренути програм, укључити или искључити звук, као и поставити приказ симулатора преко целог екрана. У средини екрана са десне стране симулатора налазе се категорије програмских блокова (3), које су приказане у различитим бојама. У оквиру сваке од категорија налазе се наредбе које омогућавају креирање програма и Мејк Код окружењу. У зависности од категорије која је одабрана могуће је приступити различитим специфичним карактеристикама микробит уређаја, основним програмским структурама, као и напредним функцијама које се користе за рад микробит уређаја. Са десне стране екрана налази се радна површина за програмирање (4) која служи за креирање и уређивање програма. Превлачењем одговарајућих блокова из палете блокова, едитор Блока пружа могућност креирања програма, који ће касније бити покренут на уређају микробит. На дну екрана налази се трака са алаткама (5). Трака са алаткама садржи дугме за чување програма на рачунару, који се копира на микробит уређај. Ова трака садржи још и опције за враћање корака, као и увећање или смањивање приказа симулатора.

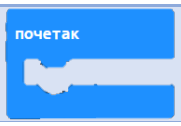
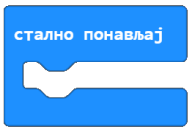
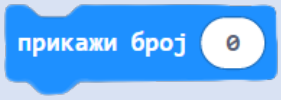
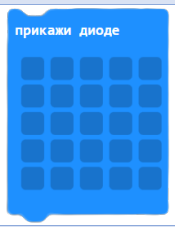
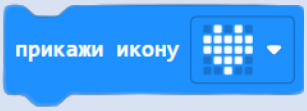
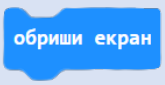
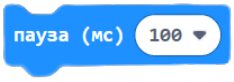
Главна предност Мејк Код окружења јесу блокови, који олакшавају почетке учења програмирања јер, приликом превлачења и повезивања блокова, није могуће направити синтаксну грешку као приликом програмирања у текстуалном програмском језику.

2.2 Категорије блокова

Програмски блокови у Мејк Код окружењу се деле у одређене категорије у зависности од тога коју улоге имају. Такође, свака од категорија је приказана различитом бојом. Категорије у Мејк Коду су: *Основни блокови, Улаз, Музика, Лед диоде, Радио, Петље, Услови, Променљиве, Математика и Напредно*.

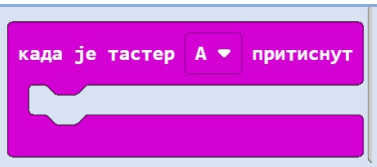
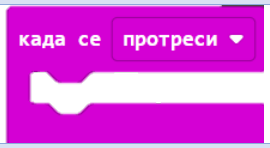
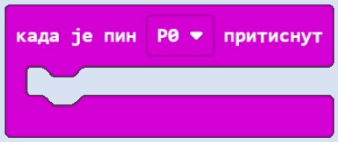
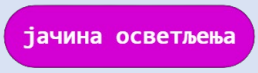
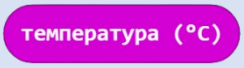
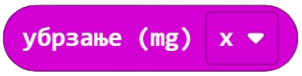
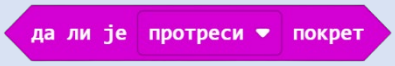
У категорији *Основни блокови* налазе се блокови са основним функцијама за рад са микробит уређајем. Блокови категорије Основни блокови су плаве боје. У табели 2.2.1 налазе се називи и објашњење неких од блокова који ће бити коришћени у задацима.

Табела 2.2.1: Блокови у категорији Основни блокови

Блок	Објашњење
	Наредбе које се налазе у блоку *почетак*, биће извршене приликом покретања микробит уређаја.
	Блок *стално понављај* је блок понављања и наредбе у оквиру овог блока се извршавају док се микробит уређај не искључи.
	Блок *прикажи број* служи за приказ нумеричких података на екрану микробит уређаја.
	Блок *прикажи диоде* служи за приказивање слика на екрану микробит уређаја, тако што се кликом на лед диоду бира која ће диода бити укључена.
	Блок *прикажи икону* служи за приказ већ креираних облика, бирањем из падајућег менија.
	Блок *прикажи текст* служи за приказ ниске карактера. На микробит уређају могу бити приказани само *ASCII* кодови знакова од 32 до 126. У те знакове спадају слова, бројеви, интерпункцијски знаци и неки симболи.
	Блок *обриши екран* служи за искључивање лед диода.
	Блок *пауза* служи за дефинисање паузе између наредби изражене у милисекундима.

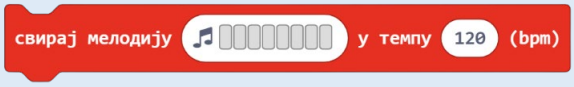
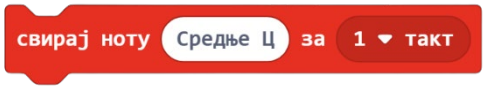
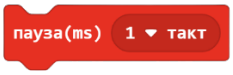
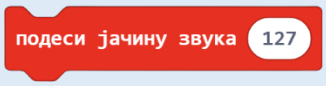
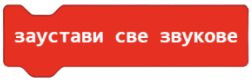
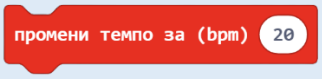
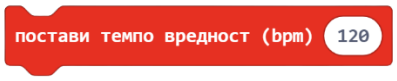
У категорији *Улаз* налазе се блокови помоћу којих је могуће очитати податке са сензора, као и блокови којима се дефинишу догађаји. Блокови категорије *Улаз* су љубичасте боје и приказани су табели 2.2.2.

Табела 2.2.2: Блокови у категорији Улаз

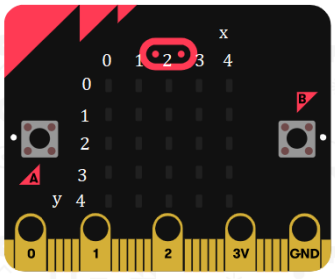
Блок	Објашњење
	Блок *када је тастер притиснут* служи за извршавање наредби уколико је притиснут тастер А, тастер В или оба тастера.
	Блок *када се* служи за извршавање кода уколико је детектован неки покрет (на пример померање микробита).
	Блок *када је пин притиснут* служи за извршавање наредби онда када је притиснут неки од пинова.
	Блок *тастер је притиснут* прихвата информацију о томе да ли неки од тастера А или В притиснут или не.
	Блок *јачина осветљења* служи за очитавање нивоа осветљења лед диода. Вредности очитавања нивоа осветљења могу бити од 0 до 255.
	Блок *смер компаса* служи за очитавање тренутне вредности компаса у степенима.
	Блок *температура* служи за очитавање температуре процесора микробит уређаја у степенима Целзијусовим.
	Блок *убрзање* мери вредност убрзања. Ако је екран микробит уређаја окренут на горе, вредности координата су x=0, y=0 и z=-1024.
	Блок *да ли је покрет* прихвата информацију томе да ли је дошло до покрета микробит уређаја.

У категорији *Музика*, као што и сам назив каже, налазе се блокови за рад са музиком, као и блокови за креирање тонова кроз пин 0. Блокови у категорији *Музика* су црвене боје и приказани су у табели 2.2.3. Да би мелодија била репродукована, микробит уређај мора, додатном опремом, бити прикључен на звучник уз помоћ пинова. Међутим, у Мејк Код симулатору ће бити репродукована мелодија.

Табела 2.2.3: Блокови у категорији Музика

Блок	Објашњење
	Блок *свирај мелодију* омогућава да микробит уређај репродукује мелодију која може бити одабрана из галерије мелодија или направљена у едитору. Такође у овом блоку је могуће одабрати брзину односно темпо у којем ће мелодија бити репродукована.
	Блок *свирај ноту* омогућава да микробит уређај репродукује ноту која може бити изабрана кликом на ноту виртуелне клавијатуре. Такође, може бити изабран и такт ноте.
	Блок *пусти тон* репродукује онај тон на микробит уређају који одаберемо кликом на виртуелну клавијатуру. Тон који је одабран биће репродукован све док се микробит не искључи.
	Блок *пауза* служи за то да се ништа не свира одређено време преко пина P0.
	Помоћу блока *подеси јачину звука*, могуће је подесити подразумевану излазну јачину синтетизатора звука.
	Блок *заустави све звукове* служи да сви звукови и мелодије које тренутн свирају буду заустављени.
	Блок *промени темпо* служи за промену темпа на основу уписане вредности.
	Блок *постави темпо*, поставља темпо мелодије на подразумевану вредност.

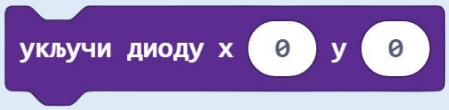
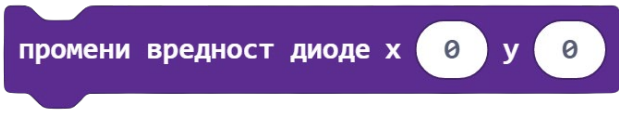
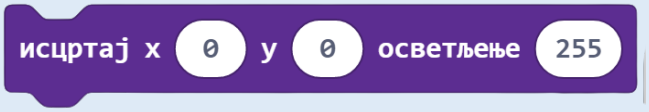
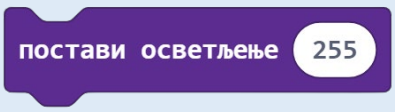
У категорији *Лед диоде* налазе се блокови за манипулацију лед диодама на екрану микробит уређаја. Екран микробит уређаја се састоји из 25 лед диода, које чине 5x5 квадратну мрежу, која је приказана на слици 2.2.1.



Слика 2.2.1: Приказ квадратне мреже

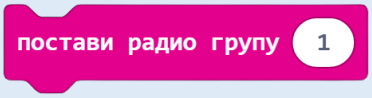
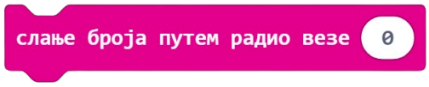
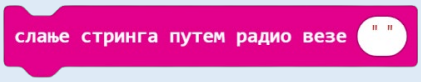
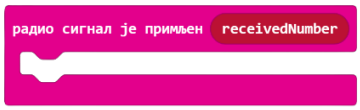
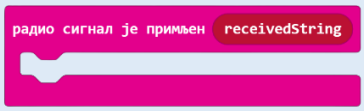
Положај лед диоде у квадратној мрежи могуће је одредити уз помоћ *x* и *y* координате. Координата *x* представља хоризонтални положај диоде, док *y* координата представља вертикални положај диоде. Прва лед диода, односно диода у горњем левом углу има координате (0,0), док последња лед диода односно диода у доњем десном углу има координате (4,4). Користећи координате положаја, лед диоде је могуће искључивати и укључивати и стварати креативне и занимљиве приказе на екрану микробит уређаја. Блокови за контролисање лед диода налазе се у табели 2.2.4.

Табела 2.2.4: Блокови у категорији Лед диоде

Блок	Објашњење
	Блок *укључи диоду*, укључује осветљење специфичне лед диоде користећи позицију диоде. Позиција се задаје уношењем *x* и *y* координате.
	Блок *промени вредност диоде*, укључује/искључује осветљење диоде у зависности од тога да ли је диода била искључена/укључена.
	Блок *искључи диоду*, искључује осветљење специфичне лед диоде користећи позицију диоде. Позиција се задаје уношењем *x* и y координате.
	Блок *диода на координати* даје стање укључено/искључено специфичне лед диоде користећи *x* и *y* координате позиције лед диоде.
	Блок *исцртај* укључује лед диоду са *x* и *y* координатама задатог интензитета осветљења.
	*Осветљење* је променљива која чува вредност интензитета светлости од 0 до 255.
	Блок *постави осветљење* поставља осветљење интензитета од 0 до 255.
	Блок *лед диода укључена* укључује, односно искључује екран.

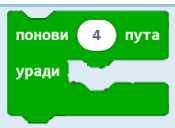
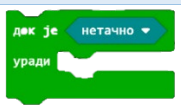
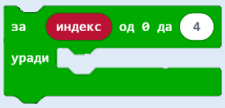
У категорији *Радио* налазе се блокови уз помоћ којих микробит може да комуницира са другим микробит уређајима успостављајући радио комуникацију. Микробит уређај комуницира са другим микробит уређајима слањем и примањем података користећи блокове које се налазе у табели 2.2.5.

Табела 2.2.5: Блокови у категорији Радио

Блок	Објашњење
	Блок *постави радио групу* служи за постављање ID групе за комуникацију бежичним путем. Микробит уређај може да прими само Idједне групе у једном тренутку.
	Блок *слање броја путем радио везе* објављује број бежичним путем сваком повезаном микробит уређајем у групи.
	Блок *слање стринга путем радио везе* преноси стринг заједно са серијским бројем уређаја и временом извршавања сваком повезаном микробит уређајем у групи.
	Овај блок извршава наредбе које су задате онда када прими број бежичним путем.
	Овај блок извршава наредбе које су задате онда када прими ниску карактера бежичним путем.

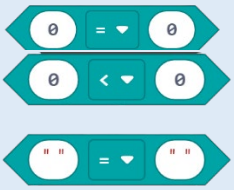
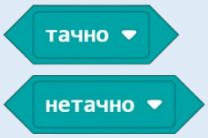
Како би програми били занимљивији често је потребно да се одређене наредбе или блокови наредби понове више пута. Понављање спада у један од основних концепата програмирања. Када је у програму потребно неке наредбе поновити виш пута, тада програм садржи петље. Петље представљају снажан концепт у програмирању. Мејк Код окружење садржи три врсте блокова које је могуће користити у програмима са понављањима и у које је потребно убацити друге блокове чије извршење треба поновити. Блокови се могу понављати: одређени број пута, бесконачан број пута све док корисник не прекине програм и док се не испуни одређени услов. Неки од блокова који се налазе у категорији Петља, као и објашњења њихових примена, налазе се у табели 2.2.6.

Табела 2.2.6: Блокови у категорији Петља

Блок	Објашњење
	Блок *понови* извршава код онолико пута колико корисник унесе у предвиђено поље.
	Блок *док* (енгл. while) извршава код све док је услов испуњен.
	Блок *за* (енгл. for) извршава код одређени број пута користећи бројач (индекс).
	Блок *прекид* прекида петљу и извршава прву наредбу након петље.
	Блок *настави* прескаче тренутну итерацију и наставља са следећом итерацијом у петљи.

Сваки човек се свакодневно сусреће са доношењем одлука. Одлучивање у програмима је један од основних и најважнијих концепта програмирања. У зависности од тога да ли је услов испуњен или није извршава се једна, односно друга група наредби. У програмима одлучивања користи се наредба гранања. Приликом писања услова у оквиру наредбе гранања потребно је користити логичке операторе, као и операторе поређења. У табели 2.2.7. приказани су блокови из категорије *Услови*.

Табела 2.2.7: Блокови у категорији Услови

Блок	Објашњење
	Блок *ако је* извршава блок наредби само ако је испуњен услов. Ако услов није испуњен прескаче се наредба гранања и извршава се следећа наредба.
	Овај блок представља потпуни облик наредбе гранања. У зависности од испуњености услова извршава се једна или друга група наредби.
	Приликом писања и постављања услова потребно је користити операторе поређења. Уз помоћ ових оператора могуће је проверити да ли су два броја једнака, као и да ли је један мањи од другог и обрнуто. Такође, могуће је поредити и ниске карактера, односно стрингове. Резултат поређења може бити тачно (енгл. true) или нетачно (енгл. false).
	Поред оператора поређења, приликом писања услова могуће је користити и логичке операторе. Логички оператори су неопходни онда када је потребно комбиновати више услова.
	У овим блоковима се налазе логичке вредности тачно и нетачно.

Променљиве представљају једну од основних програмских структура. Променљиве у програмирању јесу објекти који имају вредност. Вредност променљиве се може мењати у току извршавања програма, али није могуће мењати њен тип. У оквиру категорије *Променљиве* могуће је направити нову променљиву.

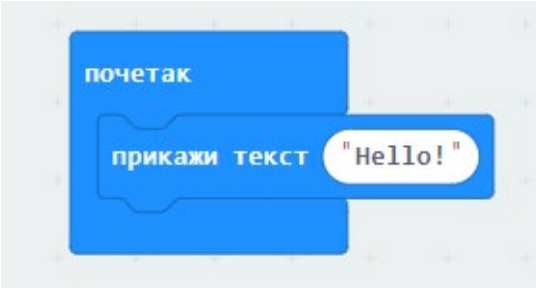
Основне рачунске операције, дељивост и математичке функције представљају незаобилазан сегмент приликом писања програма. У категорији *Математика* налазе се блокови за аритметичке операције и разне математичке функције.

2.3 Покретање програма

Покретање и тестирање програма представљају једне од најважнијих корака у програмирању. У Мејк Код окружењу постоји могућност покретања и тестирања програма у микробит симулатору. Међутим, постоји могућност и покретања програма на самом микробит уређају. Пре него што буде приказано преузимање и покретање програма на микробит уређају потребно је, за почетак, написати уводни програм. Углавном ,приликом првог сусрета са било којим програмским језиком, уводни програм јесте да на екрану буде исписана порука „Zdravo,

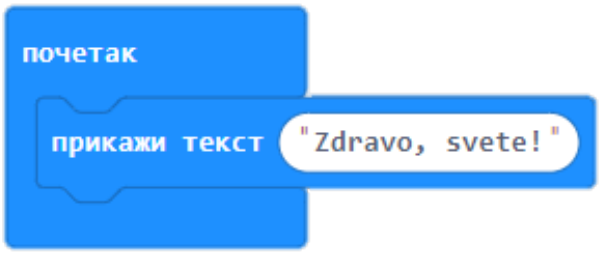
svete!“. Управо то ће бити уводни пример и у овом раду. Дакле, потребно је да на микробит уређају буде исписана порука „Zdravo, svete!“.

Прво, потребно је посетити веб адресу https://makecode.microbit.org и направити нови пројекат као што је објашњено у одељку 2.1. Приликом отварања пројекта у делу за програмирање биће приказани основни блокови *почетак* и *стално понављај*, које је могуће обрисати уколико тренутно нису потребни. За овај уводни пример биће потребан само блок *почетак*. Дакле, основни блок стално понављај је потребно обрисати. Да би блок био обрисан, потребно је десним тастером миша кликнути на њега, а затим одабрати опцију брисање блока. Сада, потребно је одабрати блок *прикажи текст*, који се налази у категорији *Основни блокови* и држећи леви тастер миша превући га до блока *почетак*. Код треба да изгледа као на слици 2.3.1.



Слика 2.3.1: Блок почетак

У делу предвиђеном за унос текста, уместо текста „Hello!“ треба уписати „Zdravo, svete!“. Тест мора бити унет латичиним писмом. На крају, код треба да изгледа као на слици 2.3.2.

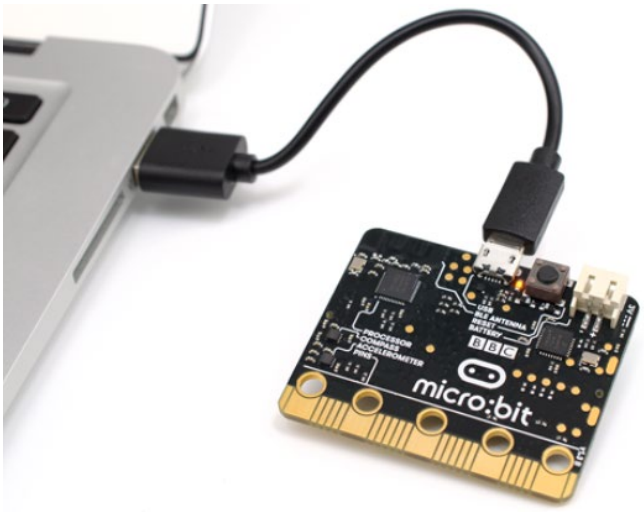


Слика 2.3.2: Измена садржаја у текстуалном блоку

За покретање и тестирање програма постоје две начина: покретање у симулатору или пребацивање програма на микробит уређај.

Кликом на зелено дугме које се налази испод симулатора покреће се програм у Мејк Код симулатору.

Да би програм био пребачен на микробит уређај, прво је потребно преузети програм на рачунар. Кликом на дугме преузимање на рачунар ће бити преузета датотека са екстензијом hex, коју је потребно пребацити на микробит уређај. Да би фајл био преузет на микробит потребно је повезати рачунар и микробит уређај помоћу УСБ кабла као што је приказано на слици 2.3.3.



Слика 2.3.3: Повезивање микробит уређаја и рачунара помоћу УСБ кабла

Да би било јасније на који начин се чува програм у микробит уређају, потребно је рећи више о меморијама у рачунару: променљивој и трајној меморији. Променљива меморија представља место у којем се складиште информације, али смо док је рачунар прикључен на напајање. Садржај меморије нестаје, онда када се рачунару, односно микробит уређају уклони напајање. Трајна меморија складишти и чува информације чак и онда када рачунар није прикључен на напајање.

Након повезивања на рачунар микробит уређај ће се приказати у делу за нове повезане уређаје. Сада, потребно је пребацити .hex фајл на уређај микробит. Пребацивање је могуће урадити превлачењем и отпуштањем (енгл. drag and drop) преузетог фајла на микробит уређај. Процес пребацивања датотеке назива се треперење (енгл. flashing). Приликом пребацивања фајла на микробит уређају ће треперети лед диода која се налази на његовој полеђини односно задњој страни. Када диода почне константно да светли, тада је пренос програма на микробит уређај завршен. Сада, програм може бити покренут на микробит уређају, али микробит мора бити прикључен на екстерну батерију. Повезивање на материју се врши помоћу конектора који се налази на крају држача за батерије. На микробит уређају може бити сачуван само један програм, који ће бити извршаван све док се на уређај не пребаци други програм.

Приликом пребацивања програма на микробит, он се пребацује у једну врсту трајне меморије која се зове флеш меморија. Све информације које су складиштене у овој меморији биће доступне микробиту кад год је укључен. Из тог разлога, ако се прекине веза између рачунара и микробита, па се после неког времена поново успостави, на микробиту ће аутоматски бити покренут програм који је последњи учитан. Приликом покретања програма, микробит користи рам меморију. То значи да ће све промене које буду извршене приликом покретања програма бити заборављене ако се микробит уређај ресетује или се укључи и искључи напајање.

2.4 Слагање блокова

Слагање, односно низање блокова представља један од најважнијих концепата приликом процеса учења програмирања. Да би рачунар извршио задатак, потребно је да му програмер зада низ корака које треба да изврши у правилном редоследу. Програмер прво треба да осмисли алгоритам за решавање неког проблема. Алгоритам представља скуп операција које треба

извршити по тачно одређеном редоследу. Алгоритми, односно низови корака користе се у свакодневном животу. Приликом састављања алгоритма за решавање неког проблема, пре свега је потребно разумети проблем који је потрено решити. Врло често се дешава да је почетни проблем потребно раставити на више мањих, познатих проблема. Када се почетни проблем рашчлани на више мањих проблема, онда је потребно осмислити кораке, односно упутства за њихово решавање. Размишљање о проблему и писање алгоритма за његово решавање представља кључни појам у савладавању концепта програмирања.

Постоје три врсте алгоритама:

1. линијски алгоритми;
2. циклични алгоритми;
3. условни (разгранати) алгоритми.

У линијском алгоритму наредбе се извршавају редом, онако како су написане, корак по корак, при чему се свака од наредби извршава тачно једанпут.

У цикличним алгоритмима одређене наредбе се могу поновити више пута, углавном у зависности од испуњености неког услова. Приликом писања цикличних алгоритама користе се петље.

У условним алгоритмима неке наредбе ће се извршити, а неке не, у зависности од испуњености услова.

Дакле, приликом решавања проблема потребно је саставити алгоритам, а након тога рачунару задати низ корака које треба да изврши.

Приликом решавања проблема, потребно је саставити низ корака, односно наредби које треба да изврши неки уређај. Тај низ корака назива се програм или скрипта. У Мејк Код окружењу програм је састављен од блокова који су повезани. Дакле, приликом задавања низа корака у Мејк Код окружењу није потребно куцати наредбе, већ је довољно сложити блокове тако да се добије смислен програм.

У програмирању, постоји више фаза приликом решавања проблема.

Фаза 1: Размишљање о проблему и састављање алгоритма за његово решење.

Фаза 2: Поступајући по корацима из алгоритма, сложити блокове из различитих категорија.

Фаза 3: Тестирати програм.

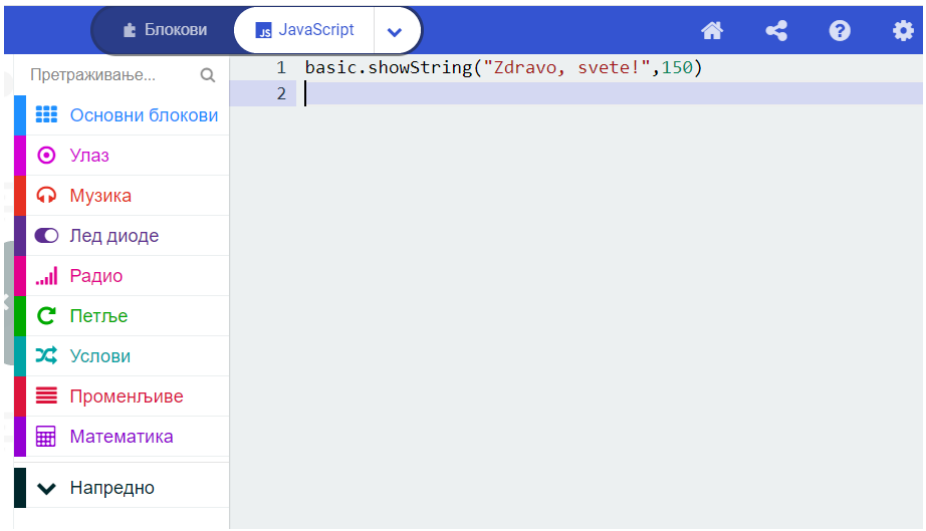
Фаза 4: Исправљање грешака.

Пролазећи кроз све фазе решавања проблема ученици основних школа развијају дигиталне компетенције и стичу вештине критичког размишљања. У првој фази потребно је да ученици пре свега разумеју дати проблем и размишљањем, проматрањем и анализом дођу до низа корака за његово решавање. Затим, потребно је саставити алгоритам за његово решавање. На основу алгоритма, низањем корака рачунару треба дати низ инструкција које треба да изврши. Јако важне фазе у решавању проблема јесу тестирање програма и исправљање грешака у програму. Пре свега, треба предвидети шта све може да се догоди у програму. Уколико се приликом тестирања програма са различитим величинама прикаже грешка, треба анализирати програм корак по корак. То је најбољи начин за поступно и темељно отклањање грешака у програму, јер је онда могуће увидети да ли је грешка у идеји решења проблема или је грешка у слагању блокова. Након што се увиди где је проблем у програму, потребно је исправити грешке и након исправке поново тестирати програм. То је изузетно важно за ученике основних школа јер на својим грешкама могу много брже и лакше да науче програмирање.

Визуелно програмско окружење, као што је Мејк Код, пружа могућност да, слагањем (низањем) блокова, ученици решавају једноставне, али и комплексне проблеме, програмирају игре и имају пуно могућности да изразе креативност.

Слагање блокова представља методу програмирања којом је потребно одабрати категорију и блок, а након тога превући тај блок до радне површине за програмирање. Превлачењем блокова и надовезивањем једног на други рачунар добија комплетан код, односно низ наредби које треба да изврши. Да би уређивач блокова (енгл. block editor) препознао код који треба бити креиран, сваки блок мора бити унутар управљача догађаја (енгл. event handler). То су блокови: почетак, стално понављај, кад је тастер A притиснут итд.., који омогућавају осталим блоковима да кликну изнутра. Ако је нека команда превучена у уређивач блокова ван управљача догађајима, биће прекрижена и означена сивом бојом.

У претходном одељку приказано је решење једноставног почетног примера у којем је на екрану микробит уређаја исписана порука „Zdravo, svete“. У том примеру искоришћена је једна од основних програмских структура, стрингови. Приликом коришћења стрингова односно ниске карактера у програмирању микробит уређаја, треба водити рачуна о томе да на екрану микробит уређаја могу бити приказани само *ASCII* кодови знакова од 36 до 126. У ове знакове спадају велика и мала слова, бројеви, интерпункцијски знакови и неки од специјалних карактера. Сви остали знаци се приказују као ? на екрану микробит уређаја. Уређивач блокова Мејк Код не дозвољава кориснику да подеси брзину којом ће се приказивати стринг на екрану. Међутим, уколико је потребно променити брзину појављивања стринга, то се може постићи преласком на ЈаваСкрипт (енгл. JavaScript) код, приказаног на слици 2.4.1. Онда је потребно, у функцији *basic.showstirng()*, додати још један параметар који ће представљати брзину кретања слова.



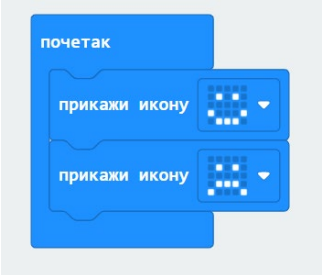
Слика 2.4.1: Јава скрипт код

У наставку ће бити приказана решења још неких једноставнијих линијских програма, који приказују коришћење основних блокова.

1. Написати програм који приказује смењивање насмејаног и тужног лица.

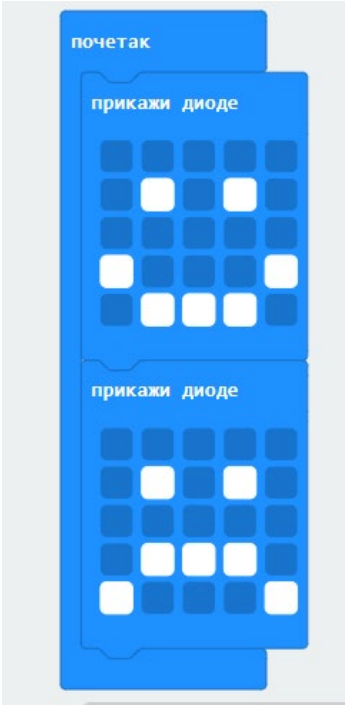
Приказ срећног или тужног лица на екрану микробита представља укључивање и искључивање одређених светлећих диода. Најпре је потребно, да из дела за основне блокове,

буде одабран блок *почетак*. За приказ слика на екрану микробит уређаја постоје два начина: користећи блок *прикажи диоде* или користећи блок *прикажи икону*. Користећи блок *прикажи икону*, из падајућег менија је могуће одабрати једну од понуђених, већ направљених слика. Међутим уколико се међу уграђеним сликама не налази она која је потребна, онда је неопходно користити блок *прикажи диоде*. Да би биле приказане слике насмејаног и тужног лица, потребно је два пута превући блок *прикажи икону* или *прикажи диоде* на радну површину за програмирање, тако да се укопе са блоком *почетак*. Ако је одабран блок *прикажи иконе*, тада из падајућег менија треба одабрати слике насмејаног и тужног лица, као што је приказано на слици 2.4.2.



Слика 2.4.2: Приказ насмејаног и тужног лица

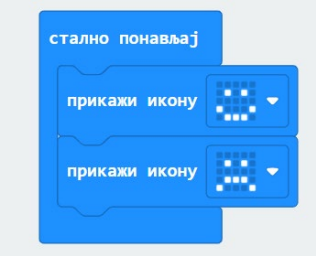
Ако је одабран блок *прикажи диоде*, онда је кликом на квадратну мрежу која је приказана могуће одабрати диоду која ће бити осветљена, као што је приказано на слици 2.4.3.



Слика 2.4.3: Приказ насмејаног и тужног лица коришћењем квадратне мреже

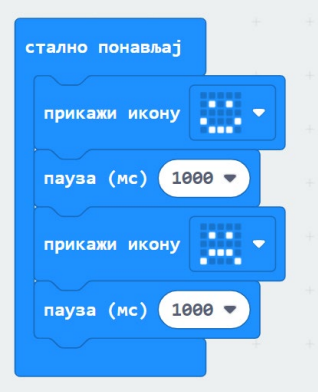
Када је слагање блокова завршено, тада је потребно покренути програм и евентуално отклонити грешке у програму. Претходни програм нема грешака, али програм не ради баш оно

што је требало. У овом задатку треба постићи то да се изрази лица смењују све док се микробит уређај не рестартује или искључи. Блок *почетак* омогућава да се наредбе које се налазе у оквиру овог блока изврше само једном. Зато је блок почетак потребно заменити блоком *стално понављај*, као што је приказано на слици 2.4.4.



Слика 2.4.4: Приказ насмејаног и тужног лица коришћењем блока стално понављај

Мана овог решења јесте та што се слике брзо мењају. Зато је између блокова прикажи икону, потребно додати блок *пауза* и дефинисати интервал којим се чека извршење следећег блока. Временски интервал се дефинише у милисекундима. Код би требало да изгледа као на слици 2.4.5.



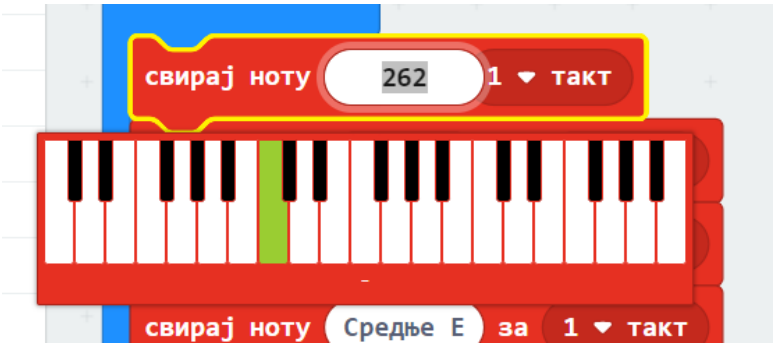
Слика 2.4.5: Приказ насмејаног и тужног лица коришћењем блока стално понављај са паузом

2. Написати програм којим је потребно направити мелодију песме „Вишњичица род родила“.

Да би на микробит уређају био одсвиран неки тон или музика, потребно је користити блокове из категорије *Музика*. Ноте за песму коју је потребно репродуковати на микробиту јесу: до, до, ре, ми, сол, сол, фа, ми, ла, сол, фа, ми, ре, ми, фа, ми, до, ре, ре. Уз помоћ блока *свирај ноту*, могуће је стварати и репродуковати музику на уређају микробит. Зато ће програм за овај задатак бити сачињен од блокова *свирај ноту*. Код треба да изгледа као на слици 2.4.6.

Слика 2.4.6: Блокови за репродукцију нота

Приликом одабира ноте, тј. кликом на поље за одабир, појављује се виртуелна клавијатура помоћу које је могуће изабрати тон, као што је приказано на слици 2.4.7.

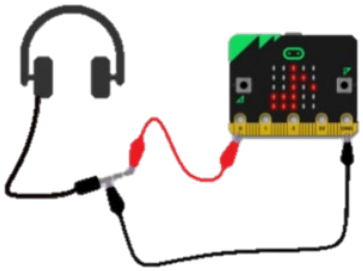


Слика 2.4.7: Виртуелна клавијатура

Из претходне слике, могуће је приметити да приликом одабира ноте, уместо њеног назива, приказује се број. Тај број представља фреквенцију тона који је одабран. У зависности од фреквенције, тонови су дубљи или виши. Микробит може да одсвира одређени тон, на основу његове фреквенције. Најдубљи понуђени тон у виртуелној клавијатури има фреквенцију 131Hz, док највиши има фреквенцију 998Hz. Поред ових датих фреквенција, могуће је унети број није превазилази овај опсег и на тај начин добити друге тонове, који имају већу или мању фреквенцију. У том случају, уместо назива тона, приказаће се његова фреквенција. Микробит уређај нема уграђен звучник и не може директно да репродукује звук. Зато је, да би музика била репродукована, неопходно микробит повезати на слушалице или звучник. Да би микробит био повезан на слушалице, потребна су два кабла са крокодил хватаљкама приказаних на слици

2.4.8. Крајеве два кабла треба повезати за GND и пин 0. Преостале крајеве треба прикачити за слушалице као што је приказано на слици 2.4.9.

Слика 2.4.8: Кабл са крокодил штипаљкама



Слика 2.4.9: Повезивање слушалица и микробит уређаја

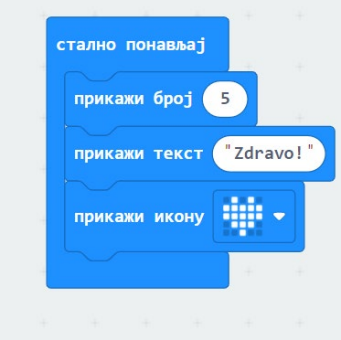
Стварање музике користећи микробит уређај представља незаобилазна приступ приликом приближавања програмирања ученицима основне школе. Користећи један блок из категорије *Музика*, ученици су у могућности да изразе своју креативност, али и развију дигиталне компетенције.

3. Написати програм којим се на екрану микробит уређаја приказује број, текст и слика. Исправити код тако да се наредбе понављају све док се микробит не искључи.

Стрингови (ниске) и бројеви представљају основне типове података у програмирању. На овом примеру ће бити приказан начин на који ови подаци могу бити приказани у блоковском програмском језику. Прво, потребно је приказати број. За приказ бројева користи се блок *прикажи број* из категорије *Основни блокови*. Онда је потребно у пољу за унос, унети број који ће бити приказан на микробиту. Нека то буде број 5. Након тога, потребно је одабрати блок *прикажи текст* и у пољу за унос откуцати текст „Здраво“. Слика која ће бити приказана на крају програма јесте слика срца, коју је могуће изабрати из падајућег менија. Код програма би требало да изгледа као на слици 2.4.10.

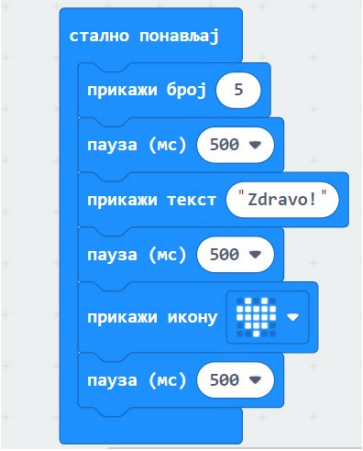
Слика 2.4.10: Приказ броја, текста и слике у оквиру блока почетак

Наредбе у коду са слике ће се извршити само једном када се покрене микробит уређај. Код треба изменити тако да се наредбе извршавају све док се микробит не искључи или рестартује. То се постиже тако што се уместо блока *почетак* искористи блок *стално понављај*. Након измене, код треба да изгледа као на слици 2.4.11.



Слика 2.4.11: Приказ броја, текста и слике у оквиру блока стално понављај

Како би приказ броја, текста и слике на микробит уређају био прегледнији потребно је још мало изменити код, тако да се између појављивања сваког од типа података дода пауза од 500 милисекунди. На крају, код треба да изгледа као на слици 2.4.12.



Слика 2.4.12: Приказ броја, текста и слике са паузом између појављивања

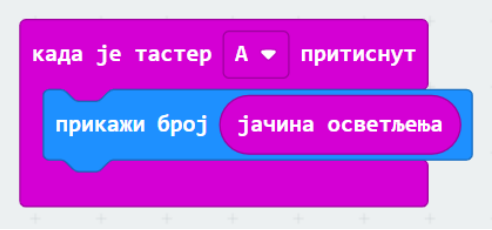
2.5 Очитавање података са сензора и контрола лед диода

Као што је већ речено у првој глави овог рада, микробит уређај садржи различите сензоре. Помоћу сензора микробит може да одреди ниво осветљености у просторији, одреди температуру просторије, одреди јачину магнетног поља, као и да препозна покрет. Сви набројани параметри које микробит може да одреди, чине да програми које је могуће написати за микробит буду још занимљивији и применљивији у свакодневном животу. Поред очитавања података са сензора, у овом делу биће приказана и манипулација лед диодама.

У наставку ће бити приказани примери који осликавају употребу управо ових сензора. Приликом решавања тих задатака биће коришћени блокови за управљање догађајима. Помоћу управљача догађајима, корисник може да одреди које наредбе ће бити извршене. У зависности од тога који је тастер притиснут, да ли је микробит померен, да ли је притиснут лого итд. биће извршене различите групе наредби.

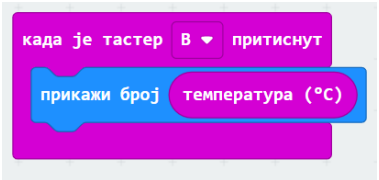
1. Написати програм који када је притиснут тастер А приказује ниво осветљености, када је притиснут тастер Б приказује температуру у степенима Целзијусовим, а када је притиснут тастер А+Б приказује смер компаса у степенима.

   Прво, из категорије *Улаз*, потребно је одабрати блок за регистровање притиска тастера. Из падајућег менија, одабрати тастер А. Ниво осветљености у просторији јесте број из интервала од 0 до 255. Ако је вредност осветљења 0, тада је у просторији мрак, а ако је вредност 255, то значи да је просторија јако осветљена. Са обзиром да је јачина осветљења број, како би био приказан, потребно је одабрати блок *прикажи број* и превући га тако да се уклопи са блоком за регистрацију притиска тастера. Након тога, из категорије *Улаз* треба изабрати блок *јачина осветљења* и превући га до дела предвиђеног за унос нумеричке вредности. Код за тастер А треба да изгледа као на слици 2.5.1.



Слика 2.5.1: Очитавање јачине светлости и приказ на екрану микробит уређаја

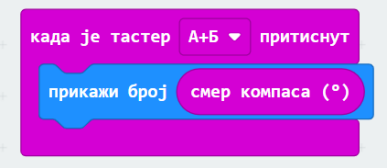
   Слично овако ће изгледати и код за тастер Б. Прво, из падајућег менија потребно је одабрати тастер Б. Како је ниво температуре такође број, потребно је одабрати блок *прикажи број*, а затим блок *температура*. Код за тастер Б треба да изгледа као на слици 2.5.2.



Слика 2.5.2: Очитавање температуре и приказ на екрану микробит уређаја

   На крају, потребно је написати код за тастер А+Б. То значи да ће се наредбе унутар овог блока извршити ако су на микробит уређају притиснута оба тастера, тј. и тастер А и

тастер Б. У симулатору ситуација је мало другачија. Наиме, како није могуће притиснути оба тастера одједном, на симулатору се приказује још једно дугме које носи назив А+Б. Код за овај тастер изгледаће исто као за претходна два дугмета, само се уместо блока за јачину светлости и температуре, превлачи блок *смер компаса*. Код је приказан на слици 2.5.3.

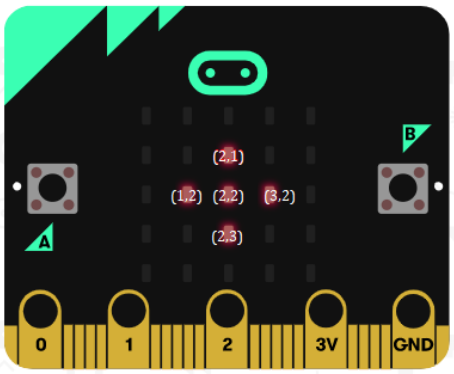


Слика 2.5.3: Очитавање смера и приказ на екрану микробит уређаја

На овом примеру, приказан је начин на који се могу очитати подаци са различитих сензора које микробит поседује. Јако је важно још једном нагласити да су вредности очитавања поменутих сензора нумеричке вредности и да се за њихов приказ мора користити блок *прикажи број*.

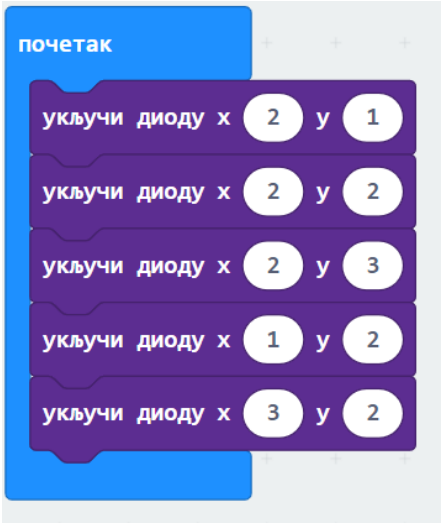
2. Написати програм којим ће се на екрану микробит уређаја приказати знак за сабирање, односно знак +.

Овај задатак помаже ученицима да схвате појам координата, као и положај тачака у координатном систему. Задатак је могуће решити применом блока *прикажи диоде* и одабиром лед диода које ће бити осветљене. Међутим, постоји још један метод за решење овог проблема, који ће бити коришћен и у наредним главама. Наиме, биће коришћена категорија лед диоде. Као што је речено у претходној глави, микробит се састоји од квадратне мреже лед диода димензија 5x5. Свака од лед диода има свој положај који се одређује уз помоћ координата. Дакле, потребно је одредити координате лед диода које требају бити укључене да би се на екрану микробит уређаја приказао знак +, као што је приказано на слици 2.5.4. Прва координата представља *x* координату, а друга *y* координату.



Слика 2.5.4: Приказ координата лед диода које треба да буду укључене

Сада, када је позната позиција лед диода, односно њихове координате, потребно је одабрати блок *укључи диоду* и унети координате лед диода. Код ће изгледати као на слици 2.5.5.

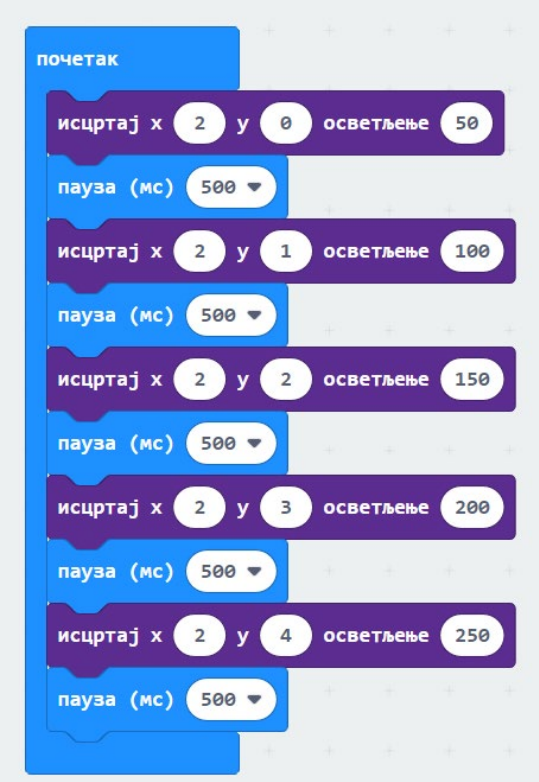


Слика 2.5.5: Унос координата лед диода

Приликом укључивања диода, могуће је подесити и ниво осветљења лед диода, који је сразмеран јачини светлости. Блок који то омогућује јесте *постави осветљење*, који пружа могућност уноса броја од 0 до 255. Ако је параметар 0, то значи да је диода искључена, а ако је параметар 255, онда је диода потпуно осветљена. Уколико корисник не употреби овај блок, лед диода ће подразумевано бити максимално осветљена.

3. Написати програм који на сваких пола секунде укључује диоде које се налазе вертикално на средини екрана микробит уређаја. Прва диода треба да има осветљеност 50, а свака наредна за 50 више.

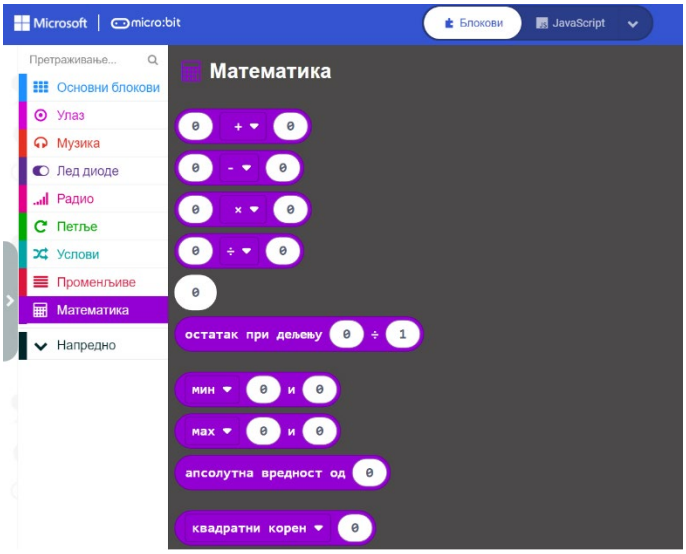
Када је микробит укључен треба да се укључују једна по једна лед диода, а да се осветљење повећава. За потребе овог задатка биће коришћен блок *исцртај* у оквиру којег је могуће подесити осветљење диоде. Са обзиром да треба да буду укључене диоде на средини по вертикали, *x* координата сваке диоде ће бити 2, док ће *y* координата узимати вредности од 0 до 4. Такође, након исцртавања сваке диоде треба бити направљена пауза од пола секунде, што се постиже блоком *пауза*. Решење овог задатка приказано је на слици 2.5.6.



Слика 2.5.6: Решење задатка 3

2.6 Аритметичке операције и промењљиве

У оквиру категорије *Математика* постоје блокови који служе за извршавање основних рачунских операција, одређивање остатка при дељењу, одабир насумичног броја, одређивање квадратног корена неког позитивног реалног броја итд.. Ови блокови представљају незаобилазан део свих компликованијих програма. Блокови којима је могуће извршавати аритметичке операције, али и друге математичке функције налазе се на слици 2.6.1.

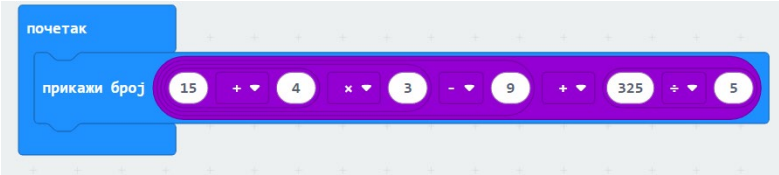


Слика 2.6.1: Блокови за аритметичке операције

У наредном задатку биће приказан начин коришћења аритметичких оператора у Мејк Код окружењу. Важно је да се ученици на једноставном примеру упознају са коришћењем основних рачунарских операција, јер ће их користити и у много комплекснијим задацима. Наредни задатак приказује употребу све четири рачунске операције и приликом решавања овог задатка ученици посебну пажњу треба да посвете редоследу слагања блокова и угнеждавању једних блокова у друге.

1. Израчунати вредности израза: $(15 + 4) \cdot 3 - 9 + 325 : 5$.

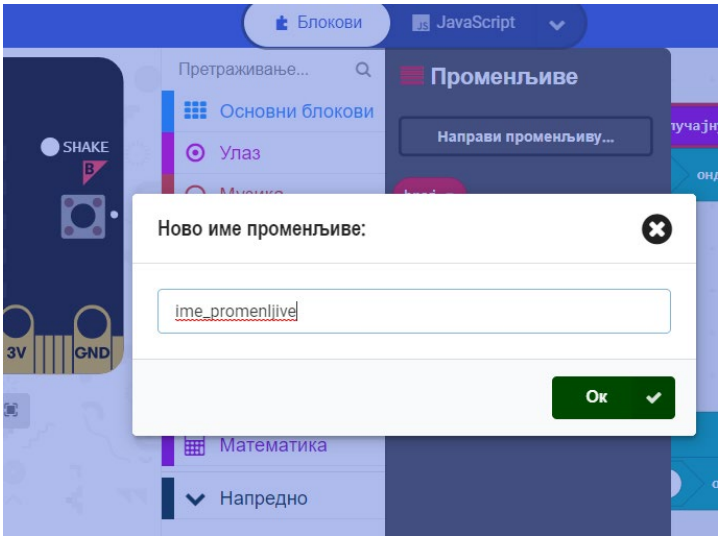
   Главни циљ решавања овог задатка јесте да ученици разумеју да је, програмирањем у Мај Код окружењу, могуће извести основне рачунске операције. Код за извршење тражених рачунских операција у Мејк Код окружењу приказано је на слици 2.6.2.



Слика 2.6.2: Примена блокова за аритметичке операције

Променљива преставља простор у меморији који служи за складиштење података и информација. Променљива има три особине: назив, тип и вредност. У зависности од врсте података који се чувају у променљивој, постоје различити типови променљиве. Вредност променљиве односи се на информацију која је сачувана у променљивој. Појам промењиве и њена имплементација у програмима је јако важна у програмирању. Ученици на једноставнијих примерима треба да уоче значај променљивих у програмирању. У уређивачу блокова, односно Мејк Код окружењу, постоји посебна категорија блокова *Променљива*, помоћу које је могуће дефинисати променљиве. Одабиром блока *Променљиве*, а након тога кликом на *Направи променљиву* отвара се прозор у коју треба унети назив променљиве, а онда кликнути на *Ок* (приказано је на слици 2.6.3). Онда када је променљива креирана, појављују се још два нова

блока унутар категорије: *постави* и *промени*. Блок *постави* даје могућност промене вредности промењљиве, док блок *промени* даје могућност мењања вредности промењљиве за неки број.

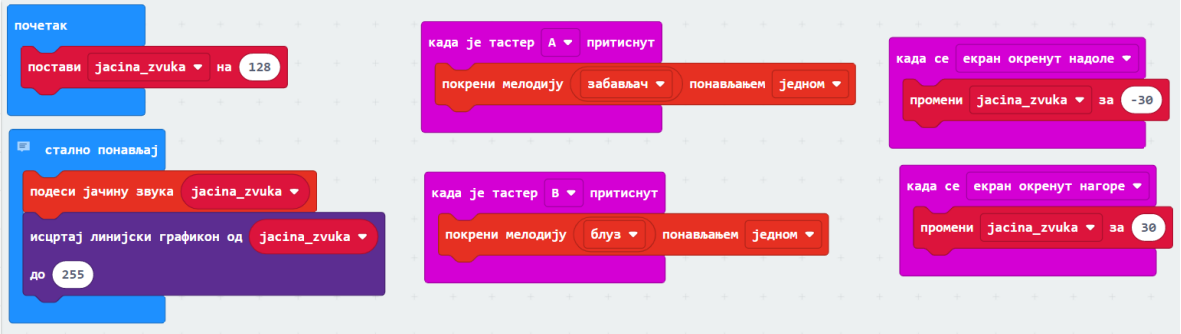


Слика 2.6.3: Задавање назива променљивој

Решавањем наредног задатка ученици треба боље да се упознају са концептом променљиве. Као што је већ речено, концепт променљивих је јако важан у програмирању и треба га што више приближити ученица. Са тим у вези, пред ученицима је постављен реалан и занимљив проблем који могу решити употребом променљивих.

2. Написати програм који симулира микробит џубокс. Притиском на тастер А или Б репродукује музику. Покретом микробит уређаја ка горе, звук се појачава, а покретом микробита на доле, звук се смањује. Притиском на тастер А+Б зауставља се репродукција музике.

   Овим задатком контролише се јачина звука на микробит уређају. Микробит треба да репродукује различите мелодије, притиском на тастере А или Б. Такође, потребно је прилагодити јачину звука нагињањем микробит уређаја ка горе или ка доле. Већ је речено да је за програме написане у Мејк Код окружењу могуће користити микробит симулатор. Међутим, ако се користи уређај микробит, онда га је потребно повезати са слушалицама или звучником, као што је приказано на слици 2.4.9 у одељку 2.4. На почетку, потребно је дефинисати променљиву *jacina_zvuka* у којој ће бити сачувана тренутна јачина звука. Приликом задавања назива променљивој, увек је добро да описује оно што Код микробит уређаја, јачина звука се креће у интервалу од 0 до 255. У овом примеру, на почетку програма ће јачина звука бити 128. Акцелерометар детектује покрет микробит уређаја. Када се уређај нагне на горе, вредност променљиве *jacina_zvuka* се повећава за 30. Одабрано је баш 30 (а не 1), зато што тада може да се чује промена у јачини звука. Када се микробит нагне ка доле, вредност променљиве *jacina_zvuka* се смањује за 30. Блок *понављај заувек* стално ажурира јачину звука микробит уређаја на основу променљиве *jacina_zvuka* која се мења услед нагињања микробит уређаја. Како би био могућ и визуелни приказ јачине звука, на екрану микробит уређаја биће приказан графикон у зависности од повећања или смањења јачине звука. За исцртавање графикона на микробит уређају, треба користити блок *исцртај линијски графикон*, који се налази у категорији Лед диоде и поставити одговарајуће параметре. Код треба да изгледа као на слици 2.6.4.



Слика 2.6.4: Код програма за задатак 2

Са обзиром да се јачина звука микробит уређаја налази у интервалу од 0 до 255, потребно је дефинисати понашање програма ако је вредност променљиве *jacina_zvuka* мања од 0 или већа од 255. За ово унапређење програма потребно је додавање наредбе гранања, о којој ће бити речи у наредном одељку. Такође, у наредном одељку ће бити приказан начин за унапређење овог програма.

2.7 Контрола тока програма – гранање и понављање у програмима

2.7.1 Одлучивање у програмима

Програми који су до сада приказани у овом раду били су линијски програми, односно програми код којих се свака наредба извршава тачно једном и у одређеном редоследу. У овом делу рада биће описани условни, односно разгранати програми. У програмима одлучивања неке наредбе ће бити извршене, а неке не, у зависности од испуњености неког услова. Дакле, за разлику од линијских програма, у програмима одлучивања неће све наредбе бити извршене у току покретања програма.

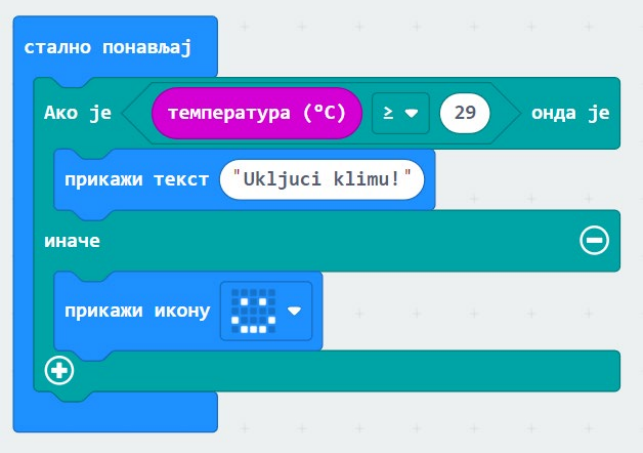
Приликом писања услова у разгранатим програмима неопходно је користити операторе поређења и логичке операторе. Оператори поређења, као што и сам назив каже, користе се за поређење величина, док логички оператори служе за комбиновање више услова.

У наставку ће бити приказани задаци који дочаравају употребу одлучивања у програмима, као и коришћење оператора. Одлучивање, односно гранање је први комплекснији концепт са којим се ученици сусрећу. Зато је јако важно да примери који осликавају овај концепт буду занимљиви и реални, како би ученицима привукли пажњу.

1. Написати програм у којем се, ако је температура већа или једнака од 29 степени исписује порука „Укључи климу!“. Ако је температура мања од 29 степени, на екрану микробита треба да буде приказана слика смајлија.

   Како би микробит уређај стално мерио температуру и у односу на вредност коју очита извршавао наредбе, потребно је одабрати блок *стално понављај*. У блок *стално понављај* треба превући блок за наредбу гранања, односно наредбу *ако-иначе*. Услов из текста задатка говори да у овом задатку треба користити оператор поређења. Решавајући овај задатак ученици треба да се упознају са наредбом гранања и операторима поређења. На овом примеру треба да разумеју програмску структуру *ако-иначе* (енгл. if-else). Већина компликованијих програма се састоји управо из ове структуре. Из категорије услов треба

одабрати оператор поређења и уклопити га у наредбу гранања. Сада, из падајућег менија треба одабрати релацију ≥. Са обзиром да треба проверити да ли је температура већа или једнака од 29, са леве стране неједнакости треба да стоји блок *температура*, док са десне стране треба да стоји број 29. Ако је услов испуњен на екрану се приказује порука „Укључи климу!“. У супротном, тј. ако услов није испуњен на екрану се приказује икона смајлија. Код треба да изгледа као на слици 2.7.1.

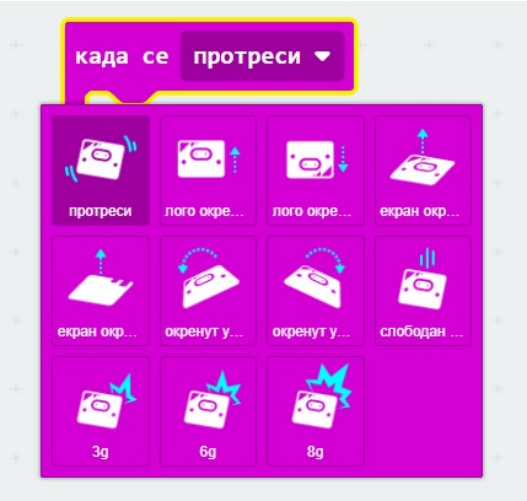


Слика 2.7.1: Решење задатка 1

2. Написати програм који ће бити симулација бацања коцкице.

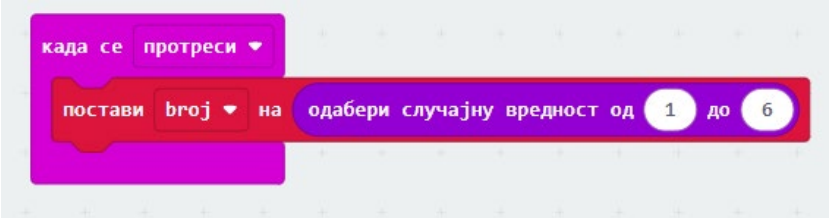
Када се играју друштвене игре, коцкица је незаобилазан реквизит. Међутим, уколико нам коцкица није при руци, могуће је направити користећи микробит. Приликом првих сусрета са програмирањем, јако је важно ученицима приближити и показати да програми које пишу имају примену у свакодневном животу. Прављење симулације коцкице за игру је програм који управо то приказује на сјајан начин.

Решавајући овај проблем ученици могу боље да савладају појам логичких оператора и услова. Када на играча дође ред да игра, потребно је баци коцкицу и погледа број који је пао. Слично тако, микробит уређај треба да реагује на неки покрет како би био приказан број, односно његова графичка репрезентација. Како би то било постигнуто, потребно је одабрати категорију *Улаз*, а након тога блок којим је могуће бирати покрет. Покрет који нам треба у овом задатку јесте покрет протреси. Дакле, наредбе унутар овог блока биће извршене само ако микробит уређај региструје покрет. Поред покрета протреси, из падајућег менија могуће је одабрати још десет врста покрета које микробит може да региструје, као што је приказано на слици 2.7.2.



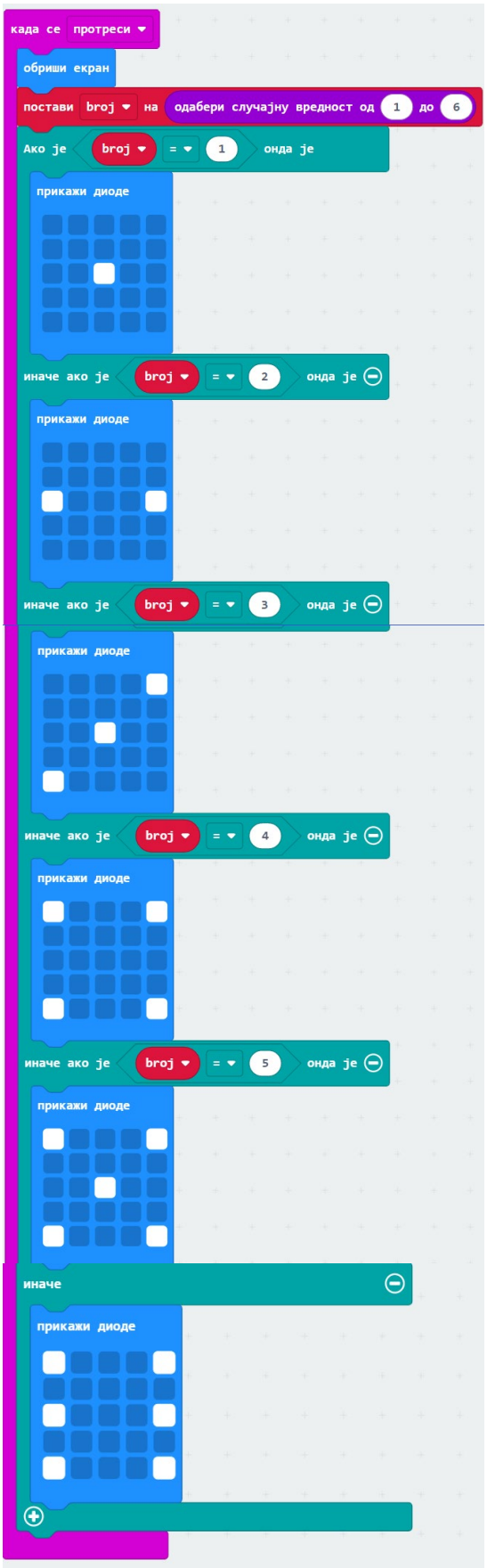
Слика 2.7.2: Падајући мени за одабир покрета

Сада, када је одабран покрет, потребно је обезбедити насумичан избор бројева од 1 до 6, укључујући и граничне бројеве. То се постиже одабиром блока *одабери случајну вредност* из категорије *Математика*. У поља предвиђена за крајње вредности интервала потребно је откуцати бројеве 1 и 6. Међутим, како би овај блок било могуће повезати, потребно је креирати променљиву у којој ће бити сачувана нумеричка вредност. За овај задатак треба искористити блок за постављање вредности променљиве у оквиру којег треба повезати блок за одабир случајног броја, као што је приказано на слици 2.7.3.



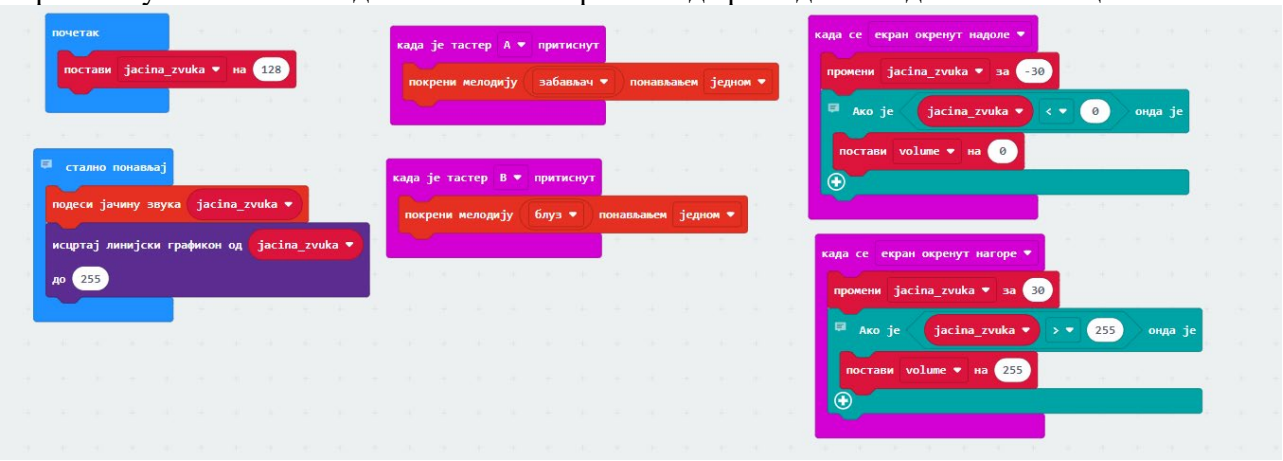
Слика 2.7.3: Блок за одабир случајне вредности броја

У зависности од броја који је изабран на случајан начин на екрану микробита треба бити приказана његова графичка репрезентација. Управо за тај део задатака треба искористити наредбу гранања. Дакле, потребно је проверити да ли је изабрани број једнак броју 1. Ако јесте, на екрану се приказује графика иста као на коцкици. У супротном, проверава се да ли је изабрани број једнак броју 2. Ако јесте, приказује се графичка репрезентација броја 2. На сличан начин, код се понавља све док се не изврше све провере. Код треба да изгледа као на слици 2.74.



Слика 2.7.4: Код за симулацију бацања коцкице

Као што је речено у претходном одељку у овом одељку биће приказан начин за унапређење програма микробит џубокс. Када микробит детектује покрет ка горе, потребно је променити вредност променљиве за 30, али и проверити да ли је вредност променљиве *jacina_zvuka* већа од 255. Ако јесте, вредност променљиве треба поставити на 255, јер је јачина звука микробит уређаја не може бити већа од 255. На сличан начин, када микробит детектује покрет ка доле, потребно је променити вредност променљиве за -30, али и проверити да ли је вредност променљиве *jacina_zvuka* мања од 0. Ако јесте, вредност променљиве треба поставити на 0, јер је јачина звука микробит уређаја не може бити мања од 0. Дакле, потребно је додати наредбе гранања у блоковима за детектовање покрета. Код треба да изгледа као на слици 2.7.5.



Слика 2.7.5: Унапређен код програма микробит џубокс

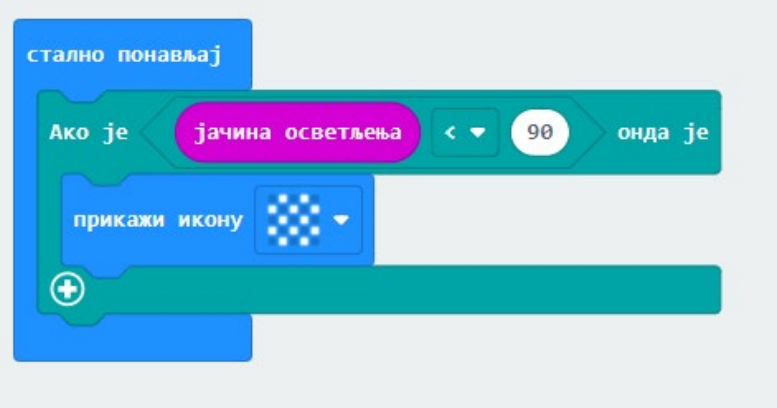
2.7.2 Понављање у програмима

У програмима се често јавља потреба да се одређене наредбе понове више пута. Петље чине програмску структуру која то омогућава. Коришћење петљи представља један од основних концепата програмирања. Програми код којих се секвенце кода понављају више пута називају се програми понављања или циклични програми. Ефекат понављања дела кода се може постићи и линијским програмима, али то није пракса у програмирању јер би тада код имао много блокова и био би непрегледан. Мејк Код окружење садржи три врсте блокова којим је могуће написати цикличне програме. У оквиру тих блокова могуће је превући друге блокове који се могу поновити: одређени (коначан) број пута, бесконачан број пута или док се не испуни неки услов.

Задаци који су приказани у наставку помажу ученицима да савладају концепт понављања у програмима. Важно је да ученици, на наредним примерима, савладају овај концепт, увиде његове предности и разумеју зашто је понављање јако важно.

1. Написати програм који приказује слику као упозорење да треба укључити светло ако је ниво осветљења мањи од 90.

   За решење овог задатка потребно је да микробит уређај мери јачину светлости све док је укључен. Ако је измерена јачина светлости мања од 90, на екрану се приказује слика чиме се сигнализира да је добро укључити извор светлости у тој просторији. Код за решење овог задатка дат је на слици 2.7.6.

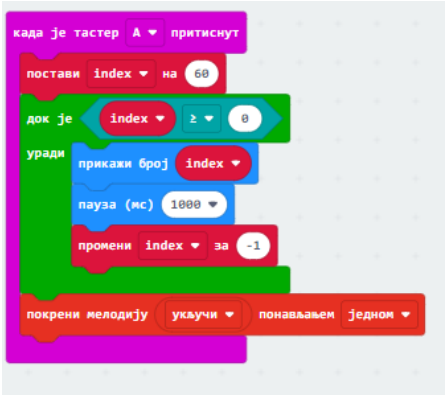


Слика 2.7.6: Решење задатка 1

2. Написати програм који представља штоперицу која може да одбројава 60, 120 или 180 секунди. Притиском на тастер А одбројава 60, на тастер Б 120, а на тастер А+Б 180 степени. Када дође до 0 репродукује се звук.

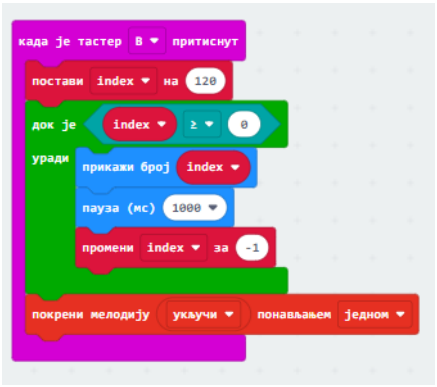
У оквиру овог задатка ученици треба да увиде употребу петље код које се услов проверава на почетку. Дакле, неке наредбе ће се понављати све док је испуњен услов. Ова врста петље, односно понављања се користи у многим програмима, али и у текстуалном програмирању. Зато је јако важно да се ученици за почетак уз помоћ блокова упознају са овим начином размишљања. Штоперица је алат који се користи и на часовима других предмета, на пример на часовима физичког васпитања. Приликом учења основних концепата програмирања, јако је важно да ученици виде продукт свог рада, као и да могу да га употребе у свакодневном животу и на другим часовима.

За почетак, потребно је размислити о проблему и направити схему која ће представљати решење овог проблема. Потребно је да се на екрану приказују бројеви који представљају одбројавање у секундама. Када истекне време, односно када је на екрану приказана 0, треба бити репродукована музика. У зависности од тога које је дугме притиснуто, микробит треба да одбројава 60, 120 или 180 секунди. Дакле, потребно је да постоји једна променљива која ће представљати време у секундама и која ће се смањивати до нуле. Јако је важно напоменути да између појављивања два суседна броја мора проћи једна секунда, како би одбројавање било у реалном времену. Зато је потребно направити паузу од једне секунде између појављивања бројева на екрану микроби уређаја. На почетку, њена вредност биће 60, 120 или 180 у зависности од тога које је дугме притиснуто. Када се променљивој додели вредност 0, тада се на микробит уређају репродукује музика. Вредност променљиве ће се смањивати све док је она већа или једнака од нуле. Дакле, потребно је користити блок „*док је - уради*“. Код за притисак тастера А приказан је на слици 2.7.7.

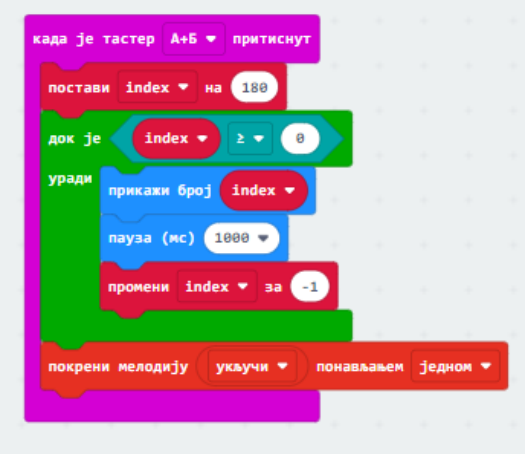


Слика 2.7.7: Код за одбројавање 60 секунди

Код за дугме Б и А+Б је јако сличан претходном. Разлика је у томе што променљива индекс на почетку има вредност 120 односно 180.



Слика 2.7.8: Код за одбројавање 120 секунди

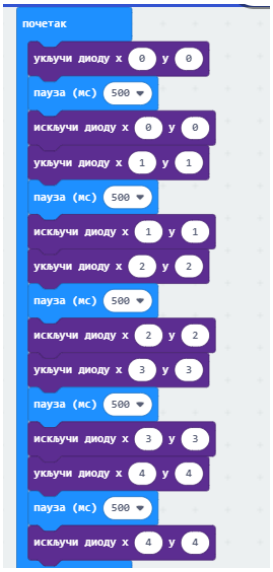


Слика 2.7.9: Код за одбројавање 180 секунди

3. Написати програм којим се, са паузом од пола секунде, укључује и искључује једна по једна диода на дијагонали екрана микробит уређаја.

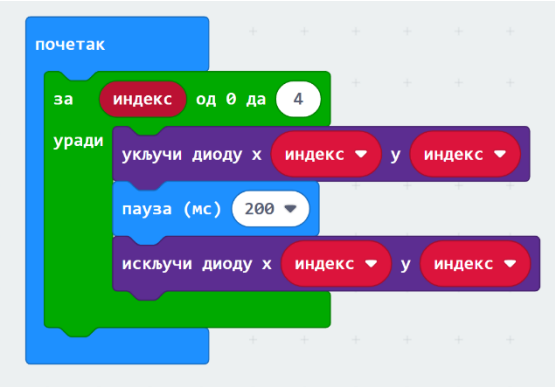
Екран микробит уређаја садржи две дијагонале. У овом задатку ће се укључивати и искључивати лед диоде које повезују диоду са координатом (0,0) и диоду са координатом (4,4). Посматрајући микробит уређај и координате лед диода, може се закључити да су *x* и *y* координата лед диода на дијагонали микробит уређаја једнаке. Овај задатак могуће је урадити низањем блокова који представљају укључивање одговарајуће диоде, паузу и

искључивање исте диоде. Овако урађен програм имао би петнаест наредби, које се налазе једна испод друге, као на слици 2.7.10.



Слика 2.7.10: Решење задатка без коришћења for петље

Постоји и други начин за решавање овог задатка, а то је коришћење понављања у програмима, односно петље. Наиме, са обзиром да су *x* и *y* координата сваке лед диоде која се налази на дијагонали екрана микробита једнаке, за решење овог задатка се може употребити блок за *for* петљу, односно блок „*за-уради*“. Како *x* и *y* координата лед диоде може бити најмање 0, а највише 4, потребно је поставити да променљива која носи назив *индекс* узима вредности од 0 до 4. Наредбе које се налазе у оквиру овог блока извршиће се пет пута. Сада, потребно је да се укључи лед диода чије су и *x* и *y* координата једнаке са променљивом индекс. Након тога потребно је направити паузу од пола секунде, па искључити диоду са координата једнаким са променљивом индекс. Дакле, у првом кораку, вредност променљиве *индекс* је једнака нули, па се укључује диода са координатама (0,0), прави се пауза и искључује та диода. У другом кораку вредност променљиве *индекс* је један, па се укључује лед диода са координатом (1,1), прави пауза и искључује та лед диода итд. Код написан на овај начин (приказан на слици 2.7.11) има пет наредби које се извршавају и запис је једноставнији и прегледнији у односу на прво решење. У овом примеру, задатак је било могуће урадити на два начина. Међутим, није увек тако. За неке задатке неопходно је коришћење блокова понављања, зато је важно да ученици на основним примерима виде предности коришћења петљи. Дакле, блокови понављања омогућавају кориснику да програми буду прегледнији и имају мање линија кода. Коришћењем петљи у програмима, ученици основних школа подстичу се на логичко размишљање, као и решавање проблема на најкраћи начин.



Слика 2.7.11: Решење задатка коришћењем for петље

Више занимљивих идеја и примера из ове области могу се пронаћи на адреси https://microbit.org.

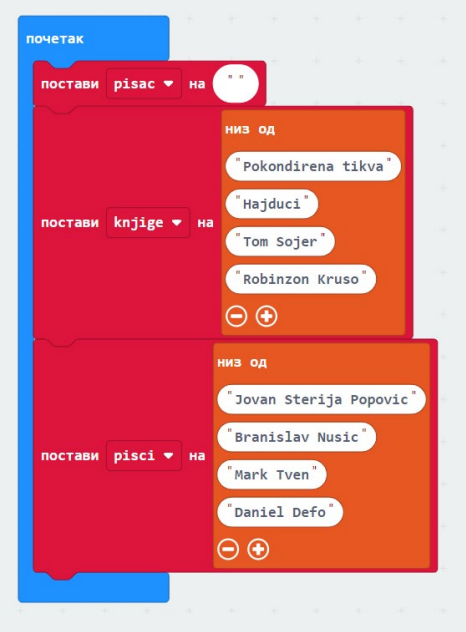
2.8 Низови

Низови представљају колекцију података у којем је могуће сачувати више вредности истовремено. У низовима могу бити сачувани бројеви, ниске или музичке ноте који се називају елементи низа. Сваки елемент низа има своју позицију, односно индекс. Помоћу индекса могуће је приступити сваком елементу низа појединачно. Први елемент низа има *индекс* 0, док последњи елемент има позицију за једну мању од броја елемената низа. У Мејк Код окружењу за креирање низова и листе постоји посебна категорија Низови, која се налази у оквиру категорије Напредно.

1. Написати програм који за, на случајан начин, одабран роман приказује писца који је написао тај роман.

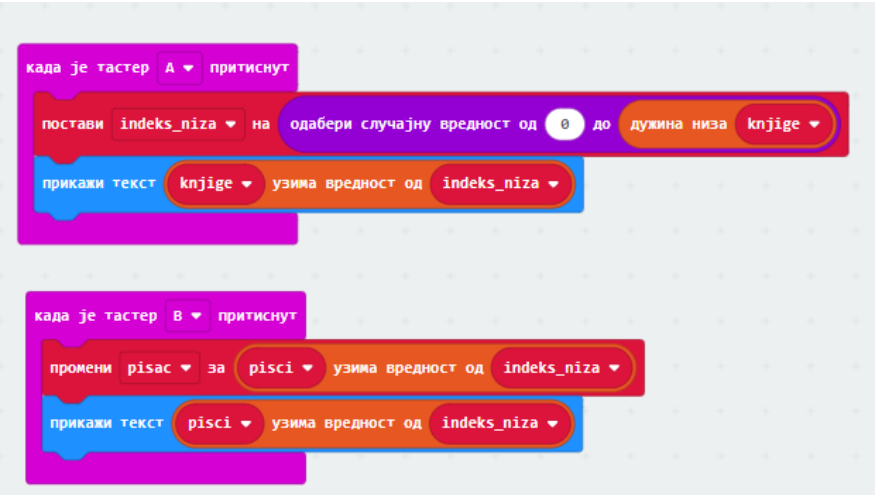
   На овом примеру ученици треба да се боље упознају са низовима и разумеју да сваки члан низа има своју позицију, односно индекс. Кликом на дугме А приказује се назив књиге, док се кликом на дугме Б приказује писац. За потребе овог примера биће коришћене књиге „Покондирена тиква“, „Хајдуци“, „Том Сојер“ и „Ромбинзон Крусо“. Писци су редом Јован Стерија Поповић, Бранислав Нушић, Марк Твен и Данијел Дефо. Дакле, потребно је креирати два низа: један ће представљати називе књига, а други имена и презиме писаца. Кликом на дугме А потребно је насумично одабрати књигу и приказати њено име на екрану. Елементу низа могуће је приступити само путем његове позиције у низу, односно путем индекса. Дакле, за потребе овог задатка треба користити и насумични одабир броја. Такође, потребно је креирати и нову променљиву писац ради одабира правог писца за одабрану књигу.

   Приликом покретања микробит уређаја, потребно је креирати променљиву и два низа, као што је приказано на слици 2.8.1.



Слика 2.8.1: Креирање низова

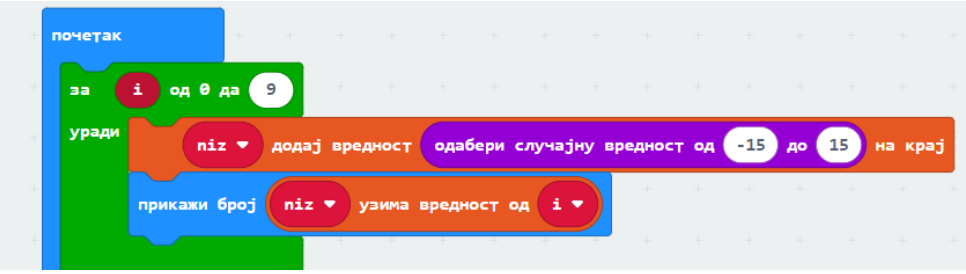
Члановима низа могуће је приступити само путем њихове позиције у низу, односно путем блока *узима вредност од* који се налази у категорији Низови. Приликом коришћења овог блока потребно је навести низ којем се приступа, као и индекс, односно позицију елемента у низу. Дакле, притиском тастера А потребно је одабрати насумичан број из интервала од 0 до дужине низа и приказати елемент низа чији је индекс тај изабрани број. Затим, притиском на тастер Б потребно је вредност промењљиве писац променити тако да буде одговарајући елемент низа писци. Након тога, потребно је приказати име и презиме писца на екрану. Кодови за дугме А и Б приказани су на слици 2.8.2.



Слика 2.8.2: Одабир насумичног броја и приказ елемната низа на екрану

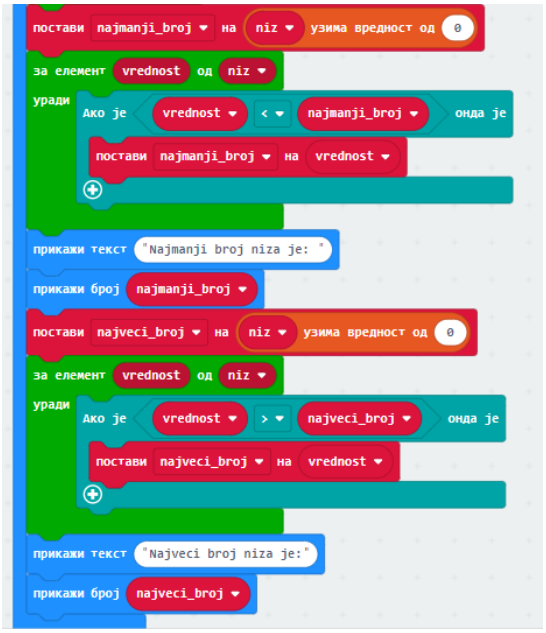
2. Написати програм којим се креира низ који има 10 елемената, чији су елементи насумични бројеви из интервала [-15;15], а затим одредити најмањи и највећи елемент низа.

Први део овог задатка представља креирање низа од 10 елемената, чији су елементи насумични бројеви од -15 до 15. Најбољи начин за решавање овог дела задатка јесте коришћење петље. Потребно је користити блок код којег се наредбе понављају одређени број пута. Са обзиром да низ треба да има 10 елемената, променљива у оквиру овог блока ће узимати вредности од 0 до 9. У оквиру овог блока потребно је превући блок којим се додају елементи низа на крај. Сваки елемент низа је насумичан број од -15 до 15. Након сваког додатог елемента, потребно је да буде приказан на екрану. Код за овај део програма дат је на слици 2.8.3.



Слика 2.8.3: Додавање елемента у низ

Сада је потребно одредити најмањи елемент низа. Променљивој која представља најмањи елемент низа је потребно доделити вредност првог елемента низа. Онда је потребно проћи кроз сваки елемент низа и проверити да ли је мањи од првог. Ако јесте онда променљива предвиђена за најмањи елемент узима вредност тог члана низа. За овај сегмент задатка је потребно поново користити петљу. Када се изврше наредбе унутар петље, потребно је на екрану приказати најмањи елемент. На сличан начин се може одредити највећи елемент низа, као што је приказано на слици 2.8.4.



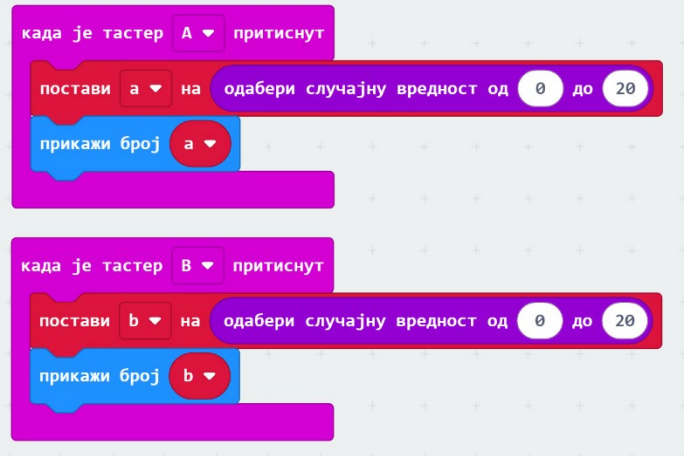
Слика 2.8.4: Одређивање најмањег елемента низа

2.9 Функције

Како би програми били разумљивији, програмери често користе функције. Уколико се одређене наредбе понављају више пута у програму јако је корисно тај део програма издвојити као посебну целину, односно функцију. Функција представља скуп наредби која улазне параметре трансформише у податак који се назива излазна вредност функције или резултат функције. Главна предност коришћења функција јесте избегавање понављања дела кода, а самим тим и код постаје разумљивији. Основни елементи функција су: назив функције, улазни параметри, тело функције и излазни резултат. У програму функцију позивамо наводећи име функције и задавањем параметара. Поред блокова у категорији *Математика* помоћу којих је могуће користити математичке функције као што су квадратни корен, апсолутна вредности, минимум и максимум, у Мејк Код окружењу функцију је могуће креирати одабиром блока *Функције*, који се налази у оквиру категорије *Напредно*. Кликом на дугме направи функцију, потребно је написати име функције и одабрати улазне параметре. Улазни параметри могу бити различитог типа: број, текст, низ, слика. Након тога, у оквиру креиране функције потребно је превући блокове који ће бити извршени приликом позива функције у програму.

1. Написати функцију која рачуна производ два броја. Кликом на дугме А насумично се бира први чинилац из интервала од 0 до 20, док се кликом на дугме Б насумично бира други чинилац из истог интервала. Кликом на дугме А+Б рачуна се производ два изабрана броја.

   На овом примеру ученици треба да разумеју појам функције, улазних параметара као и излазни резултат. Кликом на дугме А потребно је креирати нову променљиву која ће имати вредност насумичног броја из интервала од 0 до 20, а након тога је приказати на екрану микробита. Кликом на дугме Б креира се друга променљива која ће такође имати вредности насумичног броја од 0 до 20. Код за дугме А и Б приказан је на слици 2.9.1.

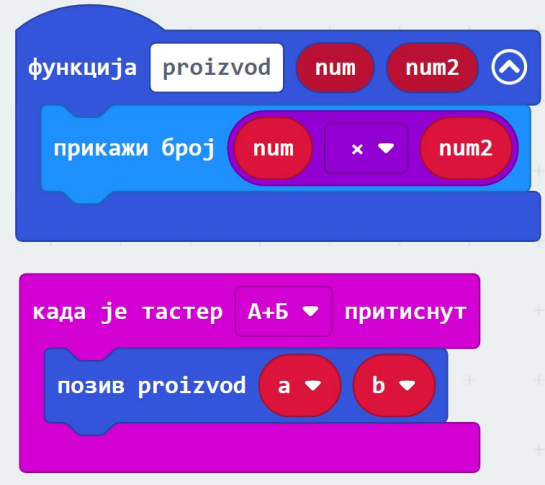


Слика 2.9.1: Задавање улазних параметара

   Након овог дела, потребно је креирати функцију чији су улазни параметри бројеви а и б који су креирани кликом на дугме А и Б. Излазни параметар је производ ова два броја. Важно је напоменути да приликом дефинисања функције, улазни параметри не морају носити назив променљивих који престављају чиниоце. Називе променљивих чији производ је потребно израчунати треба да буду наведени приликом позивања функције, односно притиском на тастер А и Б. У овом случају називи улазних параметара су num и num2. Тело

функције има једну наредбу, односно један блок у којем се приказује број који представља производ улазних параметара.

На притисак дугмета А+Б позива се функција и као улазни параметри наводе се промењљиве добијене кликом на дугме А и дугме Б, као што је приказано на слици 2.9.2.



Слика 2.9.2: Креирање функције

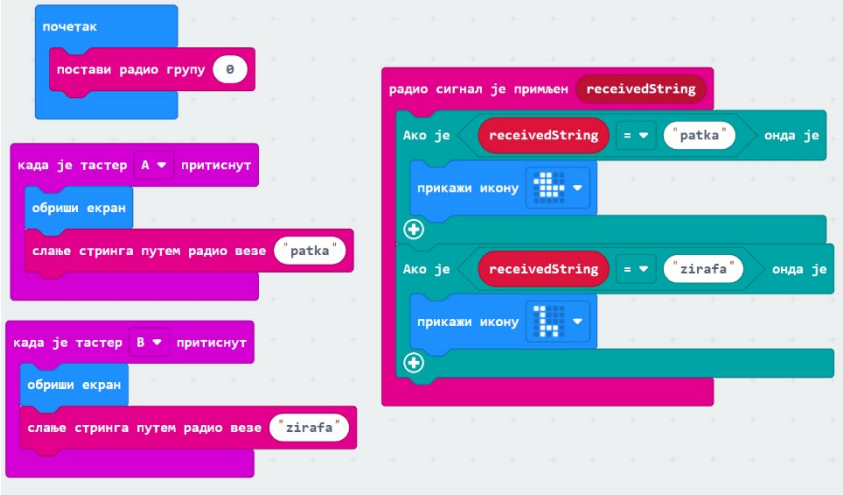
2.10 Радио веза и блутут

Процесор микробит уређаја има радио модул који омогућава бежично слање и примање порука, односно комуникацију са другим микробит уређајима. Поруке могу бити послате и примљене бежично до удаљености око 70 метара. Како би различити пројекти и програми били реализовани неопходно је да микробит уређаји међусобно комуницирају. У Мејк Код окружењу то је могуће урадити коришћењем блокова из категорије *Радио*. Блок постави радио омогућава повезивање микробит уређаја на виртуелну групу и тиме омогућава повезивање са другим микробит уређајима који припадају истој виртуелној групи. У блоку постави радио групу могуће је уписати број од 0 до 255, који представља ИД групе. Микробит уређај може бити члан само једне групе одједном. Микробит уређаји могу комуницирати и са другим уређајима, што омогућава коришћење уређаја на различите и креативне начине. Да би микробит уређај био повезан са Андроид телефоном потребно је да на том телефону буде инсталирана апликација.

1. Написати програм који притиском на тастер А једног микробит уређаја на другом приказује слику патке, док се притиском на тастер Б приказује слика жирафе.

Комуникација између микробит уређаја чини да програми буду занимљивији. На првом примеру ученици треба да се упознају са блоковима за радио комуникацију и разумеју на који начин микробит уређаји комуницирају. На самом почетку програма потребно је поставити ИД за комуникацију и то може бити било који број. У овом примеру комуникација ће бити успостављена слањем текста, односно стринга другом микробит уређају. Ако је притиснут тастер А, биће послат стринг патка, а уколико је притиснут тастер Б биће послат стринг жирафа. Онда када други микробит прими стринг, потребно је употребити наредбу гранања. Ако је примљен стринг патка, онда се приказује слика патке, а ако је примљен стринг жирафа, приказује се слика жирафе. Овај пример представља најједноставнији пример употребе радио комуникације. Међутим, иако пример делује једноставно, на добар

начин приказује ученицима употребу *Радио* категорије као и пут којим уређаји комуницирају. Решење овог задатка је приказан на слици 2.10.1.



Слика 2.10.1: Комуникација између два микробит уређаја

Како би програм био покренут на микробит уређајима потребно је преузети програм и пребацити га на оба уређаја.

Више примера и појашњења везана за програмирање микробит уређаја у Мејк Код окружење могу се пронаћи у књизи „*Getting started with the micro:bit*“ аутора Wolfram Donata.

2.11 Игре

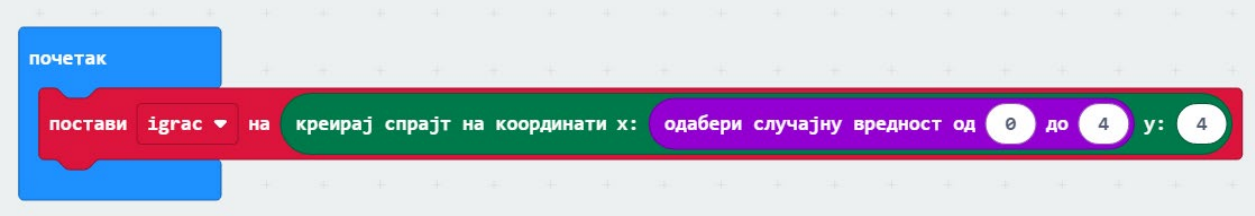
У Мејк Код окружењу могуће је, слагањем блокова направити игрицу за микробит уређај. У оквиру категорије *Напредно*, налази се блок *Игре* у коме се налазе блокови за креирање игрица. За потребе игрице потребно је креирати лед диоду или која се може кретати по екрану микробит уређаја. Та лед диода назива се спрајт. Лед диода може да се креће десно, лево или да буде ротирана за одређени угао. Приликом креирања игре лед диоде се понашају као графички кориснички интерфејс конзоле за видео игре. Блокови категорије *Игре* јесу зелене боје.

У наредним задацима биће приказано како се, у Мејк Код окружењу, могу креирати игре за микробит уређај. Задаци у оквиру којих треба направити игру могу се користити као пројектни задаци, јер сваки од задатака захтева рашчлањивање задатака на више мањих целина и проналажење решења за сваку од тих целина. Креирајући игре, ученици пролазе кроз све концепте програмирања о којима је било речи у овом раду. Креирање игара ученицима представља изазов јер их подстиче на размишљање, унапређивање програма и на крају, могу да виде и испробају продукт свог рада.

1. Написати програм који представља игру „Избегни препреке“ у којој корисник треба да избегне препреке које се појављују на екрану микробит уређаја. Кликом на тастер А корисник се креће улево, док се кликом на тастер Б креће удесно. Кликом на А+Б започиње се игра од почетка. Игра се завршава онда када корисник не успе да избегне препреку и приказује се резултат.

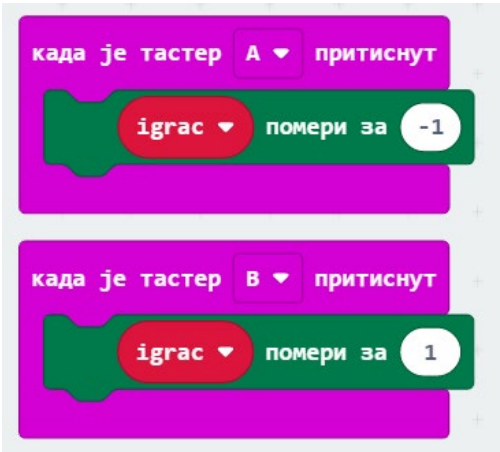
За потребе креирања игрице, биће потребно користити блокове из категорије *Игре*, као и блокове за понављања, услове и креирање променљивих. Да би игрица била креирана потребно је познавати све блокове о којима је било речи у овом раду.

Прво што је потребно урадити јесте направити једну променљиву која ће представљати играча. Онда треба креирати спрајт односно једну лед диоду којој ће бити задате координате. Спрат се креира превлачењем блока креирај спрат и куцањем координата, као што је приказано на слици 2.11.1. Када се игра покрене, играч треба да буде позициониран на последњој врсти, што значи да ће *x* координата бити случајан број који се налази у интервали од 0до 4, док ће *y* координата бити 4. Са обзиром да је играч треба да се појави на почетку игре, овај блок превлачи се и уклапа са блоком почетак.



Слика 2.11.1: Креирање спрајта

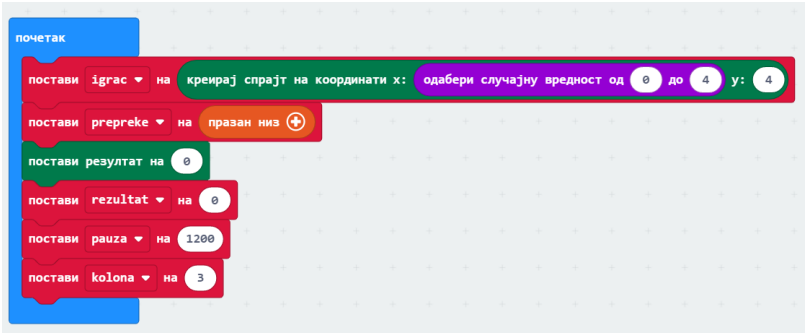
Сада, треба омогућити да се притиском на тастере А и Б играч креће. Кликом на тастер А играч треба да се помери улево за једно место, док се кликом на тастер Б играч помера за једно место удесно. Овај ефекат се може постићи блоком помери у оквиру категорије Игре. Када је притиснут тастер А *x* координата се смањи за 1, а када се притисне тастер Б *x* координата се повећа за 1. Кодови за тастер А и Б приказани су на слици 2.11.2.



Слика 2.11.2: Кретање лед диода по екрану микробит уређаја

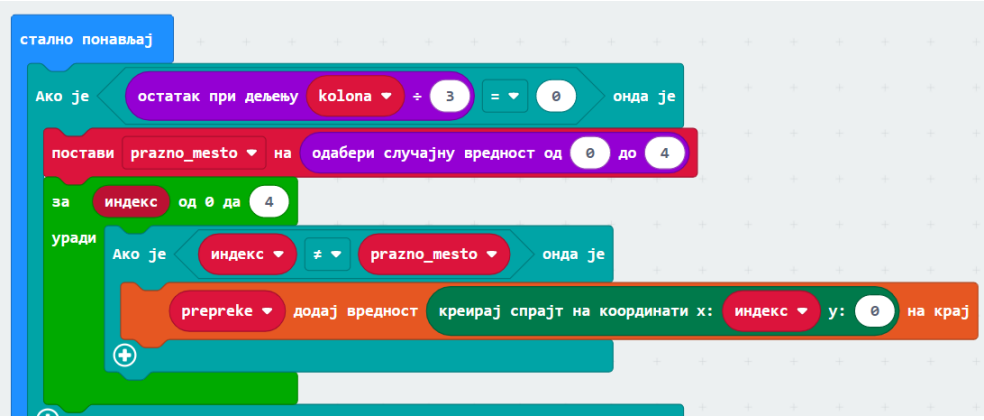
Сада, када је омогућено кретање играча, потребно је направити препреке и омогућити да се препреке приказују на екрану микробит уређаја. Ово је уједно и најтежи део овог задатка. Препреке се приказују као врсте у којима једна, насумично изабрана, лед диода не светли. За реализацију овог дела задатка потребно је креирати празан низ. Како је у задатку потребно приказати резултат, као и да се препреке брже крећу након сваког поена, потребно је декларисати још две променљиве: резултат и пауза. Променљива резултат има вредност 0, док променљива пауза има вредност 1200, што ће представљати време у милисекундима.

Дакле, у оквиру блока почетак биће повезана четири блока, као то је приказано на слици 2.11.3.



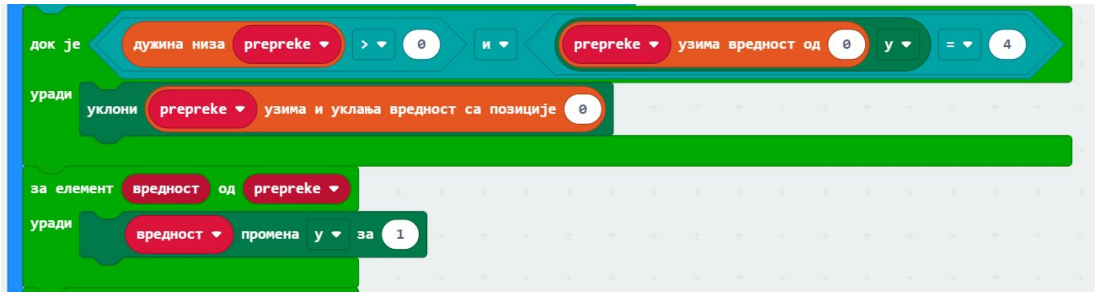
Слика 2.11.3: Блокови за почетак програма

Сада, треба користити блок стално понављај, у оквиру којег ће бити приказане препреке и њихово кретање, као и крај игре. Како препреке не би биле близу једна друге и како би играч имао довољно времена да избегне препреке, препреке ће се појављивати у сваком трећем реду. Потребно је направити нову променљиву која ће имати назив колона. Ако је број колоне непаран број, онда ће се приказати препрека. Прво, потребно је креирати празно место у препреци, тј. креирати нову променљиву која се зове препрека која ће узимати насумичну вредност од 0 до 4. Сада, потребно је креирати препреку која ће имати четири диоде које светле и једну која је угашена. Онда, креирану препреку, односно светлеће диоде, треба додати као елементе низа, као што је приказано на слици 2.11.4.



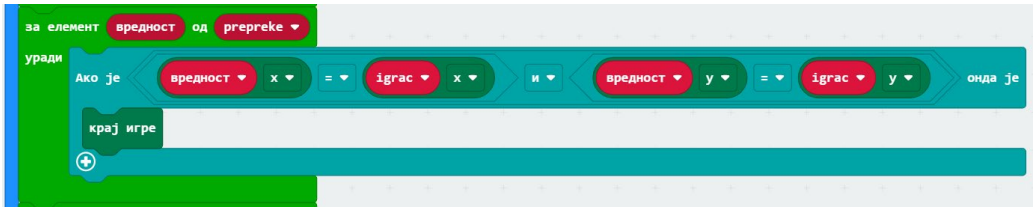
Слика 2.11.4: Додавање елемената низа

Када, дође до краја екрана микробит уређаја, препрека треба да нестане. То се може постићи провером два услова: да ли је дужина низа већа од нуле и да ли је *y* координата сваког члана низа једнака 4. Такође, треба постићи и кретање препрека на микробит уређају, што се може постићи коришћењем петље, као што је приказано на слици 2.11.5.



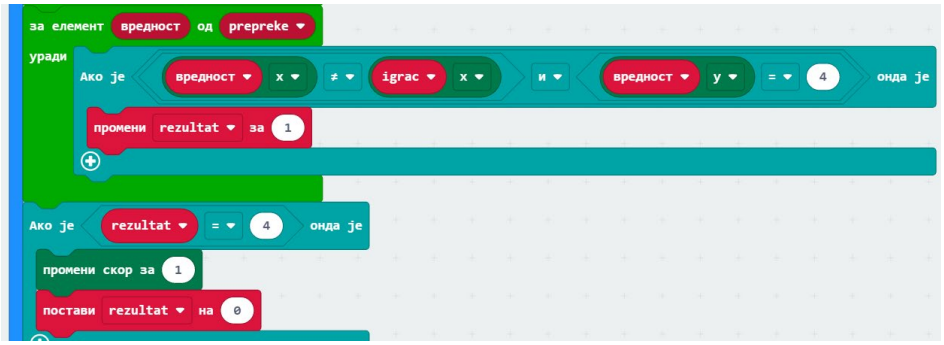
Слика 2.11.5: Кретање препрека

Крај игре јесте када се *x* и y координате играча и препреке (бар једне светлеће лед диоде) поклапају. Зато треба проћи кроз сваки елемент низа и проверити да ли се *x* и y координате поклапају, као што је приказано на слици 2.11.6.



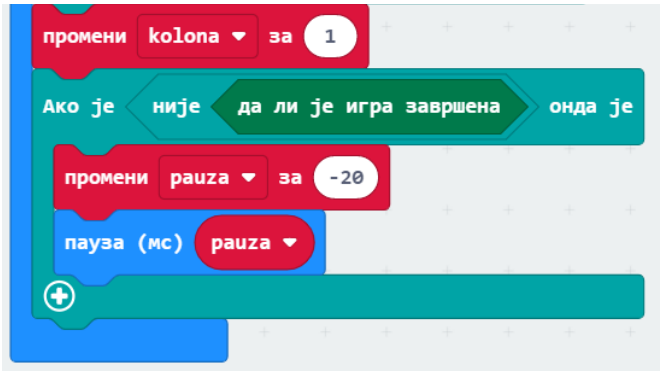
Слика 2.11.6: Дефинисање краја игре

Сада, треба израчунати резултат, односно број проласка играча кроз препреку. У оквиру категорије Игре постоји већ уграђена променљива која се зове скор. Међутим, за потребе ове игрице биће потребно увести још једну променљиву која ће служити за одређивање резултата. Да би резултат био постигнут играч треба да задовољи два услова: да ли је *x* координата играча различита од *x* координате препреке и да ли је *y* координата препрека једнака 4 (то значи да је препрека дошла до последње врсте). Међутим, како се ови услови проверавају за сваки члан низа тј. сваку светлећу диоду препреке, резултат уграђене променљиве скор би био 4 уместо 1. Зато је неопходно увести још једну променљиву која се зове резултат. Ако су испуњени ови услови, вредност променљиве резултат се повећава за 1. Сада, треба проверити да ли је вредност променљиве резултат 4. Ако јесте, скор повећавамо за 1, а вредност променљиве резултат враћамо на 0. Овај део задатка приказан је на слици 2.11.7.



Слика 2.11.7: Повећавање резултата

На крају, треба променити вредност променљиве колоне за један и постићи да се препреке убрзавају након сваког поена. Овај део кода приказан је на слици 2.11.8. Убрзање препреке се у ствари постиже смањивањем паузе између појављивања две препреке. Ако игра није завршена, онда се вредност променљиве пауза смањује за 20, а вредност паузе између појављивања две препреке има вредност променљиве пауза, који представљају време у мили секундама.

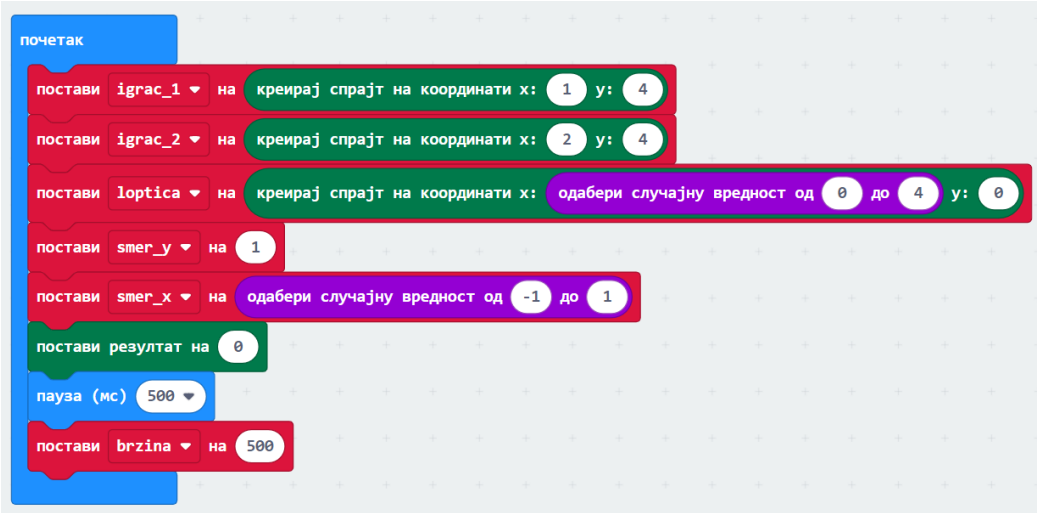


Слика 2.11.8: Промена вредности променљиве и подешавање паузе

2. Написати програм који представља игру у којој корисник треба што дуже да одржи „лоптицу“ у игри. Лоптица је представљена као једна светлећа диода која мења своје место. Корисник се креће тако што помера микробит уређај надесно или налево.

У овој игрици играча представљају две светлеће диоде које се налазе једна поред друге на последњој врсти екрана микробит уређаја. Играч треба да „ухвати“ лоптицу и не дозволи да падне. Након сваког поена, брзина лопте постаје већа.

У блоку *почетак*, потребно је креирати променљиве играч1 и играч2, лоптица, смер_x, смер_y, резултат и брзина. Затим је потребно за променљиве играч1 и играч2 креирати спрајт са координатама *x* и *y*. Смер за y осу треба бити постављен на 1, а смер за *x* осу је насумичан број из интервала од -1 до 1. Код за блок *почетак* приказан је на слици 2.11.9.



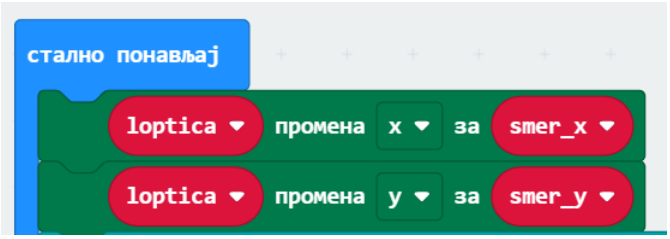
Слика 2.11.9: Почетак програма

Сада, треба постићи кретање играча покретима на десно или на лево. На препознавање покрета улево, променљиве играч1 и играч2 треба да се помере за једно место улево. На препознавање покрета удесно, играч 1 и играч 2 треба да се помере за једно место удесно. Међутим, покрет улево је могуће реализовати само ако је *x* координата играча 1 већа од нуле, док је покрет удесно могуће направити само ако је *x* координата играча 2 мања од 4. Код за кретање играча приказан је на слици 2.11.10.

Слика 2.11.10: Кретање играча

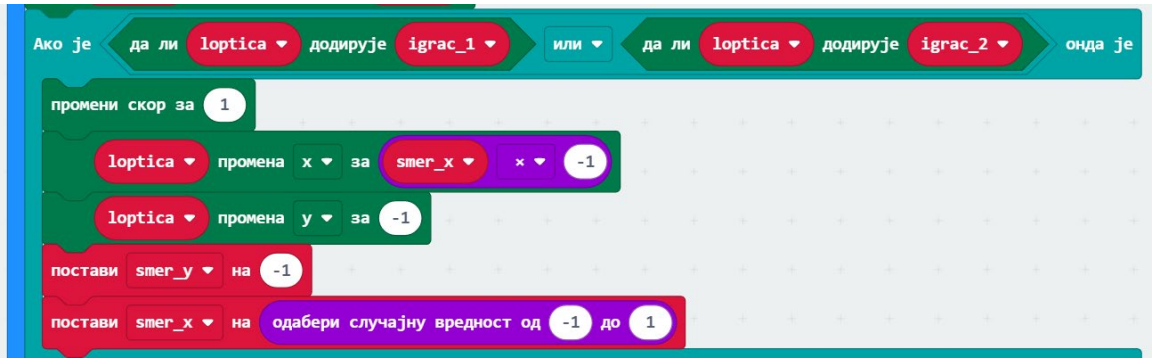
У оквиру блока *стално понављај* треба реализовати кретање лоптице, одбијање лоптице о играча, рачунање резултата, као и одређивање краја игре.

Прво, треба постићи кретање лоптице по екрану микробит уређаја. То је могуће постићи променом *x* и *y* координате спрајта који представља лоптицу, као што је приказано на слици 2.11.11.



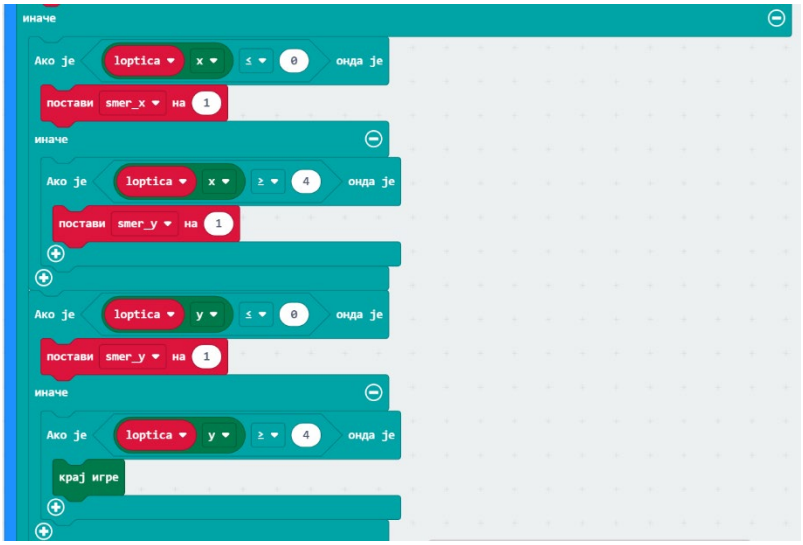
Слика 2.11.11: Кретање лоптице по екрану

Сада, треба проверити да ли је играч постигао поен тј. ухватио лоптицу. Ако играч додирује лоптицу, онда треба променити смер лоптице, тако да *x* координата лоптице буде супротна од оне која је била пре додира са играчем, а *y* координата се мења за -1. Такође, потребно је променити и смер тако да смер *y* осе постаје -1, а смер *x* осе постаје насумичан број од -1 до 1. Овим се постиже да лоптица насумично мења смер, тако да се никада не креће по истој путањи. Такође скор треба повећати за 1. Овај део кода приказан је на слици 2.11.12.



Слика 2.11.12: Провера да ли је играч постигао поен

Ако лоптица није додирнула играча, треба да се креће по екрану микробит уређаја. Ако је *x* координата лоптице мања или једнака од 0, смер кретања у правцу *x* осе постаје 1, а ако је *x* координата лоптице већа или једнака од 4, смер кретања постаје -1. Ако је *y* координата лоптице мања или једнака од 0, смер кретања у правцу у осе постаје 1, а ако је *y* координата лоптице већа или једнака од 4, тада је крај игре, јер значи да играч није стигао да покупи лоптицу. Код је приказан на слици 2.11.13.



Слика 2.11.13: Кретање лоптице

На крају, потребно је омогућити да се брзина лоптице повећава након сваког поена. То је могуће постићи на сличан начин као у претходном примеру, као што је приказано на слици 2.11.14.

Слика 2.11.14: Промена брзине лоптице

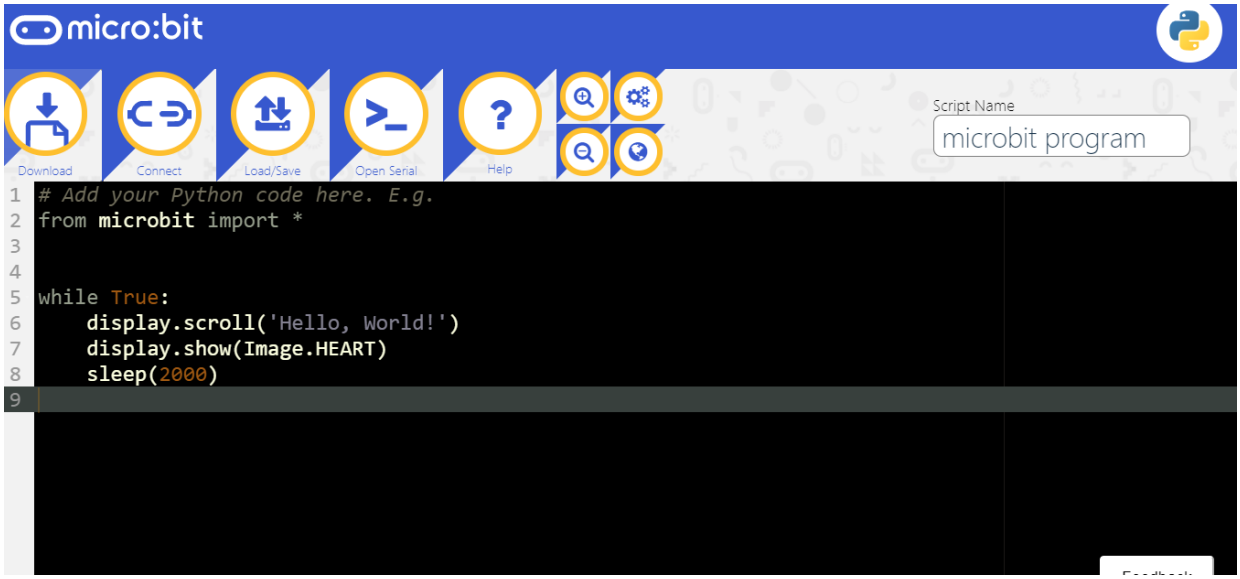
Више занимљивих идеја и задатака је могуће пронаћи у приручнику „ *Програмирање микробит уређаја у МејкКоду – приручник за пети разред*“ који се налази на веб-адреси https://petlja.org/biblioteka/r/kursevi/microbitbc#id2.

3 Микропајтон и Мју окружење

Ученици се са текстуалним програмирањем сусрећу у шестом разреду основне школе учећи програмски језик Пајтон (енгл. Python). Многи елементи који се користе у блоковском програмирању, користе се и у текстуалном програмирању, само је запис другачији. У Мејк Код окружењу је довољно превући блокове наредби на одговарајуће место, док је за програмирање у текстуалном програмском језику потребно савладати синтаксу, јер се наредбе пишу у специфичној форми енглеског језика. Програмски језик Пајтон је текстуални програмски језик чије наредбе представљају комбинацију симбола, речи и бројева. Микропајтон је верзија програмског језика Пајтон, али има мање функционалности. Међутим, синтакса Микропајтона је потпуно идентична као и синтакса програмског језика Пајтон у деловима у којима се поклапају.

Програмирање у Микропајтону може се извршити на два начина: онлајн или путем Мју едитора.

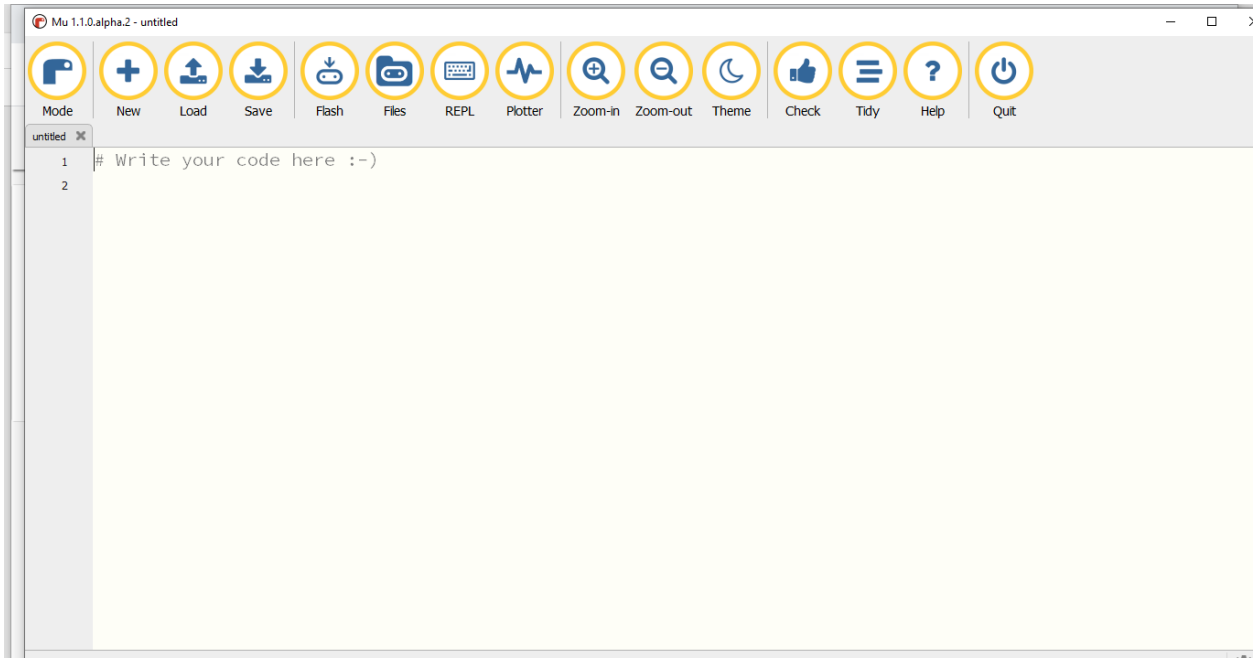
Уколико се програмирање у Микропајтону врши путем претраживача, потребно је отићи на адресу: https://microbit.org/code/, а затим одабрати Пајтон едитор. Тада ће се приказати прозор као на слици 3.1:



Слика 2.11.1: Онлајн Мју окружење

Међутим, програмирање у Микропајтону се може вршити и на рачунару, инсталирањем Мју едитора. Инсталацију Мју едитора могуће је пронаћи на адреси: https://codewith.mu/en/download. Потребно је одабрати оперативни систем и након тога преузети апликацију. Преузети фајл ће највероватније бити у фолдеру „Преузимања“. Након покретања преузете датотеке потребно је пратити кораке за инсталацију Мју окружења.

На самом покретању Мју апликације потребно је припремити окружење на рад са микробитом. То се може урадити кликом на Mode, а затим одабиром „BBC microbit“ мода. Мју окружење је веома прегледно и једноставно за рад и приказано је на слици 3.2.



Слика 2.11.2: Mју апликација

Главни мени састоји се из следећих ставки: *Режим* (енгл. *Mode*), *Ново* (енгл. *New*), *Учитај* (енгл. *Load*), *Сачувај* (енгл. *Save*), *Flash*, *Датотеке* (енгл. *Files*), *REPL*, *Plotter*, *Увећај* (енгл. *Zoom-In*), *Умањи* (енгл. *Zoom-Out*), *Тема* (енгл. *Theme*), *Провери* (енгл. *Check*), *Tidy*, *Помоћ* (енгл. *Help*) и *Излаз* (енгл. *Quit*).

Дугме *Режим* служи за одабир уређаја који је потребно програмирати. Дугме *Ново* се користи када је потребно отворити нови фајл. Кликом на дугме *Учитај* могуће је одабрати Пајтон фајл који се налази на рачунару. Могуће је отворити и фајл са екстензијом .hex. Дугме *Сачувај* чува копију програма на рачунару, али са екстензијом .py. Уз помоћ *Flash* дугмета могуће је учитавање написаног програма директно на микробит уређај. Кликом на дугме *Датотеке* отварају се два прозора: прозор са садржајем микробит уређаја и прозор са садржајем рачунара. Ово дугме омогућава међусобно пребацивање фајлова. Дугме *Repl* (енгл. *Read Evaluate Print Loop*) омогућава комуникацију са микробит уређајем. Дугме *Plotter* служи за графички приказ очитавања података са сензора микробит уређаја у реалном времену. Дугме *Увећај* повећава величину текста, док дугме *Умањи* смањује величину текста у едитору. У Мју окружењу могуће је, кликом на дугме *Теме*, одабрати три теме за рад: светлу, тамну и високи контраст. Кликом на дугме Check могуће је проверити да ли програм има грешака. Како би код програма био уреднији, може се користити дугме *Tidy* које уређује програм тако што уклања вишак размака. Кликом на дугме *Помоћ*, у подразумеваном прегледачу отвара се страница предвиђена за помоћ у раду са Мју едитором. На крају, дугме *Излаз* затвара едитор.

3.1 Основне функције у Микропајтону

На самом почетку, потребно је упознати се са синтаксом програмског језика Микропајтон. Први задатак је да се на микробит уређају прикаже порука „Здраво,свете!“, који представља први пример приликом учења било којег програмског језика. Прва линија кода је обавезна за сваки програм написан за микробит уређај, без обзира које наредбе треба бити извршене. То је линија кода: from microbit import* . Ова наредба говори програмском језику Пајтон да ће у том задатку бити коришћене функције и наредбе предвиђене за микробит које се налазе у библиотеци која се назива microbit. Без ове наредбе не би било могуће управљати екраном микробит уређаја, улазно-излазним пиновима, сензорима, као и свим другим карактеристикама микробит уређаја. Након ове линије потребно је откуцати наредбу којом ће на микробит уређају бити приказа порука „Zdravo, svete!“. Наредба за приказ текста на екрану јесте *display.scroll('Zdravo, svete!')*. Као што је већ било речи, на екрану микробита могу бити приказани само латинични карактери. Ова наредба говори микробиту да је потребно искористити фунцкију scroll која се налази у оквиру категорије *display*. Важно је нагласити да приликом програмирања у програмском језику Пајтон постоји разлика између писања наредби малим и великим словима. Зато је важно да се приликом писања нареби употреби правилна комбинација малих и великих слова. Дакле, код за први програм би требало да изгледа као на слици 3.3.

```
1  from microbit import *
2  display.scroll('Zdravo, svete!')
3
```

Слика 3.1.1: Приказ текста на екрану

Да би овај програм био покренут на микробит уређају потребно је притиснути дугме *Flash* или преузети датотеку на рачунар, а након тога ту датотеку пребацити на микробит уређај.

3.1.1 Приказ текста на екрану микробит уређаја

Претходни задатак је могуће модификовати тако да се порука приказује све док је микробит уређај укључен тј. док није прекинут извор напајања. То је могуће урадити наредбом while True: и у оквиру ове наредбе откуцати наредбу за приказ текста. Важно је да наредба за приказ текста буде увучена за четири размака у односу на наредбу while. Симбол : означава почетак петље и свака наредба која је са четири размака увучена у односу на њу ће се извршавати све док је услов испуњен. Уколико је наредба while на адекватан начин откуцана, приликом притиска тастера ентер на тастатури курсор за куцање наредбе у оквиру петље ће бити постављен на одговарајуће место. Ова погодност смањује време писања програма. Међутим, са обзиром да програмски језик Пајтон не може да предвиди када је потребно изаћи из петље, потребно је обрисати направљене размаке користећи дугме *Backspace* на тастатури. Код треба да изгледа као на слици 3.1.2.

```
from microbit import *
while True:
    display.scroll('Zdravo, svete!')

```

Слика 3.1.2: Приказ текста на екрану микробит уређаја све док је укључен

3.1.2 Улазни параметри

Како би се одређене наредбе извршавале све док је микробит уређај укључен, прве две линије кода које су приказане на претходној слици су неопходне за решење већине задатака. Сада, потребно је модификовати претходни код тако да се порука приказује само онда када је притиснут тастер А. У оквиру петље, потребно је проверити да ли је притиснут тастер А. То се постиже куцањем наредбе *if button_a.is_pressed()*. Након куцања ове наредбе потребно је ставити симбол „:“ , који означава почетак наредбе гранања у оквиру које ће бити откуцана наредба за приказ текста. Важно је разумети да ће наредба гранања бити увучена у односу на петљу, док ће наредба за приказ текста бити увучена у односу на наредбу гранања. Уколико одговарајуће линије кода нису увучене у односу на наредбе понављања и гранања, програм ће приказивати грешку и неће радити. У неким случајевима, могуће је да програм не прикаже грешку уколико наредбе нису увучене, али онда програм неће урадити оно што је требало. Код треба да изгледа као на слици 3.1.3.

```
from microbit import *
while True:
    if button_a.is_pressed:
        display.scroll('Zdravo, svete!')
```

Слика 3.1.3: Приказ текста ако је дугме А притиснуто

Претходни програм је могуће надоградити тако да се након приказане поруке прикаже икона насмејаног лица. Наредба која омогућава приказ слика на екрану микрбит уређаја јесте функција show, која се налази у оквиру категорије *display*. Приказ слике на екрану постиже се куцањем наредбе *Image.HAPPY*, као аргумента функције show. Дакле, приказ слика на екрану постиже се извршавањем наредбе: *display.show(Image.HAPPY)*. Ова наредба треба да буде откуцана испод наредбе за приказ текста, како би се извршиле једна за другом. Код за програм треба да изгледа као на слици 3.1.4:

```
from microbit import *
while True:
    if button_a.is_pressed:
        display.scroll('Zdravo, svete!')
        display.show(Image.HAPPY)

```

Слика 3.1.4: Приказ текста и слике ако је дугме А притиснуто

Наравно, постоји још много слика које је могуће приказати на екрану микробит уређаја. Неке од њих су приказане у табели 3.1.1.

Табела 3.1.1: Списак наредби којима се приказују слике на екрану микробит уређаја

Наредба	Приказ на екрану
Image.HEART	Икона срца
Image.SAD	Тужно лице
Image.UMBRELLA	Кишобран
Image.HOUSE	Кућа
Image.DIAMOND	Дијамант
Image.GIRAFFE	Жирафа
Image.DUCK	Патка
Image.PACMAN	Пакман
Image.RABBIT	Зец
Image.HOUSE	Кућа
Image.SQUARE	Квадрат
Image.TRIANGLE	Троугао
Image.ARROW_S, Image.ARROW_N	Стрелице у различитим правцима
Image.COW	Крава
Image.SNAKE	Змија
Image.MUSIC_CROTCHET	Нота
Image.TSHIRT	Мајица

Посебно треба водити рачуна да назив слике буде написан великим словима. Поред слика које већ постоје, могуће је направити произвољну слику контролом лед диода микробит уређаја. На пример, слика насмејаног лица би могла бити приказана и наредбом *display.scroll("00000:09090:00000:90009:09990")*. Параметри од 0 до 9 означавају колико ће одговарајућа диода бити осветљена. Параметар 0 означава да је диода искључена, док параметар 9 означава максималну осветљеност диоде. Приликом задавања осветљености диода задаје се прво први ред, па други и тако редом. Осветљеност диода задаје се здесна налево.

Већ је речено да се предња страна микробит уређаја састоји од два дугмета која је могуће програмирати. На претходном примеру, приказано је како се програмира дугме А. Аналогно програмирању дугмета А, могуће је програмирати дугме Б, али и дугме А+Б. Дакле, потребно је направити програм у којем се притиском на тастер А приказује срећно лице, притиском на Б појављује се тужно лице, док се притиском на А+Б приказује срце.

Слично као код програмирања дугмета А, потребно је додати наредбу *if button_b.is_pressed():* којом се проверава да ли је притиснуто дугме Б. Ако јесте на екрану треба приказати тужно лице. Да би било проверено да ли су оба дугмета притиснута потребно је додати наредбу *if button_a.is_pressed() and button_b.is_pressed():*. Овом наредном биће проверено да ли су тастери А и Б притиснути истовремено. Ако јесу, на екрану ће бити приказана икона срца. Дакле, на крају, програм треба да изгледа као на слици 3.1.5.

```
from microbit import *
while True:

    if button_a.was_pressed():
        display.scroll("00000:09090:00000:90009:09990")
    if button_b.was_pressed():
        display.show(Image.SAD)
    if button_a.is_pressed() and button_b.is_pressed():
        display.show(Image.HEART)

```

Слика 3.1.5: Програмирање дугмића А, Б и А+Б

3.1.3 Пинови

На претходном примеру показано је да, иако на микробит уређају постоје два програмабилна дугмета, постоји могућност за давање три улазна параметра. Међутим, то некада није довољно за потребе већине задатака. За већину задатака потребно још улаза, као што су улазно-излазни пинови, који су нумерисани и налазе се на дну микробит уређаја. Микробит региструје додир ових пинова и у складу са тиме је могуће направити програм. Пинови представљају занимљиву алтернативу за физичку дугмад и представљају одличан начин за додавање додатних улаза за програм без куповине додатних компоненти. За почетак, потребно је на примеру показати на који начин је могуће користити пинове у програмима.

Потребно је написати програм у којем се приказује слика срећног лица ако је притиснут пин 0, а слику тужног лица ако је притиснут пин 1.

Поред стандардних линија кода које морају бити у сваком програму који је написан у Микропајтону, потребно је проверити да ли је пин 0 притиснут. То се постиже наредбом if pin0.is_touched(), при чему на крају if наредбе обавезно мора стајати знак :. Овом наредбом проверава се да ли је пин 0 притиснут. На исти начин, могуће је проверити да ли је пин 1 притиснут. На крају, код програма треба да изгледа као на слици 3.1.7.

```
from microbit import *
while True:
    if pin0.is_touched():
        display.show(Image.HAPPY)
    if pin1.is_touched():
        display.show(Image.SAD)


```

Слика 3.1.6: Пинови

Као што је речено у одељку 3.2. променљиве представљају концепт у којем се вредност променљиве мења приликом сваког покретања програма. Сада је потребно упознати се са начином на који се променљиве користе у Микропајтону. Променљива може бити било који тип података: број, стринг (ниска), чак и део где је могуће нацртати слику на екрану микробит уређаја. Променљиве се састоје из два дела: назива, односно имена променљиве и типа податка. Назив променљиве треба на најбољи могућу начин да опише променљиву, како би било јасније за шта се користи променљива. За назив променљиве, није могуће користити резервисане речи за Микропајтон. Такође, назив променљиве не сме почињати бројем и не сме садржати размаке или специфичне симболе.

Примену променљивих у Микропајтону, најбоље је описати кроз пример. Потребно је написати програм којим је потребно избројати број додира пина 0.

Дакле, потребно је увести нову променљиву која ће се звати: *broj_dodira*. Као што је речено у претходном пасусу, назив променљиве треба на најбољи могући начин да опише променљиву. Дакле, идеја је да вредност променљиве на почетку буде 0, а онда, ако је пин 0 притиснут вредност променљиве се повећава за 1. Код би требало да изгледа као на слици 3.1.8.

```
1 from microbit import *
2 broj_dodira = 0
3 While True:
4     if pin0.is_touched():
5         broj_dodira+=1
6
```

Слика 3.1.7: Промена вредности променљиве

3.1.4 Очитавање података са сензора

У Микропајтону могуће је очитавати податке са сензора. О врстама сензора било је речи у глави 1. На пример, потребно је очитати температуру са микробит уређаја, користећи сензор за температуру. То је могуће урадити наредбом: *display.scroll(str(temperature()))*. Ова линија кода представља комбинацију три наредбе: *display.scroll* која служи за приказ стринга на микробит уређају, *temperature()* којом се очитава податак са сензора и *str* која број конвертује у стринг. Дакле, програм треба да изгледа као на слици 3.1.9.

```
1 from microbit import *
2 while True:
3     display.scroll(str(temperature()))
4
5
6
```

Слика 3.1.8: Очитавање и приказ очитане температуре

Наредба, која је приказана на слици изнад и која представља комбинацију више наредби, наглашава кључну особину програмирања у Микропајтону: уместо да, као аргумент функције буде написан број или само једна променљива, могуће је ставити једну целину која се састоји из више инструкција и на екрану ће бити приказана излазна вредност. У овом примеру то је очитавање података са сензора за температуру.

У претходном примеру, програм је заправо готов, али није лак за читање. Број ће бити приказан на екрану микробита, али без назнаке шта он заправо значи. То може да се реши додавањем још једне наредбе у којој ће додатно бити појашњено шта тај број значи. То је наредбе: *display.scrool('°C')*. Програм треба да изгледа као на слици 3.1.10.

```
1 from microbit import *
2 while True:
3     display.scroll(str(temperature()))
4     display.scrool("°C")
5
```

Слика 3.1.9: Јаснији приказ очитане температуре

Поред сензора за очитавање температуре, микробит садржи и акцелерометар. Акцелерометар је један од два сензора који се налазе на задњој страни микробит уређаја. Акцелерометар мери убрзање у три равни: *X*,*Y* и *Z*. Акцелерометар се може користити за препознавање различитих типова покрета. Ово је најједноставнији начин интеракције са сензором. Акцелерометар је јако моћан алат приликом решавања различитих задатака. Поред тога, акцелерометар може да се користи за израчунавање угла под којим је микробит уређај позициониран, пратећи силу гравитације.

У наредном примеру ће бити описано на који начин је могуће употребити акцелерометар. Написати програм у којем се на екрану приказује икона изненађеног лица уколико се микробит протресе. Дакле, прво је потребно проверити да ли је микробит протресен. Ако јесте, на екрану микробит уређаја се приказује изненађено лице. Користећи наредбу одлучивања, као и наредбу *accelerometer.was_gesture("shake")*, потребно је проверити да ли је микробит протресен. Програм ради тако што користи акцелерометар да тражи један од низа покрета. У овом примеру то је протресање (енгл. shake). Код треба да изгледа као на слици 3.1.11.

```
1 from microbit import *
2 while True:
3     if accelerometer.was_gesture("shake"):
4         display.show(Image.SURPRISED)
5
6
```

Слика 3.1.10: Детектовање покрета

Протресање је само један од низа покрета које микробит може да препозна. Остали покрети су:

- 8g – Покрет 8g се покреће само када је убрзање у било којем смеру осам пута веће од гравитације;
- 6g – Покрет 6g се налази између брзих (8g) и нежнијих покрета убрзања (3g);
- 3g – Покрет 3g функционише слично као и 8g, али се покреће на мања убрзања.
- екран окренут на горе (енгл. face up) – овај гест се покреће онда када је предња страна микробит уређаја окренута према кориснику;
- екран окренут на доле (енгл. face down) – овај гест се покреће онда када је предња страна микробит уређаја окренута на доле.
- слободан пад (енгл. free fall) – овај гест се покреће само када је микробит испуштен и пада ка земљи;
- протреси (енгл. shake – овај гест се активира снажним протресањем микробит уређаја;
- окренут улево (енгл. left) – овај покрет је заснован на углу микробит уређаја и покреће се само када је микробит нагнут ка левој страни корисника;
- окренут удесно (енгл. right)- овај покрет је заснован на углу микробит уређаја и покреће се само када је микробит нагнут ка десној страни корисника.

3.1.5 Функција за насумичан избор бројева

У неким задацима или пројектима ће бити потребно одабрати насумичну вредност. То се може постићи функцијама*: random.randint(a, b), random.random(), random.randrange(n) u random.choice(seq*). Ове функције су дефинисане у стандардној библиотеци програмског језика

Пајтон. Помоћу горе наведених функција могуће је добити различите вредности насумично одабраних бројева.

- Функцијом *random.randint(a,b)* добија се насумичан цео број између *a* и *b*, укључујући и границе.
- Функцијом *random.random()* добија се насумичан реалан број између 0 и 1.
- Функцијом *random.randrange(n)* добија се насумичан цео број од 0 до n-1.
- Функцијом *random.choice(seq)* бира се насумичан број из неке секвенце, најчешће листе.

3.1.6 Листе

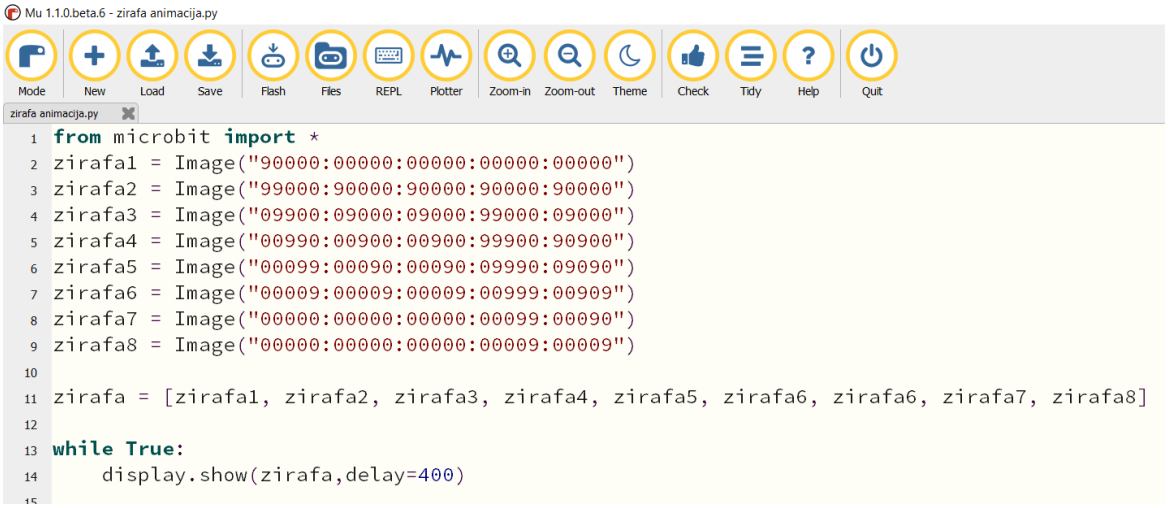
У неким задацима или пројектима неопходно је креирати листу неких елемената. Листа представља низ неколико промењљивих које су груписане под истим именом. Коришћење листа у програмима значајно смањује дужину кода и олакшава његово коришћење. Уместо писања много промењљивих, могуће је дефинисати једну промењљиву и приступати њеним члановима. Листу је могуће креирати тако што је потребно прво написати име промењљиве, а након тога у угластим заградама написати вредности промењљивих одвојене запетом: *Ime_promenljive = [vrednost1, vrednost2, vrednost3…].* Свака од вредности у листи има свој индекс. Прва промењљива има индекс 0, друга индекс 1 и тако редом. Вредностима листе приступа се тако што се напише име промењљиве и у угластим заградама се наведе индекс промењљиве којој је потребно приступити. Листе су структура података јако слична низовима, о којима је било речи у првој глави овог рада. Прва разлика између листе и низа јесте што елементи листе не морају бити истог типа, док је код низова то другачије. Такође сви елементи листе могу бити приказани на екрану без коришћења петље, док је за приказ свих елемената низа неопходно коришћење петље.

Листе је, на пример, могуће користити за креирање анимација. Ова употреба листи биће приказа на наредном примеру. Написати програм у којем се на екрану микробит уређаја приказује жирафу која се креће. За почетак, потребно је направити промењљиве које ће представљати елементе листе. Те промењљиве ће бити направљене коришћењем контроле лед диода, тако да праве симулацију кретања жирафе. Промењљиве ће изгледати као на слици 3.1.12.

```
1  from microbit import *
2  zirafa1 = Image("90000:00000:00000:00000:00000")
3  zirafa2 = Image("99000:90000:90000:90000:90000")
4  zirafa3 = Image("09900:09000:09000:99000:09000")
5  zirafa4 = Image("00990:00900:00900:99900:90900")
6  zirafa5 = Image("00099:00090:00090:09990:09090")
7  zirafa6 = Image("00009:00009:00009:00999:00909")
8  zirafa7 = Image("00000:00000:00000:00099:00090")
9  zirafa8 = Image("00000:00000:00000:00009:00009")
10
11
```

Слика 3.1.11: Креирање елемената листе

Након што су промењљиве креиране, потребно је направити листу која ће садржати ове промењљиве. Нека листа има назив *zirafa*. Када је то урађено, потребно је приказати све направљене слике на екрану. На крају, програм ће изгледати као на слици 3.1.13.

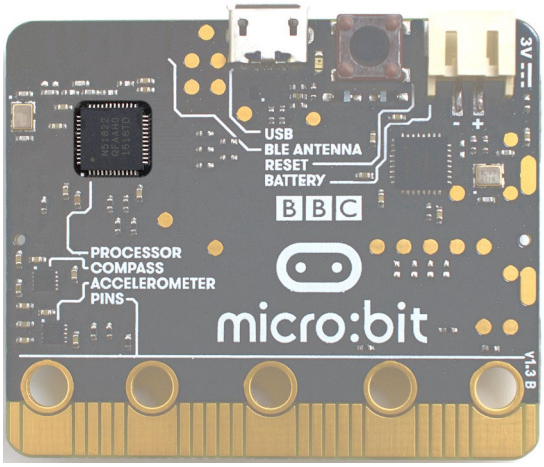


Слика 3.1.12: Креирање листе и приказ анимације на екрану

У оквиру функције *display* може се додати још један параметар *delay*. Овај параметар омогућава да се између слика прави одређена пауза у милисекундима.

3.1.7 Комуникација између микробит уређаја

Једна од најмоћнијих карактеристика микробит уређаја јесте уграђени радио модул. Радио модул је мали црни чип (слика 3.1.14) уграђен на полеђини микробит уређаја и уграђен је у његов процесор. Користећи радио могуће је омогућити комуникацију два микробит уређаја, али и могу постојати групе од десетине микробита који међусобно деле податке. Највећа предност је што нема потребе за повезивањем спољне антене, већ је радио модул могуће користити чим је микробит купљен.



Слика 3.1.13: Приказ радио чипа

Иако постоји само један физички чип, постоје две радио функције. Прва функција представља *peer-to-peer* комуникацију између микробит уређаја. Ова комуникација подразумева слање сигнала који могу примити било који микробит уређаји у близини. *Peer-to-peer* радио функција се користи онда када је потребно креирати пројекте у којима микробит уређаји комуницирају једни са другима. Друга функције је *Bluetooth Low Energy (BLE*), која представља варијанту Bluetooth бежичног стандарда који се налази у свим паметним телефонима и већини таблета, али функционише на мањој удаљености у односу на *peer-to-peer* комуникацију.

Најједноставнија употреба радио модула јесте комуникација један на један, што подразумева да један микробит комуницира са другим микробит уређајем.

Основне функције које се користе за комуникацију између микробит уређаја су:

- *radio.on();*
- *radio.off();*
- *radio.send(poruka);*
- *radio.recive();*
- *radio.config();*
- *radio.reset().*

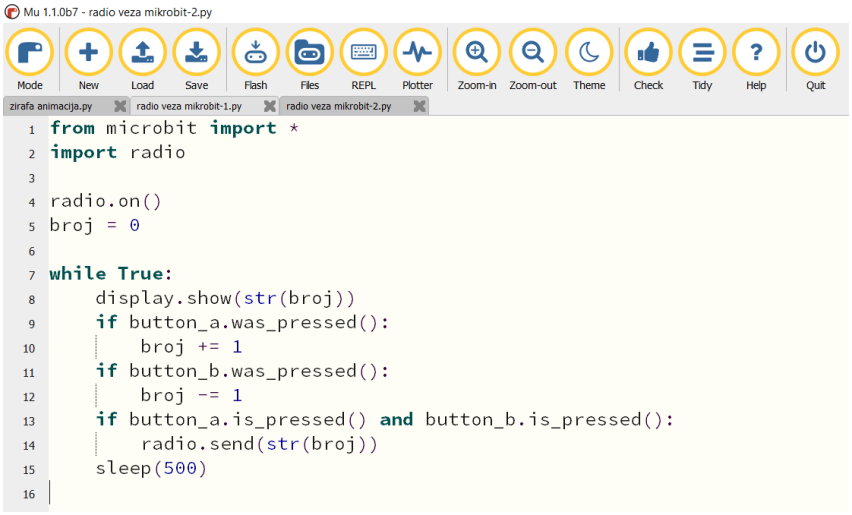
Прва наведена функција користи се за укључивање радио везе, док се друга наведена функција користи за искључивање радио функције. Ове две функције су неопходне за комуникацију са обзиром да микробит уређај троши много енергије када је омогућена радио веза. Онда када је радио веза укључена, са микробит уређаја је могуће послати неки податак путем функције *radio.send(poruka)*, при чему порука може бити стринг односно ниска карактера. Други микробит или микробитови могу да приме поруку помоћу функције *radio.recive()*. Преостале две функције користе се за комуникацију између више микробит уређаја, о чему ће бити речи у наставку ове главе.

У наредном примеру ће бити приказан начин комуникације између два микробит уређаја. Приликом решавања задатка биће коришћене функције које су описане у претходном пасусу. Примене комуникације између микробит уређаја су многобројне, али је јако важно да се ученици упознају са овим концептом кроз једноставан пример.

1. Написати програм који омогућава комуникацију између два микробит уређаја. Ако се са једног микробит уређаја пошаље неки број, на другом микробит уређају треба да се појави два пута већа вредност од послате.

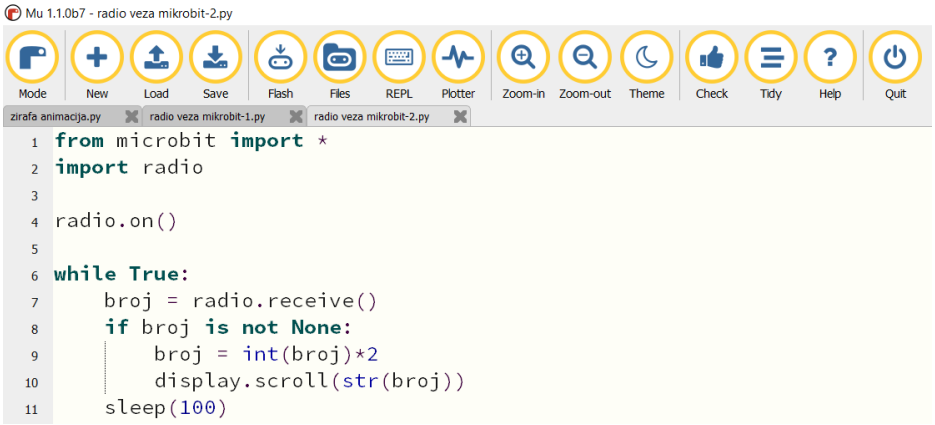
   Овај пример представља најједноставнију употребу радио везе за комуникацију између два микробит уређаја. Важно је да, ученици на једноставном примеру разумеју како функционише радио веза, како би је искористили у знатно сложенијим примерима.

   Дакле, потребно је написати програм за два микробит уређаја. На првом микробит уређају треба написати програм за слање поруке, односно броја. Прво, треба одабрати који број ће бити послат. То се решава на следећи начин. Потребно је направити променљиву чија ће вредност на почетку програма бити 0. Након тога, притиском на тастер А број се повећава за 1, притиском на тастер Б број се смањује за 1, док се притиском на оба тастера истовремено шаље порука другом микробит уређају. Програм за први микробит приказан је на слици 3.1.15.



Слика 3.1.14: Програм за први микробит

Други микробит уређај ће примити вредност, увећаће је два пута и приказати на екрану. Програм за други микробит приказан је на слици 3.1.16.



Слика 3.1.15: Програм за други микробит

Често је комуникацију између микробит уређаја могуће организовати врло једноставно. На пример један микробит шаље поруку а други је прима, након чега они могу да замене улоге. У многим применама је ово сасвим довољно. У нешто сложенијим сценаријима може, на пример, бити потребно да више микробитова шаље поруке, а да их један микробит прима и обавља одговарајуће радње по пријему сваке од њих. У том случају послату поруку може да прими сваки микробит, али је само један од њих програмиран да то и ради. Могућа је и обрнута организација, у којој само један микробит шаље поруке а сви остали их примају и спроводе одговарајуће акције. Редак је случај да је потребна комуникација која је сложенија од поменутих.

Када у једној учионици две групе ученика раде на истом пројекту који користи радио везу, може да се деси да уређаји из једне групе ненамерно примају поруке од уређаја из друге групе и да то омета жељени начин функционисања и збуњује учеснике у пројекту. Тада је потребно раздвојити микробит уређаје у групе, тако да свака група комуницира за себе и да се групе не мешају једна другој у разговор. То је могуће остварити помоћу функције *radio.config()*. Ова функција служи за подешавање параметра комуникације. Један од параметара комуникације је *channel* (канал). Канали за комуникацију могу имати вредност од 0 до 83 и

параметар *channel* може да се постави на било коју од њих. На почетку, док вредност канала не буде промењен, сви микробитови комуницирају преко канала 7. Како би се избегло мешање једне групе другој у разговор, довољно је подесити да сви микробитови једне групе користе један канал, на пример *radio.config(channel* = 3), а да сви микробитови друге групе користе неки други канал, на пример *radio.config(channel* = 4), или их оставити на каналу 7, ако нема треће групе којој би то сметало.

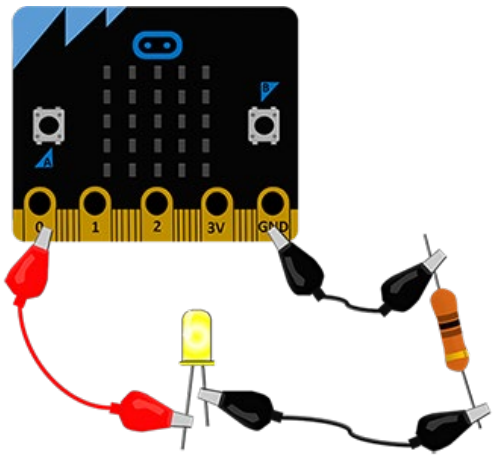
Функцијом *radio.config()* могу се подесити и други параметри комуникације, као што су, на пример, максимална дозвољена дужина поруке, максималан број порука које могу да чекају да буду прочитане (ако се за врло кратко време пошаље превише порука, неке ће бити изгубљене), јачина радио сигнала (јачи сигнал значи већи домет али и брже трошење батерије) итд. али остали параметри неће бити коришћени у овом раду. Функција *radio.reset()* служи за враћање свих ових параметара на почетне, односно подразумеване вредности. Пример повезивања више микробит уређаја биће приказан у наставку овог рада.

Више примера, као и детаљнија појашњења везана за ову тему могу се пронаћи у књизи „ *The Official BBC micro:bit User Guide“* аутора Gareth Halfacree.

3.1.8 Повезивање додатних компоненти

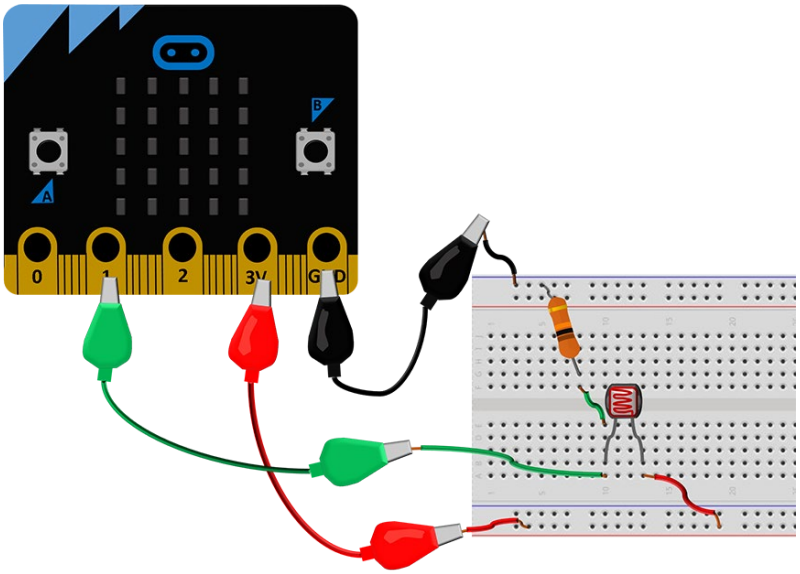
У одељку 1.1. било је речи о пиновима и њиховој важности за различите задатке. На микробит уређају се осим пет основних пинова налази још двадесет мањих пинова, којима је могуће приступити преко додатног конектора тј. ивичног конектора (енгл. edge connector). Велики број задатака и пројеката могу бити реализовани без додатне опреме. Али, увек је занимљивије када ученици могу да виде како нешто засветли, помера се или чују различите звуке. Уз додатне компоненте могуће је управљати уређајима који захтевају много више снаге него што микробит може да пружи. У овом одељку биће приказана употреба неколико додатних компоненти. Ове компоненте могуће је купити путем интернета или у продавницама електронске опреме. За повезивање и коришћење додатних компоненти, углавном је потребна додатна помоћ. Зато се пројекти који захтевају додатне компоненте углавном реализују у сарадњи са наставницима технике и технологије, који су такође могли да похађају обуку за коришћење микробит уређаја.

Додатна компонента која се јако често користи у пројектним задацима јесте светлећа диода. Светлећа диода се састоји из две електроде: дуже и краће. Дужа електрода јесте позитивна електрода и назива се анода, док је краћа електрода негативна електрода и назива се катода. Како би светлећа диода била повезана са микробит уређајем, потребно је користити проводнике са крокодил-штипаљкама. Светлеће диоде могу бити црвене, жуте или зелене боје напона између 1.8V и 2.2V. На пиновима микробит уређаја је напон 3.3V, па је неопходно да са светлећом диодом буде повеза и отпорник од 100Ω. На слици 3.1.16 је приказан начин повезивања светлеће диоде са микробит уређајем.



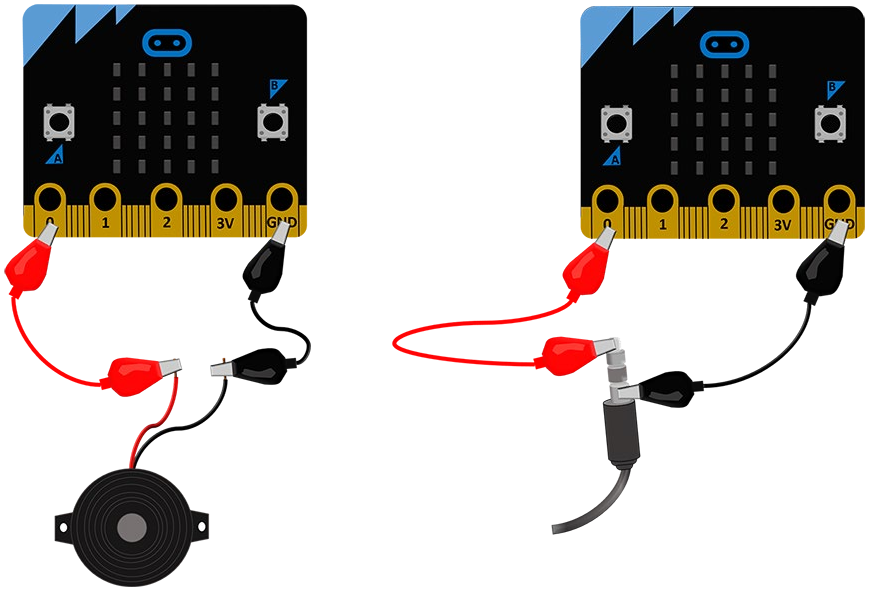
Слика 3.1.16: Повезивање светлеће диоде и микробит уређаја

Друга компонента која може бити занимљива и применљива у разним пројектима јесте фото-отпорник. Фото-отпорник може да се користи као сензор за одређивање осветљености просторије, Иако микробит већ садржи сензор за одређивање осветљености, фото-отпорник је доста прецизнији. Када није осветљен, отпорност фото-отпорника је јако велика, док са повећањем осветљености његова отпорност нагло опада. На слици 3.1.18 је приказан начин за повезивање микробит уређаја са фото-отпорником коришћењем додатне матичне плоче.



Слика 3.1.17: Повезивање фото-отпорника и микробит уређаја

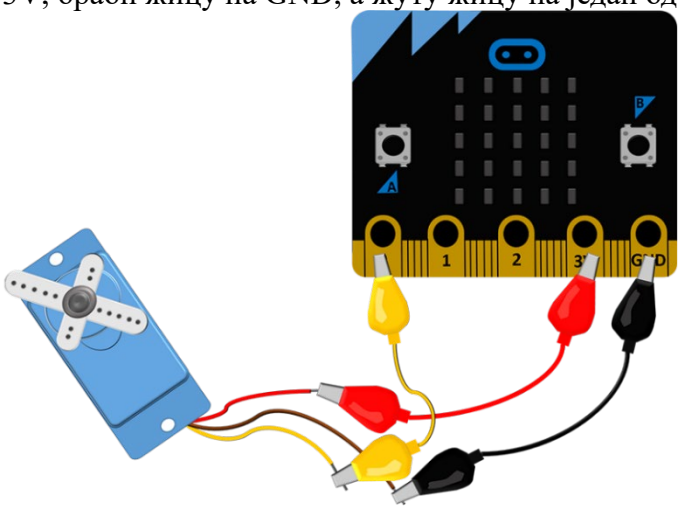
Микробит уређај нема уграђени звучник, али уз помоћ пинова и додатне опреме, могуће је повезати екстерни звучник или слушалице.



Слика 3.1.18: Повезивање микробит уређаја са екстерним звучником и слушалицама

На слици 3.1.19 је приказан начин за повезивање микробит уређаја са звучником или слушалицама. Приликом повезивања звучника, црна жица повезује се на GND пин, а црвена на пин P0, путем којег се шаљу сигнали за генерисање звукова. Код слушалица, доњи део прикључка (ближи пластици) се везује за GND пин, а врх на пин P0. Поступак програмирања мелодије је исти без обзира на то који уређај ће бити коришћен за слушање музике.

Још један занимљива компонента која може бити повезана са микробит уређајем јесте серво мотор. Помоћу серво мотора могуће је контролисати угаони положај, који има распон од 0 до 180 степени. Серво мотор се састоји из три жице тј. контакта: црвене, жуте и браон. Повезивање серво мотора и микробит уређаја врши се на следећи начин: црвену жицу повезати са позитиван напон тј. пин 3V, браон жицу на GND, а жуту жицу на један од пинова 0,1 или 2.



Слика 3.1.19: Повезивање серво мотора и микробит уређаја

Са обзиром да је напон напајања серво мотора 4.8-6V, савет је да се серво мотор не напаја директно преко микробит уређаја, већ да буде повезан на додатно напајање. Међутим,

иако серво мотор не буде повезан на додатно напајање, вероватно ће бити могуће управљање серво мотором.

Детаљнија упутства о овој теми је могуће пронаћи у књизи „*Beginning BBC micro:bit: A Practical Introduction to micro:bit Development“*.

3.2 Разни задаци

У овом поглављу биће приказане неке од идеја за пројекте које је могуће реализовати на часовима информатике и рачунарства уз сарадњу са наставницима технике и технологије. За неке од пројеката није потребна никаква додатна опрема, осим микробит уређаја, док је за неке потребно обезбедити проводнике, светлеће диоде или додатне сензоре. Сваки пројектни задатак се састоји из три фазе: истраживање, решавање проблема и презентовање рада и резултата истраживања.

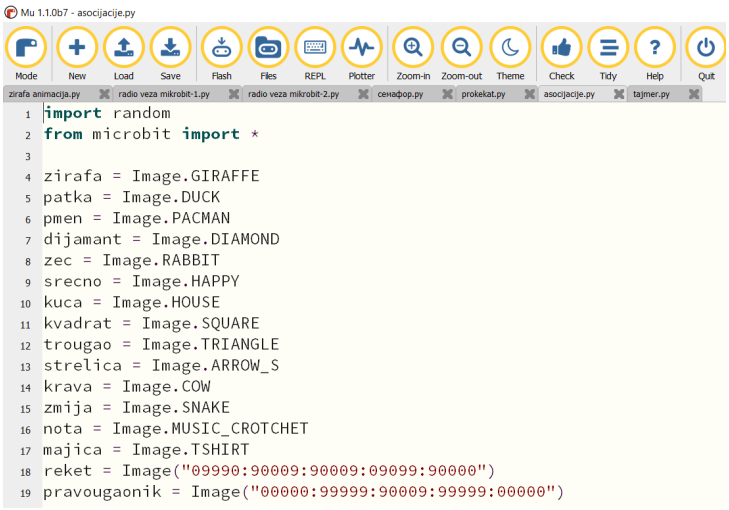
3.2.1 Игра асоцијација

Прва идеја за пројекат јесте игра асоцијација. За ову игру нису потребне додатне компоненте, већ два микробит уређаја и батерије за напајање. За игру асоцијација постоји апликација коју је могуће преузети за Андроид телефон. Програмирајући ову игру за микробит уређај, ученици могу да виде повезаност са апликацијама које већ постоје. Наравно, игра коју ученици програмирају је доста једноставнија од оне која већ постоји за Андроид. У педагошком смислу, важно је то што ученици могу да програмирају, а након тога да се забаве користећи програм који су сами креирали. Правила игре су следећа: један учесник држи микробит уређај на челу. За почетак игре учесник притиска тастер А и на екрану микробит уређаја се приказује слика. Други учесник треба да објасни појам који је приказан. Уколико први учесник погоди појам помера микробит ка горе и број поена се повећава, а уколико не погоди помера микробит ка доле, прескаче тај појам и приказује се следећи. За приказ броја бодова потребно је притиснути тастер Б. Други микробит уређај ће служити као тајмер. Игра може да траје 60,120 или 180 секунди. Притиском на тастер А микробит одбројава 60 секунди, притиском на Б микробит одбројава 120, а притиском тастера А и Б истовремено 180 секунди. Када истекне време, на микробит уређају се емитује музика. Наравно, да би музика била емитована, потребно је микробит уређај повезати са звучником. Дакле, пре него што почну да програмирају, ученици морају да осмисле који објекти ће бити приказани на екрану. Могу да користе већ постојеће слике, али могу и да креирају нове осветљавање лед диода на екрану микробит уређаја. Затим, потребно је да програмирају други микробит уређај који представља тајмер. Програм за микробит уређај који представља тајмер приказан је на слици 3.2.1.

```
from microbit import *
import music

if button_a.is_pressed:
    for i in range(60, -1, -1):
        display.show(str(i))
        sleep(1000)
    music.play(music.DADADADUM)

if button_b.is_pressed:
    for i in range(120, -1, -1):
        display.show(str(i))
        sleep(1000)
    music.play(music.DADADADUM)

if button_a.is_pressed and button_b.is_pressed:
    for i in range(180, -1, -1):
        display.show(str(i))
        sleep(1000)
    music.play(music.DADADADUM)

```

Слика 3.2.1: Микробит као тајмер

Сада, треба написати програм за микробит на којем ће се приказивати слике. За почетак, потребно је креирати промењиве које ће представљати слике које ће насумично бити приказиване на екрану микробит уређаја. Овај део кода приказан је на слици 3.2.2.



Слика 3.2.2: Креирање слика

Даље, потребно је креирати листу чији ће елементи бити промењиве које су направљене. Такође, потребно је креирати промењиву која ће бројати поене. На почетку програма вредност ове промењиве је 0. У бесконачној петљи треба креирати две наредбе гранања. Ако је притиснут тастер А, на екрану треба приказати насумично одабрану слику. То се постиже функцијом *random.choice(parametar)*. Ако је детектован покрет надоле, слика се мења и број поена се повећава за 1, а ако је детектован покрет на горе само се приказује наредна слика. Ако је притиснут тастер Б на екрану се приказује број бодова и игра је завршена. Овај део програма приказан је на слици 3.2.3.

```
broj = 0
slike = [
    zirafa,
    dijamant,
    zec,
    srecno,
    kuca,
    kvadrat,
    trougao,
    strelica,
    majica,
    krava,
    zmija,
    nota,
    reket,
    patka,
    pravougaonik,
    pmen,
]
while True:
    if button_a.is_pressed():
        display.show(random.choice(slike))

    if accelerometer.was_gesture("face down"):
        display.show(random.choice(slike))
        broj += 1

    if accelerometer.was_gesture("face up"):
        display.show(random.choice(slike))

    if button_b.is_pressed():
        display.scroll("Broj bodova je " + str(broj))
        display.scroll("Igra je gotova!")
```

Слика 3.2.3: Креирање листе и детектовање покрета

3.2.2 Микробит лампа

Други пројекат који ће бити приказан у овом раду јесте микробит лампа. Овај пројекат састоји се из више мањих целина које треба спојити заједно. За овај пројекат потребно је:

- микробит уређај;
- батерије;
- картон, маказе, колаж папир.

У основи, овај пројекат се састоји из решавања најједноставнијих проблема. Међутим, иако програмирање није изазован задатак у овом пројекту, главни циљ овог пројекта је слагање свих компоненти у једну. Дакле, циљ пројекта јесте креирање лампе уз помоћ микробит уређаја , који ће ученици свакодневно користити и заправо видети продукт њиховог рада.

За почетак, потребно је написати код за микробит уређај тако да се лед диоде укључе онда када је осветљеност у просторији мања него што треба. У овом примеру лед диоде ће се укључити ако је осветљеност у просторији мања од 60. Прво треба проверити да ли је очитана вредност осветљења мања од 60. То се постиже наредбом гранања и провером услова *display.read_light_level() < 50*. Ако је услов задовољен, на екрану се укључују све лед диоде са максималним осветљењем, а ако није испуњен онда се лед диоде гасе. Ниво осветљења се проверава на сваких сат времена. Код за микробит уређај треба да изгледа као на слици 3.2.4.

```
from microbit import *

upaljeno_svetlo = False
while True:
    if display.read_light_level() < 50:
        display.show(Image("99999:99999:99999:99999:99999"))
    else:
        display.clear()
    sleep(3600)
```

Слика 3.2.4: Код микробит уређаја за пројекат лампа

Наравно, приказани код је могуће унапредити. Један од начина јесте да се лед диоде укључују и онда када микробит региструје звук, као што је тапшање или пуцање прстију. За овакво унапређење, неопходно је користити нову верзију микробит уређаја (V2). Да би овај ефекат био постигнут у програму је неоходно користити променљиву која се зове *upaljeno_svetlo* како би пратио стање светла: да ли је светло упаљено или није. Ова променљива јесте посебна врста променљиве, логичка променљива. Логичка променљива може имати само две вредности: тачно (енгл. true), односно укључено или нетачно (енгл. false), односно искључено. Када микрофон микробит уређаја детектује гласан звук, код мења вредност променљиве *upaljeno_svetlo* постављајући њену вредност на *false*, односно искључено. То значи да када микробит детектује звук (на пример тапшање), у коду се вредност променљиве мења у нетачно, односно false. То значи да, онда када је детектован звук, ако је вредност променљиве *upaljeno_svetlo* нетачно и лед диоде искључене, тада вредност променљиве постаје тачно и укључују се лед диоде. Ако је вредност променљиве *upaljeno_svetlo* тачно и ако су лед диоде биле укључене, тада вредност променљиве постаје нетачно и лед диоде се искључују тј. приказује се празан екран. Унапређен код треба да изгледа као на слици 3.2.5.

```
from microbit import *

upaljeno_svetlo = False
while True:
    if display.read_light_level() < 50:
        display.show(Image("99999:99999:99999:99999:99999"))
    else:
        display.clear()
    if microphone.was_event(SoundEvent.LOUD):
        upaljeno_svetlo = not upaljeno_svetlo
        if upaljeno_svetlo:
            display.show(Image("99999:99999:99999:99999:99999"))
        else:
            display.clear()
    sleep(3600)
```

Слика 3.2.5: Унапређени програмски код

Сада, ученици треба да направе изглед лампе користећи, картон, маказе и колаж папир. Ова активност може бити реализована у сарадњи са наставником/наставницом ликовне културе. На овај начин ученици, поред програмирања, могу да покажу креативност.

3.2.3 Аутоматско заливање биљке

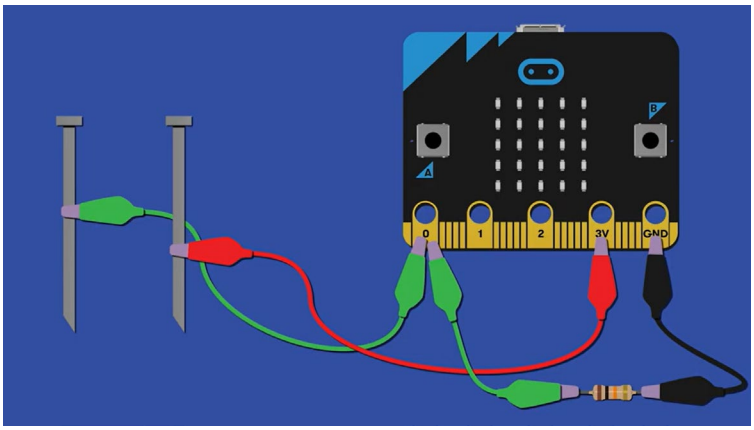
Наредна идеја за пројекат јесте аутоматско заливање биљке. За потребе овог пројекта потребни су:

- микробит уређај;
- серво мотор;
- отпорници;
- проводник;
- два ексера;
- посуда са водом;
- сламчица и
- биљка.

Идеја овог пројекта јесте да биљка буде аутоматски заливена када влажност земље буде мања од жељене. Овај пројекат може послужити за заливање кућних биљака, али указује на глобалан проблем наводњавања. Како би пројекат био успешно реализован, неопходна је употреба додатних компоненти које је могуће набавити у продавницама електронске опреме или путем интернета.

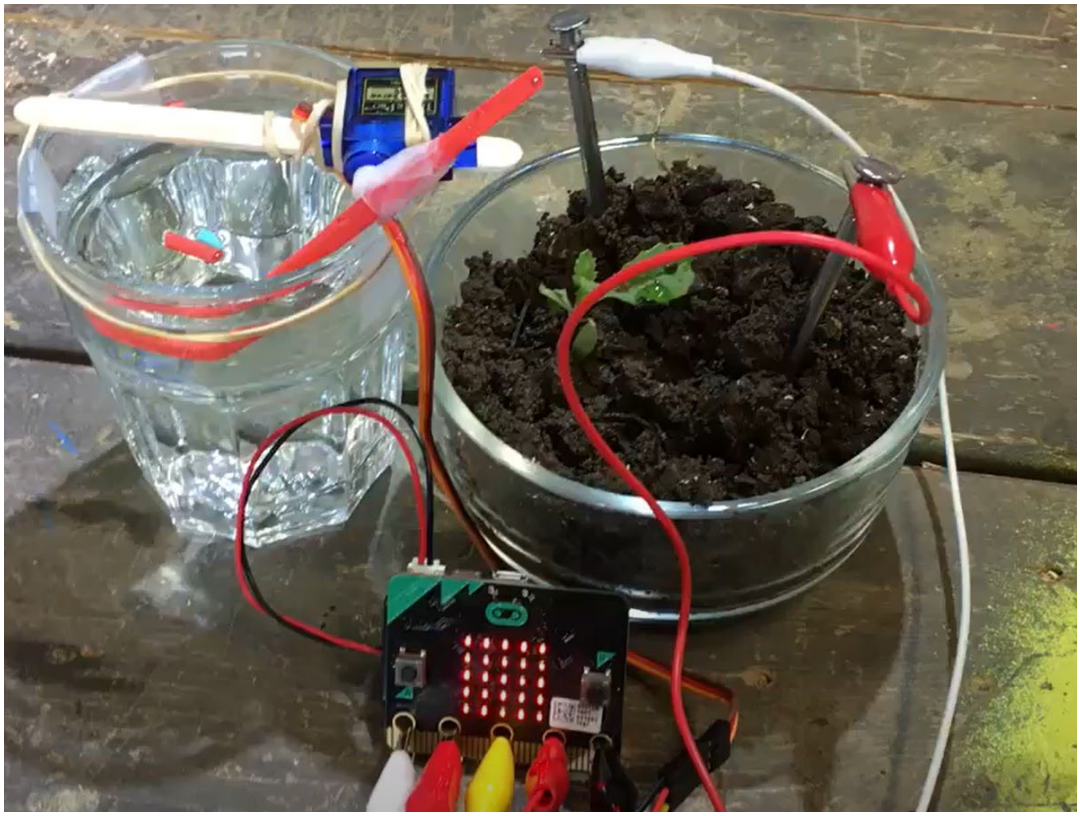
Овај пројекат је јако важан јер ученици могу да разумеју да програмирањем микробит уређаја могу да направе програме који представљају решења реалних проблема. На пример, овај пројекат могу да искористе онда када оду на летовање или зимовање, а нема ко да залива биљке њихових родитеља. Са обзиром да пројекат подразумева коришћење додатне опреме, повезивање отпорника и проводника, пожељно га је реализовати уз помоћ колега који предају технику и технологију.

Дакле, прво је потребно мерити влажност земљишта. За то ће нам послужити ексери и проводници са крокодил штипаљкама. Микробит већ поседује сензор за контролу влажности. Са обзиром да биљка помоћу корена апсорбује воду, сензори мере количину влаге у земљи мерењем количине струје помоћу ексера. Ексере треба поставити у земљу у којој се налази биљка. Познато је да је вода добар проводник електричне струје. Самим тим, што је већа влага у земљи, већа је проводност између два ексера и обрнуто. Сада, отпорник вредности 10kΩ треба прикључити на GND пин са једне стране и на пин 0 са друге стране. Затим пин 0 и пин 3V треба повезати са ексерима. Начин повезивања приказан је на слици 3.2.6.



Слика 3.2.6: Повезивање проводника (ексера) и микробит уређаја

У овом примеру, гранична вредност влажности биће 800. Ако је вредност очитавања влажности мања од 80, биљку треба залити. Поступак очитавања влажности се понавља сваке три секунде након сваког заливања, како би земља упила воду која је доливена. Сада, треба направити систем за заливање коришћењем серво мотора и сламчице. Сламчицу треба засећи са горње стране тако да може да подигне довољну количину воде. Такође је треба скратити и укосо исећи на доњем крају. Серво мотор треба причврстити за неку дрвену подлогу (на пример штапић од сладоледа) и ту подлогу причврстити за чашу у којој се налази вода. Сада, сламчицу треба залепити за серво мотор. На крају, серво мотор треба повезати са микробит уређајем тако што се црвена жица повеже на пин 3V, браон жица на GND, а жута жица на пин 2. Крајњи продукт пројекта треба да изгледа као на слици 3.2.7.



Слика 3.2.7: Крајњи изглед пројекта

Како је серво мотор повезан на пин 2, треба користити блокове за читање вредности са серво мотора како би било могуће променити угао серво мотора. Потребно је променити вредност угла на 0, сачекати да се заврши заливање и након тога променити вредност угла на 180 степени. Пауза до следеће провере влажности земљишта је 10 сати. Притиском тастера A приказује се очитана вредност влажности земљишта. Код треба да изгледа као на слици 3.2.8.

```
from microbit import *

while True:
    vlaznost = pin0.read_analog()
    if vlaznost < 800:
        pin2.write_analog(0)
        sleep(3000)
        pin2.write_analog(180)
        sleep(3000)
        pin2.write_analog(0)
    else:
        sleep(36000)
    if button_a.is_pressed():
        display.scroll(str(vlaznost))
        sleep(1000)
```

Слика 3.2.8: Програм уз пројекат „Аутоматско заливање биљке“

Више креативних идеја за пројектну наставу је могуће пронаћи у књизи „*Micro:bit for Mad Scientists : 30 clever coding and electronics “* аутора Симона Монка.

4 Закључак

Коришћење микробита у настави је осмишљено са циљем да се ученици упознају са програмирањем, али и да користе релевантне информације како би самостално креирали различите пројекте. Квалитетни и разноврсни ресурси у великој мери олакшавају упознавање са овим уређајем и учење програмирања. Микробит уређај не треба посматрати само као наставну јединицу у оквиру предмета информатика и рачунарство, већ као уређај који се користи за улазак у свет програмирања и размишљање о даљем професионалном развоју. У овом раду је показано да је програмирање микробит уређаја пројекат са јако великим потенцијалом и тенденцијом да ученицима учини програмирање занимљивим. Информатика и рачунарство је у суштини прагматичан предмет у оквиру којег се вештине најбоље стичу кроз практичан рад. Знање из овог предмета се најефикасније развија унутар практичног окружења. Међутим, ту може настати проблем када постоји ограничење у расположивој опреми која помаже наставу информатике и рачунарства. У раду је описан изглед и неке од карактеристика микробит уређаја. Са обзиром да су, у оквиру пројекта „Школе за 21. век“, све школе у Србији добиле микробит уређаје, коришћење овог уређаја у настави представља сјајан начин за превазилажење дигиталног јаза. Због наведених и описаних карактеристика микробит уређаја може се закључити да микробит уређај представља одлично наставно средство које ће кроз праксу и разноврсне примере из свакодневног живота ученицима приближити програмирање и подстаћи их да размишљају критички.

У овом раду приказано је коришћење блоковског програмског језика Мејк Код. На разноврсним примерима, који осликавају употребу блокова, сензора, лед диода и контролу тока програма, стиче се утисак да програмирање у Мејк Код окружењу у великој мери олакшава ученицима савладавање основних концепата програмирања. На примерима су приказане основне, али и напредне функционалности Мејк Код окружења. Важно је ученицима основне школе предочити васпитне, образовне и друштвене вредности наставних садржаја, као и њихово значење за развој личности и оспособљавање ученика за живот. Приликом планирања наставе, неопходно је истраживати корелативне везе садржаја информатике и рачунарства са садржајима других наставних предмета, како би се ефикасније остварио општи циљ и исходи васпитно-образовног процеса. Примери који се налазе у овом раду употпуњују сваку од тема и значајно олакшавају разумевање датих концепата и повезивање програмирања микробит уређаја са другим наставним садржајима. Такође, приказани су задаци у којима ученици треба да програмирају игре и на сјајан начин осликавају примену описаних концепата програмирања. Игре су данас ученицима јако близак концепт са корисничког аспекта, па програмирање игара представља посебан изазов за ученике. Дати примери су посебно значајни и корисни за унапређивање наставе информатике и рачунарства јер јасно приказују да микробит обилује занимљивим и конструкционим могућностима који ученицима пружају прилику да самостално направе сопствени интерактивни уређај и увиде реалне резултате свог рада директно на микробит уређају. Кроз разноврсне примере приказана је примена раније поменутих основних концепата програмирања са циљем да подстакну ученике да иду корак даље и програмирају микробит уређај у текстуалном програмском језику Микропајтон коришћењем Мју едитора. У овом раду су дате идеје за пројектну наставу које представљају значајан део наставе у којима ученици, поред програмирања, треба да примене и друга знања и вештине, како би резултат пројеката био успешнији. Пројектне идеје које су приказане осмишљене су са намером да охрабре ученике на њиховом дигиталном путу и подстакну их да након реализације ових идеја, осмисле нова, иновативнија решења за неки други реалан проблем.

Микробит као наставно средство представља заокружено технолошко решење за савладавање основних концепата програмирања. Микробит може радити као симулатор, али и као стварни уређај. У данашње време, неопходно је развијати рачунарску писменост јер је веома важна за професионални развој. Зато је важно охрабрити ученике да креирају сопствене програме и истражујући осмисле начине за њихова унапређења. Такође, још један од важних циљева овог рада је да се наставницима пружи подршка у настави. У раду су приказани различити методички приступи коришћења микробит уређаја, а примери дати у овом раду могу бити корисни наставницима који желе да унапреде квалитет наставе и учине програмирање микробит уређаја занимљивим.

Литература

Halfacree, G. (2018). *The Official BBC micro:bit User Guide.* Канада: John Wiley & Sons, Inc.

Donat, W. (2017). *Getting started with the micro:bit.* Сан Франциско: Maker Media, inc., 1700 Montgomery Street, San Francisco

Monk, S. (2019). *Micro:bit for Mad Scientists : 30 clever coding and electronics.* No Starch Press, Inc, San Francisco.

Servante, P. (2018), *Beginning BBC micro:bit: A Practical Introduction to micro:bit Development.* Шри Ланка: Apress.

Петља. (2018). *Програмирање микробит уређаја у МејкКоду – приручник за пети разред*. Преузето са: https://petlja.org/biblioteka/r/kursevi/microbitbc#id2.

Петља. (2018). *Програмирање микробит уређаја у Микропајтону – приручник за пројектну наставу за седми и осми разред*. Преузето са: https://petlja.org/biblioteka/r/kursevi/microbitprojektna .

The Micro:bit Educational Foundation (2016). Званична страница намењена микробит уређајy: https://microbit.org/.