Programi u svakodnevnom životu

Zelić, Petra

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Petra Zelić

PROGRAMI U SVAKODNEVNOM ŽIVOTU

Diplomski rad

Voditelj rada:
Doc. dr. sc. Goranka Nogo

Zagreb, rujan, 2023.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Od srca zahvaljujem svojoj mentorici doc. dr. sc. Goranki Nogo na strpljenju i pomoći pri izradi diplomskog rada te korisnim savjetima i vodstvu za vrijeme diplomskog studija. Veliko hvala mojoj obitelji i prijateljima na ljubavi i podršci tijekom studiranja.

Sadržaj

Uvod

Brz i konstantan razvoj računalne znanosti doveo je do toga da smo u svakodnevnom životu okruženi mnoštvom fizičkih objekata čiji se rad temelji na izvršavanju nekog ugrađenog programa. Za neke od tehnoloških noviteta današnjeg doba poput električnih automobila, pametnih satova i robotskih usisivača to je i prilično očito. Međutim, veliki je broj i drugih uređaja koji postoje već dulje vrijeme i iz čijeg se načina rada može štošta naučiti. Uređaji poput termostata, rasvjetnih tijela, kućanskih aparata, pokretnih stepenica i slično, neki su od takvih objekata koji rade na temelju ugrađenog programa i putem senzora ili drugih komponenti komuniciraju sa svijetom oko sebe.
U ovom diplomskom radu ćemo istražiti i proučiti neke od takvih objekata te razraditi programe koji bi mogli simulirati isto ponašanje, pomoću pločica s mikrokontrolerom. Cilj rada je osmisliti takve aktivnosti koje se mogu koristiti i za ostvarenje odgojno-obrazovnih ishoda u skladu s Kurikulumom za nastavni predmet Informatika za osnovne škole i gimnazije u Republici Hrvatskoj. Uz to, poželjno je da programi razrađeni u aktivnostima budu dovoljno poznati i zanimljivi učenicima kako bi im pobudili motivaciju za programiranjem i samostalnim otkrivanjem svijeta koji ih okružuje.

U prvom poglavlju dan je osvrt na Kurikulum i ishode predmeta Informatika koji se mogu ostvariti programiranjem mikrokontrolera te izradom programa iz svakodnevnog života.
U drugom je poglavlju dan općeniti opis Arduina i svega što on obuhvaća. Opisana je pločica na kojoj će se razvijati programi u radu.
U trećem su poglavlju dane aktivnosti primjenjive za srednju školu.
U prvoj se aktivnosti na jednostavnijim primjerima upoznaje s pločicom, njenim komponentama i softverskim alatom za pisanje programa. U drugoj se aktivnosti na primjerima semafora pišu složeniji kodovi s vlastitim funkcijama i novom komponentom zujalicom. Treća je aktivnost digitalnog klavira u kojoj učenici mogu samostalnim radom uvježbavati naučeno i unaprijediti zadatke vlastitim idejama. U četvrtoj aktivnosti uvode se dvije nove komponente, senzor pokreta i RGB LED dioda, za potrebe izrade protuprovalnog alarmnog sustava. Posljednja aktivnost opisuje rad s još dvije nove komponente, ultrazvučnim senzorom i servo motorom koji se koriste za simulaciju automatskih kliznih vrata.

Poglavlje 1

1. Kurikulum

1.1 Osvrt na ishode iz kurikuluma

U Kurikulumu nastavnoga predmeta Informatika za osnovne i srednje škole ishodi i ciljevi predmeta podijeljeni su u četiri domene: e-Društvo, Digitalna pismenost i komunikacija, Računalno razmišljanje i programiranje te Informacije i digitalna tehnologija.

Slika 1.1: Prikaz domena prema ciklusima

Domene nisu jednako zastupljene u svim razredima, što je vidljivo iz grafičkog prikaza domena prema ciklusima na Slici 1.1 (preuzeto iz Kurikuluma). Iako pojedini ishodi ne zahtijevaju jednako vrijeme učenja, iz prikaza je vidljivo da u završnim ciklusima obrazovanja domena Računalno razmišljanje i programiranje postaje najzastupljenija. Očekivano je i da će proučavanjem te izradom programa iz svakodnevnog života u nastavi biti moguće ostvariti dio ishoda iz te domene. U nastavku slijedi pregled ishoda od 5. razreda osnovne škole do 4. razreda srednje škole koji bi se mogli ostvariti izradom programa poput onih prikazanih u ovom radu.

5. RAZRED
Nakon pete godine učenja predmeta Informatika u domeni **Računalno razmišljanje i programiranje** učenik: B.5.1 koristi se programskim alatom za stvaranje programa u kojemu se koristi ulaznim i izlaznim vrijednostima te ponavljanjem B.5.2 stvara algoritam za rješavanje jednostavnoga zadatka, provjerava ispravnost algoritma, otkriva i popravlja pogreške.
6. RAZRED
Nakon šeste godine učenja predmeta Informatika u domeni **Računalno razmišljanje i programiranje** učenik: B.6.1 stvara, prati i preuređuje programe koji sadrže strukture grananja i uvjetnoga ponavljanja te predviđa ponađanje jednostavnih algoritama koji mogu biti prikazani dijagramom, riječima govornoga jezika ili programskim jezikom B.6.2 razmatra i rješava složeniji problem rastavljajući ga na niz potproblema.
Nakon šeste godine učenja predmeta Informatika u domeni **Digitalna pismenost i komunikacija** učenik: C.6.1 izrađuje, objavljuje te predstavlja digitalne sadržaje s pomoću nekoga online i/ili offline programa pri čemu poštuje uvjete korištenja programom te postavke privatnosti.
7. RAZRED
Nakon sedme godine učenja predmeta Informatika u domeni **Računalno razmišljanje i programiranje** učenik: B.7.1 razvija algoritme za rješavanje različitih problema koristeći se nekim programskim jezikom pri čemu se koristi prikladnim strukturama i tipovima podataka B.7.3. dizajnira i izrađuje modularne programe koji sadrže potprograme u programskom jeziku B.7.4. koristi se simulacijom pri rješavanju nekoga, ne nužno računalnoga, problema.

8. RAZRED
Nakon osme godine učenja predmeta Informatika u domeni **Računalno razmišljanje i programiranje** učenik: B.8.1 identificira neki problem iz stvarnoga svijeta, stvara program za njegovo rješavanje, dokumentira rad programa i predstavlja djelovanje programa drugima B.8.2 prepoznaje i opisuje algoritam sortiranja, primjenjuje jedan algoritam sortiranja za rješavanje zadanoga problema u programskom jeziku B.8.3. prepoznaje i opisuje mogućnost primjene rekurzivnih postupaka pri rješavanju odabranih problema te istražuje daljnje mogućnosti primjene rekurzije.
1. RAZRED ili 1. GODINA UČENJA
Nakon prve godine učenja predmeta Informatika u srednjoj školi u domeni **Računalno razmišljanje i programiranje** učenik: B.1.1 analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema B.1.2 primjenjuje jednostavne tipove podataka te argumentira njihov odabir, primjenjuje različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku B.1.3 razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.
Nakon prve godine učenja predmeta Informatika u srednjoj školi u domeni **Digitalna pismenost i komunikacija** učenik: C.1.1 pronalazi podatke i informacije, odabire prikladne izvore informacija te uređuje, stvara i objavljuje/dijeli svoje digitalne sadržaje. C.1.3 u online okruženju surađuje i radi na projektu.
Nakon prve godine učenja predmeta Informatika u srednjoj školi u domeni **e-Društvo** učenik: D.1.3 analizira ulogu koju pomoćna tehnologija i prilagođeni digitalni sadržaji mogu imati u životima osoba s poteškoćama.

2. RAZRED ili 2. GODINA UČENJA
Nakon druge godine učenja predmeta Informatika u srednjoj školi u domeni **Računalno razmišljanje i programiranje** učenik: B.2.1 analizira osnovne algoritme s jednostavnim tipovima podataka i osnovnim programskim strukturama i primjenjuje ih pri rješavanju novih problema. B.2.2 u zadanome problemu uočava manje cjeline, rješava ih te ih potom integrira u jedinstveno rješenje problema. B.2.4 u suradnji s drugima osmišljava algoritam, implementira ga u odabranome programskom jeziku, testira program, dokumentira i predstavlja drugima mogućnosti i ograničenja programa.
Nakon druge godine učenja predmeta Informatika u srednjoj školi u domeni **Digitalna pismenost i komunikacija** učenik: C.2.1 u suradničkome *online* okruženju na zajedničkome projektu istražuje utjecaj ugradnje računalnih sustava u razne uređaje na svakodnevni život C.2.2 analizira programe s obzirom na licenciju i na preduvjete za instalaciju programa.
Nakon druge godine učenja predmeta Informatika u srednjoj školi u domeni **e-Društvo** učenik: D.2.2 analizira i procjenjuje utjecaj informacijske i komunikacijske tehnologije na učinkovitost i produktivnost u raznim područjima i poslovima.
3. RAZRED ili 3. GODINA UČENJA
Nakon treće godine učenja predmeta Informatika u srednjoj školi u domeni **Računalno razmišljanje i programiranje** učenik: B.3.3 koristeći neki grafički modul vizualizira i grafički prikazuje neki problem iz svoje okoline B.3.5 definira problem iz stvarnoga života i stvara programsko rješenje prolazeći sve faze programiranja. Predstavlja programsko rješenje i vrednuje ga.
4. RAZRED ili 4. GODINA UČENJA
Nakon četvrte godine učenja predmeta Informatika u srednjoj školi u domeni **Računalno razmišljanje i programiranje** učenik: B.4.4 definira problem iz stvarnoga života i stvara programsko rješenje prolazeći sve faze programiranja. Predstavlja programsko rješenje i vrednuje ga.

2. RAZRED PRIRODOSLOVNO-MATEMATIČKE GIMNAZIJE (A i B)
Nakon druge godine učenja predmeta Informatika u srednjoj školi u domeni **Informacije i digitalna tehnologija** učenik: A.2.5 istražuje različite vrste ulaznih i izlaznih podataka te pretvorbu u oblik pogodan za računalnu obradu.
Nakon druge godine učenja predmeta Informatika u srednjoj školi u domeni **Računalno razmišljanje i programiranje** učenik: B.2.3 razlikuje složene tipove podataka u zadanome programskom jeziku te pri rješavanju problema koristi se funkcijama i metodama definiranima nad njima.
4. RAZRED PRIRODOSLOVNO-MATEMATIČKE GIMNAZIJE (A i B)
Nakon četvrte godine učenja predmeta Informatika u srednjoj školi u domeni **Informacije i digitalna tehnologija** učenik: A.4.1 istražuje mogućnosti različitih programskih jezika.

U Kurikulumu nema ishoda povezanih s robotikom ili programiranjem fizičkih objekata, samo se ponegdje spominju u preporukama za ostvarenje odgojno-obrazovnih ishoda. Ipak, izradom programa iz svakodnevnog života zasigurno se razvijaju vještine računalnog razmišljanja i programiranja te je time čak moguće obuhvatiti sve četiri domene i sve cikluse, kako je i vidljivo iz izdvojenih ishoda. Također, radom s mikrokontrolerima, senzorima i strujom, moguće je ostvariti poveznice i s drugim predmetima, poput Fizike i Matematike. U aktivnostima opisanima u ovom radu moguće je ostvariti sljedeći ishod iz Kurikuluma nastavnog predmeta Fizika: FIZ SŠ C.2.7. Primjenjuje zakone elektrodinamike u električnom strujnom krugu.

U današnje vrijeme dostupni su razni softverski alati prilagođeni mlađim uzrastima (npr. Scratch, mBlock) u kojima se učenici mogu upoznati s programiranjem fizičkih objekata i interakcijom s njima. Upotrebom blokovskog programiranja pojednostavljena je sama izrada programa i time približena većem broju učenika.

U srednjim školama, posebice u razredima inačica A i B prirodoslovno-matematičkih gimnazija moguća je izrada zahtjevnijih programa sa zahtjevnijim alatima, stoga ćemo se u ovom radu usmjeriti na aktivnosti izvedive u srednjoj školi na platformi Arduino.

Poglavlje 2

2. Pločica Arduino

2.1 Općenito o Arduinu

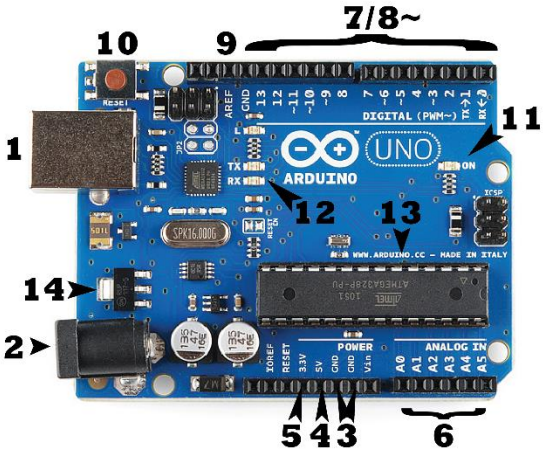
Arduino je naziv za elektroničku platformu otvorenog koda i pločice s mikrokontrolerima dizajnirane za jednostavnu izradu interaktivnih i programabilnih projekata. Platforma obuhvaća hardverske i softverske komponente te je zbog besplatnog i otvorenog koda pogodna za korištenje i daljnje razvijanje za ljude raznih tehničkih kompetencija.

Arduino pločice sadrže mikrokontroler koji omogućuje upravljanje pločicom, njenim ulaznim i izlaznim jedinicama te povezivanje s računalom. U besplatnom softverskom alatu Arduino IDE (Integrated Development Environment) moguće je napisati, kompilirati te učitati program na Arduino pločice. Za pisanje programa koristi se pojednostavljena verzija jezika C++ s uključenom bibliotekom koja omogućuje jednostavan rad sa senzorima, zaslonima, motorima i ostalim elektroničkim komponentama koje je moguće povezati s pločicama. Najnovija verzija softvera je Arduino IDE 2.2.1 te se može instalirati na Windows, Macintosh OSX i Linux operacijske sustave.

Dostupni su brojni senzori i ostali hardverski dijelovi koji se mogu povezati s pločicama, putem kojih je moguće ostvariti komunikaciju između ulaznih i izlaznih jedinica pločice. Primjerice, očitavanjem ulaznih podataka sa senzora svjetla, udaljenosti ili temperature može se upravljati LED diodama, zaslonom ili motorom. Tijekom godina izdano je više od 100 hardverskih dijelova koji obuhvaćaju pločice Arduino Nano, MKR, UNO, Leonardo, Micro, Mega, Due itd. Prva u nizu USB Arduino pločica je Arduino UNO koja je dobila ime po talijanskoj riječi za broj jedan kako bi označila prvo izdanje softvera Arduino IDE 1.0. Arduino UNO se zadržala kao najviše korištena iz obitelji Arduino pločica jer je jednostavna za korištenje, popularna zbog brojnih programa i projekata koji su napravljeni pomoću nje i dostupni na internetu, što je čini i pogodnom za početnike, učitelje i učenike kao uvod u robotiku.

U ovom radu predstavit ćemo izradu nekoliko programa na Arduino UNO pločici.

2.2 Pločica Arduino UNO



Slika 2.1: Arduino UNO

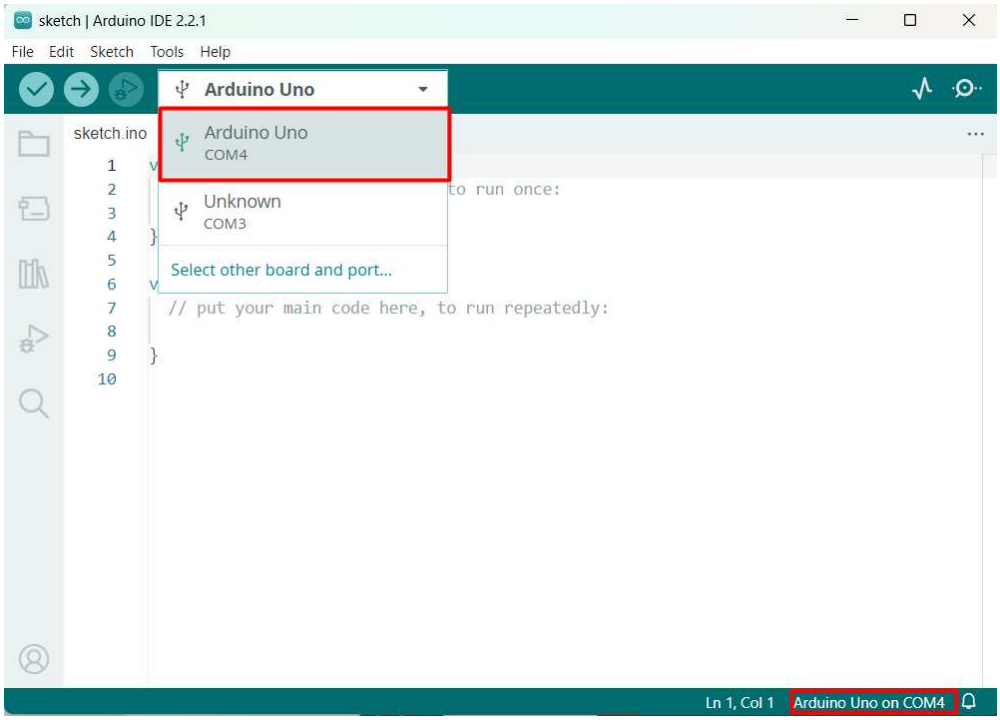
Svaka se Arduino pločica mora moći povezati s izvorom napajanja. Arduino UNO pločica može primati napon iz dva izvora: putem USB kabela povezanog u računalo i izvora označenog na slici s **(1)**, putem vanjskog punjača i izvora na slici označenog s **(2)**.
Izvodi pločice (*pins*) su mjesta koja povezujemo spojnim žicama s matičnom pločom ili vanjskim komponentama da bismo dobili strujni krug.
Izvod **(3)** je GND ili uzemljenje (*Ground)* te ih na pločici ukupno ima tri. Pod **(4)** je izvod 5V koji omogućuje 5 volti, a pod **(5)** izvod 3.3V koji omogućuje 3.3 volta.
Izvodi označeni brojem **(6)** su analogni izvodi (od A0 do A5) i služe za čitanje signala s analognih senzora.
Na drugoj strani pločice označeni brojem **(7)** su digitalni izvodi (od 0 do 13). Mogu se koristiti za digitalno čitanje ili slanje signala.
Pokraj nekih digitalnih izvoda **(8)** nalazi se znak ~. Ti se izvodi ponašaju kao i ostali digitalni izvodi, ali se mogu koristiti i za PWM (Pulse-Width Modulation), tj. mogu simulirati analogni izlaz.
Pod brojem **(9)** je izvod AREF, odnosno Analog Reference.
Arduino pločica se može ponovno pokrenuti (*reset*) isključivanjem iz struje i ponovnim uključivanjem ili pritiskom na gumb **(10)**. Time se program koji je učitan na pločicu počinje izvršavati ispočetka.
Pod brojem **(11)** nalazi se LED dioda koja bi trebala svijetliti kad god je pločica upaljena, tj. ima struje.
LED diode pod brojem **(12)**, TX (*transmit*) i RX (*receive*) služe kao potvrda da se podatci šalju ili primaju putem Arduina, kao kad se učitava novi program.
ATMega mikrokontroler označen brojem **(13)** upravlja Arduinom i omogućuje učitavanje i izvršavanje programa.
Pod brojem **(14)** nalazi se regulator napona koji kontrolira količinu napona koja ulazi na pločicu.

2.3 Softverski alat Arduino IDE

Arduinovim pločicama moguće je upravljati iz raznih softverskih alata (poput mBlocka ili Tinkercada), u kojima je omogućeno čak i programiranje blokovima za jednostavniju izradu programa. Ipak, složeniji se projekti najčešće rade u Arduino IDE alatu, koji će se koristiti i u ovom radu.

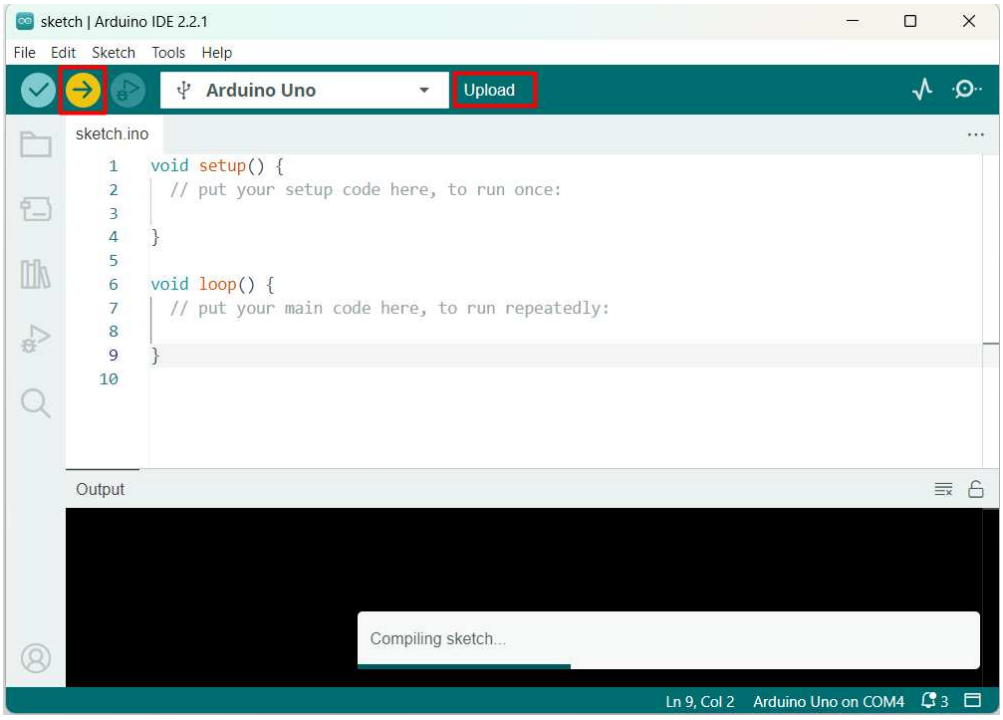
Početni kod svakog Arduino programa sadrži *setup()* i *loop()* funkcije.
Funkcija *setup()* poziva se na početku programa i služi za konfiguraciju digitalnih izvoda, inicijaliziranje varijabli, uključivanje dodatnih biblioteka i slično. Izvršava se samo jednom, nakon uključenja Arduino pločice.
Funkcija *loop()* koristi se za interaktivan rad s pločicom jer se izvršava kao beskonačna petlja pa se u njoj nalazi glavna logika programa.
U ostatku programa mogu se inicijalizirati globalne varijable te pisati nove funkcije koje se onda pozivaju unutar *setup()* i *loop()* funkcija. Ukoliko program nema jednu od ove dvije funkcije, neće se moći kompilirati.
Konfiguracija digitalnih izvoda u *setup()* funkciji radi se pozivom funkcije *pinMode(pin, mode)* kojom se određeni izvod (*pin*) postavi kao izlazna ili ulazna jedinica (*OUTPUT* ili *INPUT)*. Analogne izvode nije potrebno konfigurirati funkcijom *pinMode()* jer su oni svi ulazni.
Funkcije koje se najčešće koriste unutar funkcije *loop()* su *digitalRead(pin)* i *digitalWrite(pin, value)*. Ako je izvod konfiguriran kao izlaz, koristi se *digitalRead(pin)* za dobivanje vrijednosti koje šalje komponenta spojena na taj izvod. Budući da je riječ o digitalnom izvodu, vrijednosti mogu biti 0 ili 1 (*LOW* ili *HIGH*). Funkcija *digitalWrite(pin, value)* postavlja vrijednost (*value*) ulazne komponente (ponovno 0 ili 1) na izvodu kojeg prima kao parametar. Može se koristiti i na izlaznim komponentama za upravljanje internim otpornicima, ali u ovom je radu nećemo tako koristiti. Za analogne izvode koristi se funkcija *analogRead(pin)* koja vraća cjelobrojne vrijednosti između 0 i 1023.
Važna je još i funkcija *delay(ms)* koja zaustavlja izvođenje programa za određeno vrijeme definirano kao parametar te funkcije (u milisekundama). Ova funkcija ima razne namjene te je osobito važna kod čitanja vrijednosti s ulaznih jedinica, kao što će biti prikazano u rješenjima nekih programa.

Program napisan u Arduino IDE naziva se još i *sketch* te se sprema s ekstenzijom *.ino*. Pločicu je potrebno USB kabelom povezati s računalom, zatim odabrati odgovarajući port u Arduino IDE softveru.



Slika 2.2: Povezivanje pločice s Arduino IDE softverom

Program se preuzima na pločicu klikom na ikonu izbornika *Upload*. Program se počinje izvršavati čim se kod učita na mikrokontroler pločice.



Slika 2.3: Preuzimanje programa na Arduino UNO pločicu

Pokraj gumba *Upload* nalazi se i gumb *Verify.* Pritiskom gumba *Verify* odvija se sličan proces, kompilator prolazi kroz kod, provjerava ima li grešaka te ga kompilira, ali ne učitava program na pločicu. Preporuča se koristiti *Verify* za vrijeme pisanja koda kako bi se eventualne greške mogle otkloniti na vrijeme, prije učitavanja na pločicu.

Jedan od vrlo korisnih alata dostupnih u Arduino IDE je Serial Monitor. Služi za *debuggiranje*, testiranje i direktnu komunikaciju Arduino pločice s računalom. Veza sa Serial Monitorom uspostavlja se u *setup()* funkciji, pozivom funkcije *Serial.begin(9600),* gdje broj 9600 predstavlja najveći broj bitova koji se mogu prenositi u sekundi. Funkcijama *Serial.print()* i *Serial.println()* u Serial Monitoru se mogu ispisivati vrijednosti očitane s raznih senzora spojenih na pločicu.

Poglavlje 3

3. Aktivnosti

3.1 Upravljanje svjetlom

Opis aktivnosti: Cilj ove aktivnosti je upoznati učenike s radom na Arduino UNO pločici, što obuhvaća fizičko povezivanje komponenti te programsko rješavanje zadatka. Dano je ukupno pet zadataka koji se tiču upravljanja gumbom, svjetlom i potenciometrom. Počevši s jednostavnim zadatcima i nadograđujući ih do složenijih, učenici će otkriti razliku između analognih i digitalnih podataka te slanjem i čitanjem podataka kontrolirati ulazne i izlazne komponente Arduino UNO pločice. Aktivnost je primjerena za 2. razred srednje škole.

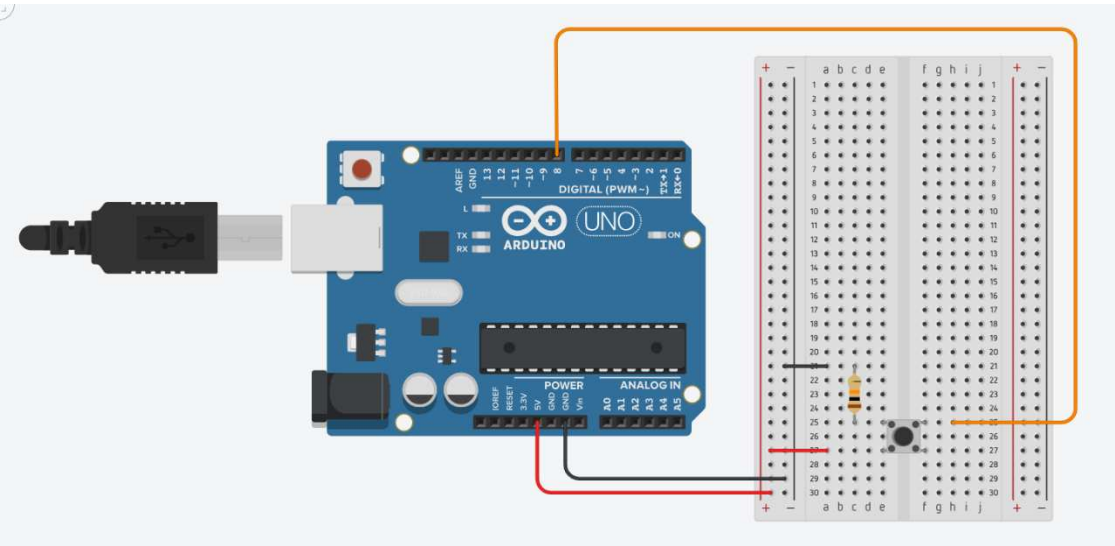
Ishodi aktivnosti: A.2.5 istražuje različite vrste ulaznih i izlaznih podataka te pretvorbu u oblik pogodan za računalnu obradu.
B.2.2 u zadanome problemu uočava manje cjeline, rješava ih te ih potom integrira u jedinstveno rješenje problema.
B.2.4 u suradnji s drugima osmišljava algoritam, implementira ga u odabranome programskom jeziku, testira program, dokumentira i predstavlja drugima mogućnosti i ograničenja programa.

Potreban materijal:
- Arduino UNO
- matična ploča
- 1 LED dioda
- 1 otpornik otpora 220 Ω i 1 otpornik otpora 10 kΩ
- tipka
- potenciometar
- spojne žice
- USB kabel

Zadatak 1:
Poveži tipku s digitalnim izvodom pločice Arduino UNO, zatim pomoću Serial Monitora istraži koje se vrijednosti očitavaju s tipke u njenim različitim stanjima.

Rješenje:
Matičnu ploču povezujemo s Arduino UNO pločicom tako da izvod pločice označen s 5V povežemo crvenom žicom s linijom matične ploče označene znakom +. Liniju

označenu znakom – povezujemo crnom žicom s izvodom GND na Arduino UNO pločici. Tipku postavljamo na matičnu ploču kao što je vidljivo na slici. Tipka ima četiri žice od kojih su po dvije nasuprotne povezane. Jednu žicu tipke povežemo na 5V na matičnoj ploči, a njoj susjednu žicu na uzemljenje, tj. izvod GND. Tipka je ulazna jedinica, stoga za nju koristimo otpornik otpora 10 kΩ kako bi se što preciznije mogao očitati unos podataka s tipke. Njoj nasuprotnu žicu povezujemo spojnom žicom za digitalni izvod Arduino UNO pločice označen brojem 8.



Slika 3.1

Program:

```
int tipka = 8; // inicijalizacija varijable kojoj pridružujemo vrijednost digitalnog izvoda povezanog s tipkom

void setup() {
  Serial.begin(9600); // uspostavljanje komunikacije između mikrokontrolera i računala
  pinMode(tipka, INPUT); // konfiguracija izvoda tipke na unos
}

void loop() {
  Serial.println(digitalRead(tipka)); // ispis stanja koje se čita s izvoda tipke
  delay(250); // zaustavljanje programa na 250 ms
}
```

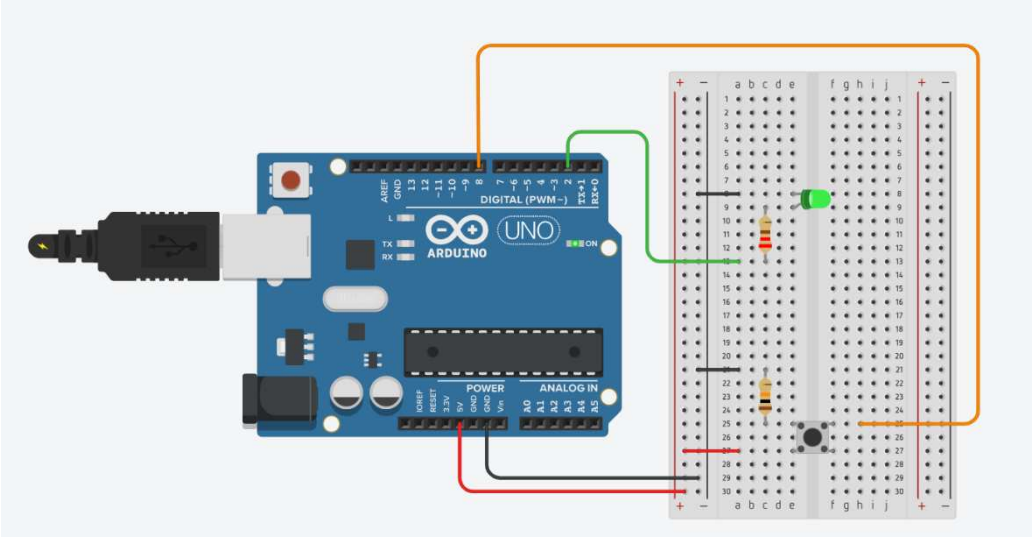
Nakon što se odradi inicijalizacija varijable i konfiguracija u funkciji *setup()*, program se dalje izvršava stalnim pozivima funkcije *loop()*. Kako bi se u Serial Monitoru mogle evidentirati promjene vrijednosti tipke, potrebno je u *loop()* funkciji pozvati funkciju

digitalRead(tipka) te dobivenu vrijednost ispisati na Serial Monitoru. Funkcija *loop()* se jako brzo izvršava pa nakon svakog ispisa kratko zaustavljamo program pozivom funkcije *delay(250),* čime jasnije vidimo promjene u ispisu. Dok tipka nije pritisnuta u Serial Monitoru se treba ispisivati broj 0, a kad se tipka pritisne broj 1.

Slika 3.2: Ispis očitanih digitalnih vrijednosti gumba u Serial Monitoru

Zadatak 2:
Napravi program u kojemu upravljaš LED diodom pomoću tipke, tako da na LED diodi bude upaljeno svjetlo dok je tipka pritisnuta, a ugašeno kad tipka nije pritisnuta.

Rješenje:
Na pločici postoji integrirana LED dioda na izvodu 13. Radi preglednosti, mi ćemo dodati LED diodu na matičnu ploču i upravljati njom, iako se program može napraviti tako da koristi diodu s izvoda 13.
Na matičnu ploču sastavljenu za prethodni zadatak dodajemo zelenu LED diodu te njenu kraću žicu (katodu) povezujemo s uzemljenjem na matičnoj ploči. LED diode ne mogu biti izravno povezane na 5V jer bi mogle eksplodirati pa dužu žicu LED diode (anodu) putem otpornika otpora 220 Ω povezujemo s digitalnim izvodom pločice označenim brojem 2.



Slika 3.3

Program:

```
int tipka = 8;
int led = 2; // inicijalizacija varijable kojoj pridružujemo vrijednost digitalnog izvoda povezanog s LED diodom

void setup() {
  pinMode(tipka, INPUT);
  pinMode(led, OUTPUT); // konfiguracija izvoda LED diode na izlaz
}

void loop() {
  if (digitalRead(tipka) == HIGH) {
    digitalWrite(led, HIGH);
  }
  else {
    digitalWrite(led, LOW);
  }
}
```

Kao i u prethodnom primjeru, na početku programa odrađuje se inicijalizacija varijabli i konfiguracija izvoda. Ponovno želimo stalno iznova čitati vrijednosti s izvoda tipke pa u funkciji *loop()* pozivamo funkciju *digitalRead(tipka)* i provjeravamo je li vraćena vrijednost *HIGH* ili *LOW* pa ujedno postavljamo istu vrijednost na izvod LED diode. Funkciju *loop()* možemo kraće napisati i na sljedeći način:

```
void loop() {
  digitalWrite(led, digitalRead(tipka)); }
```

Zadatak 3:
Napravi program koji simulira rad prekidača svjetla. Pri početnom izvršavanju programa, svjetlo LED diode treba biti ugašeno. Pritiskom tipke mijenja se stanje LED diode, tj. svjetlo se pali i ostaje upaljeno dok sljedeći pritisak tipke ne promijeni stanje LED diode i time ugasi svjetlo.

Rješenje:
Koriste se Arduino UNO pločica i matična ploča povezane sa svim komponentama iz prethodnog zadatka.

Program:

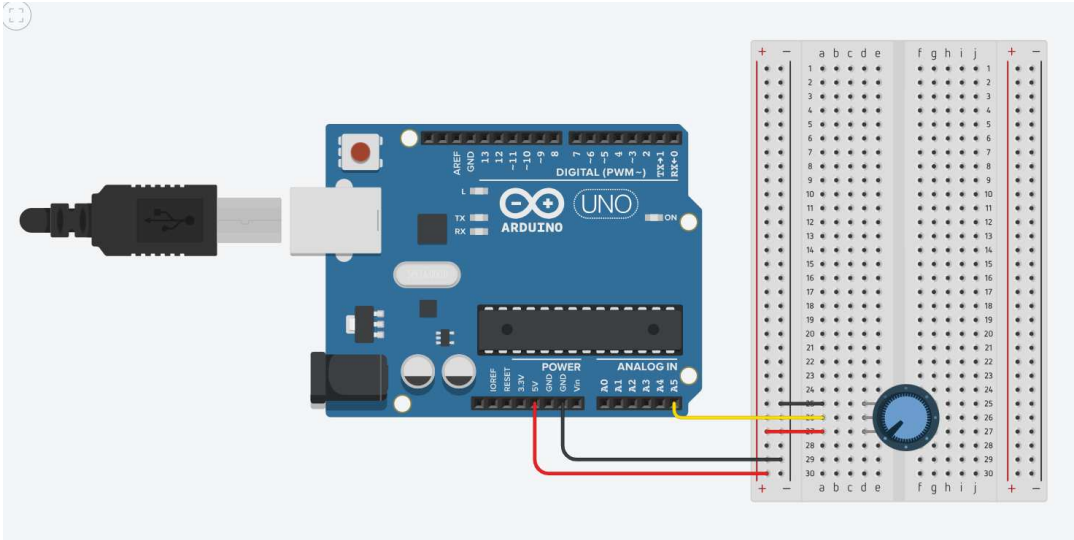
```
int tipka = 8;
int led = 2;

// inicijalizacija varijabli u kojima će se pamtiti stanje komponenti
// na početku programa tipka nije stisnuta i LED dioda nije upaljena pa
vrijednosti postavljamo na 0
int tipkaProsloStanje = 0;
int tipkaTrenutnoStanje = 0;
int ledStanje = 0;

void setup() {
  pinMode(tipka, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  tipkaTrenutnoStanje = digitalRead(tipka);
  if (tipkaProsloStanje == 1 && tipkaTrenutnoStanje == 0) {
    if (ledStanje == 0) {
      digitalWrite(led, HIGH);
      ledStanje = 1;
    }
    else {
      digitalWrite(led, LOW);
      ledStanje = 0;
    }
  }
  tipkaProsloStanje = tipkaTrenutnoStanje; // potrebno je ažurirati prošlo
stanje tipke prije nego program ponovno pročita sljedeće trenutno stanje
}
```

Da bismo mogli evidentirati promjenu stanja tipke i LED diode, inicijaliziramo nove globalne varijable kojima ćemo pridruživati prošlo i trenutno stanje tipke, kao i trenutno stanje LED diode. U funkciji *loop()* svako čitanje vrijednosti s izvoda tipke spremamo u varijablu *tipkaTrenutnoStanje*. Kao što je to najčešće slučaj kod lampi s ovakvom vrstom tipke, želimo da se promjena svjetla dogodi kad se pritisnuta tipka otpusti. To znači da moramo provjeravati je li prošlo stanje tipke 1 (*HIGH*) i trenutno stanje tipke 0 (*LOW*). Ukoliko je, provjeravamo vrijednost zapisanu u varijabli *ledStanje* pa pozivom funkcije *digitalWrite()* šaljemo suprotnu vrijednost na izvod LED diode.

Ovako napisano programsko rješenje ne daje greške, ali ne radi uvijek kako bi trebalo. Problem je što su kod fizičke izvedbe programa moguća odstupanja kod korištenja ulaznih mehaničkih komponenti. Naime, tipka se ponaša kao sklopka koja zatvara strujni krug kad se pritisne. U trenutku pritiskanja tipke može doći do odstupanja vrijednosti od očekivanih, kada signal kratko vrijeme poprima nasumične vrijednosti *LOW* i *HIGH* dok se ne stabilizira. Kod ovakvih jednostavnijih programa, taj se problem može riješiti pozivom funkcije *delay(50)* na samom početku ili kraju funkcije *loop()*. Tipka se vrlo brzo stabilizira pa ovo zaustavljanje programa neće biti očito prilikom izvedbe, a omogućit će da se za vrijeme stabiliziranja tipke, nakon njenog pritiska, njene vrijednosti ne očitavaju ponovno.

Zadatak 4:
Poveži potenciometar s analognim izvodom pločice Arduino UNO, zatim pomoću Serial Monitora istraži koje se vrijednosti očitavaju s potenciometra u njegovim različitim stanjima.

Rješenje:
Na matičnu ploču povezanu s pločicom Arduino UNO stavljamo potenciometar. To je mehanički uređaj koji osigurava promjenjivu količinu otpora koja se mijenja dok se njime upravlja. Ima ukupno tri žice, od kojih jednu krajnju spajamo s uzemljenjem, a drugu s naponom 5V matične ploče. Srednja žica služi za povezivanje s analognim izvodom, u ovom slučaju označenim A5.



Slika 3.4

Program:

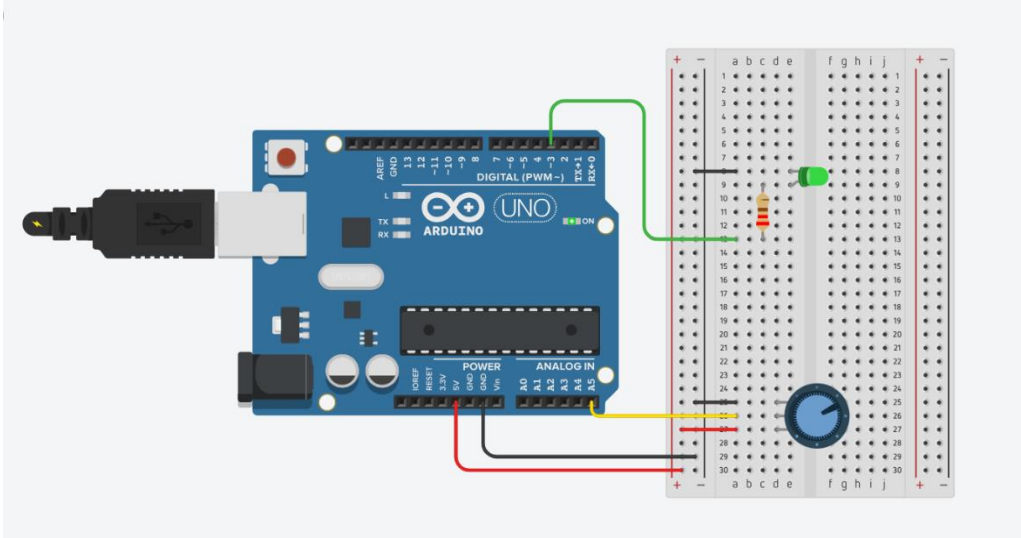
```
int potenciometar = A5; // inicijalizacija varijable kojoj pridružujemo vrijednost analognog izvoda povezanog s potenciometrom

void setup() {
  Serial.begin(9600); // uspostavljanje komunikacije između mikrokontrolera i računala
}

void loop() {
  Serial.println(analogRead(potenciometar)); // ispis stanja koje se čita s izvoda potenciometra
  delay(250); // zaustavljanje programa na 250 ms
}
```

Kao i prilikom očitavanja vrijednosti s tipke, u ovom programu samo uspostavljamo vezu sa Serial Monitorom i u funkciji *loop()* ispisujemo rezultat čitanja vrijednosti s potenciometra. Razlika je u tome što potenciometar povezujemo s analognim izvodom pa kod inicijalizacije varijable *potenciometar* ne koristimo samo broj za označavanje izvoda, već i slovo A. Također, ne možemo pozivati funkciju *digitalRead()* nego *analogRead()*. Zakretanjem potenciometra, u Serial Monitoru je vidiljivo da on šalje analogni signal s vrijednostima u rasponu od 0 do 1023.

Slika 3.5: Ispis očitanih analognih vrijednosti potenciometra u Serial Monitoru

Zadatak 5:
Napravi program koji simulira rad prigušivača svjetla tako da se svjetlina LED diode regulira pomoću potenciometra.

Rješenje:
Na matičnu ploču iz prošlog zadatka dodajemo zelenu LED diodu na isti način kao u zadatku 2.



Slika 3.6

Program:

```
int potenciometar = A5;
int led = 3;
```

```
int potenciometarVrijednost; // deklaracija varijable u koju će se
spremati vrijednost očitana s potenciometra
int svjetlina; // deklaracija varijable koja će definirati svjetlinu LED
diode

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  potenciometarVrijednost = analogRead(potenciometar);
  svjetlina = potenciometarVrijednost / 4; // mapiranje vrijednosti
očitane s potenciometra unutar raspona prikladnog za funkciju analogWrite
  analogWrite(led, svjetlina);
}
```

U ovom programu želimo analogne vrijednosti pročitane s potenciometra poslati na LED diodu. Parametar *value* funkcije *digitalWrite()* može poprimiti samo dvije vrijednosti, *HIGH* ili *LOW*, što znači da ne možemo iskoristiti tu funkciju, već funkciju *analogWrite(pin, value)*. Neki digitalni izvodi mogu služiti i za slanje PWM (Pulse Width Modulation), tj. omogućuju rad s analognim vrijednostima na digitalnim izvodima. Na Arduino UNO pločici to su izvodi 3, 5, 6, 9, 10 i 11. Budući da smo povezali LED diodu za izvod pod brojem 3, moći ćemo koristiti funkciju *analogWrite()*. Kao što smo vidjeli u prethodnom primjeru, potenciometar može slati cjelobrojne vrijednosti u rasponu od 0 do 1023. Međutim, parametar *value* funkcije *analogWrite(pin, value)* može poprimiti samo cjelobrojne vrijednosti u rasponu od 0 do 255. Da bismo omogućili rad sa svim vrijednostima potenciometra, potrebno ih je nekako mapirati na vrijednosti LED diode. Pridruživanje vrijednosti tipa *float* varijabli tipa *int* u jeziku C++ zapravo je zaokruživanje na najveće cijelo, što znači da ćemo dijeljenjem potenciometarVrijednosti brojem četiri dobiti upravo vrijednosti iz raspona 0 – 255. Za slučaj da nismo mogli tako jednostavno riješiti mapiranje vrijednosti, koristili bismo matematičku funkciju *map(value, fromLow, fromHigh, toLow, toHigh)* iz Arduinove biblioteke. Funkcija prima vrijednost koju treba mapirati, kao i donje i gornje granice raspona koji trenutno ima te onoga u koji se treba mapirati.

3.2 Semafor

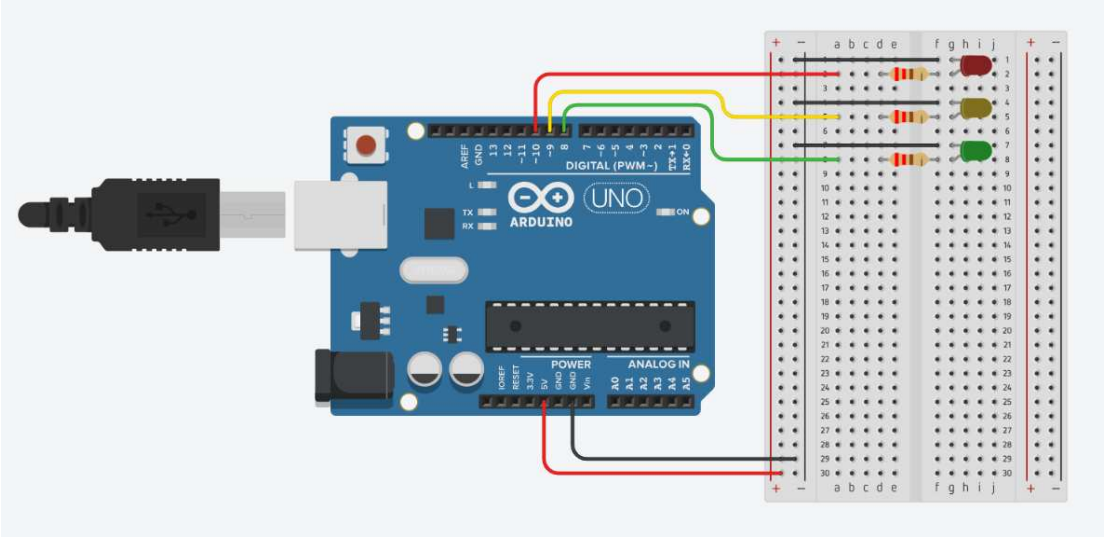
Opis aktivnosti: U ovoj su aktivnosti dana tri zadatka vezana uz izradu semafora koja se međusobno nadograđuju. Cilj je da učenici analiziraju problem, osmisle strategiju rješavanja te izradom vlastitih funkcija prilikom rješavanja omoguće jednostavna buduća unaprjeđenja i nadogradnje programa, kao i jasnoću i preglednost koda. Aktivnost je primjerena za 1. i 2. razred srednje škole.

Ishodi aktivnosti:
B.1.1 analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema.
B.1.3 razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja.
B.2.3 razlikuje složene tipove podataka u zadanome programskom jeziku te pri rješavanju problema koristi se funkcijama i metodama definiranima nad njima.

Zadatak 1:
Napravi program koji simulira rad jednog cestovnog semafora. Pri početnom pokretanju programa treba biti upaljeno crveno svjetlo, a nakon toga se svjetla semafora trebaju izmjenjivati redom ovisno o proteklom vremenu. Napiši funkcije za gašenje i paljenje svjetla.

Potreban materijal:
- Arduino UNO
- matična ploča
- 3 LED diode (jedna crvena, jedna žuta i jedna zelena)
- 3 otpornika otpora 220 Ω
- spojne žice
- USB kabel

Rješenje:
Semafor ćemo vizualno prikazati LED diodama u bojama svjetala semafora koja postavimo na matičnu ploču. Pomoću otpornika i spojnih žica, anode LED dioda povezujemo na digitalne izvode pločice Arduino UNO označene brojevima 10, 9 i 8 kao na slici. Katode LED dioda povezujemo na uzemljenje pločice Arduino UNO, odnosno izvod GND.



Slika 3.7

Program:

```
int crvena = 10;
int zuta = 9;
int zelena = 8;

void setup() {
  pinMode(crvena, OUTPUT);
  pinMode(zuta, OUTPUT);
  pinMode(zelena, OUTPUT);
}

void loop() {
  jednostavniSemafor();
}

void ugasiSvjetlo(int boja){
  digitalWrite(boja, LOW);
}

void upaliSvjetlo(int boja){
  digitalWrite(boja, HIGH);
}

void jednostavniSemafor() {
  upaliSvjetlo(crvena);
  delay(5000);
```

```
  upaliSvjetlo(zuta);
  delay(2000);

  ugasiSvjetlo(zuta);
  ugasiSvjetlo(crvena);
  upaliSvjetlo(zelena);
  delay(5000);

  ugasiSvjetlo(zelena);
  upaliSvjetlo(zuta);
  delay(3000);

  ugasiSvjetlo(zuta);
}
```

Na početku programa inicijaliziramo varijable s nazivima boja LED dioda i vrijednostima digitalnih izvoda kojima pripadaju. U *pinMode()* funkciji unutar funkcije *setup()* pomoću inicijaliziranih varijabli definiramo LED diode kao izlazne jedinice (OUTPUT) jer njima želimo upravljati iz programa tako da ih po potrebi palimo, odnosno gasimo. Kako bi kod bio pregledniji, za paljenje i gašenje LED dioda pišemo dvije nove funkcije, *upaliSvjetlo()* i *ugasiSvjetlo()*, u kojima pozivamo funkciju *digitalWrite()*.

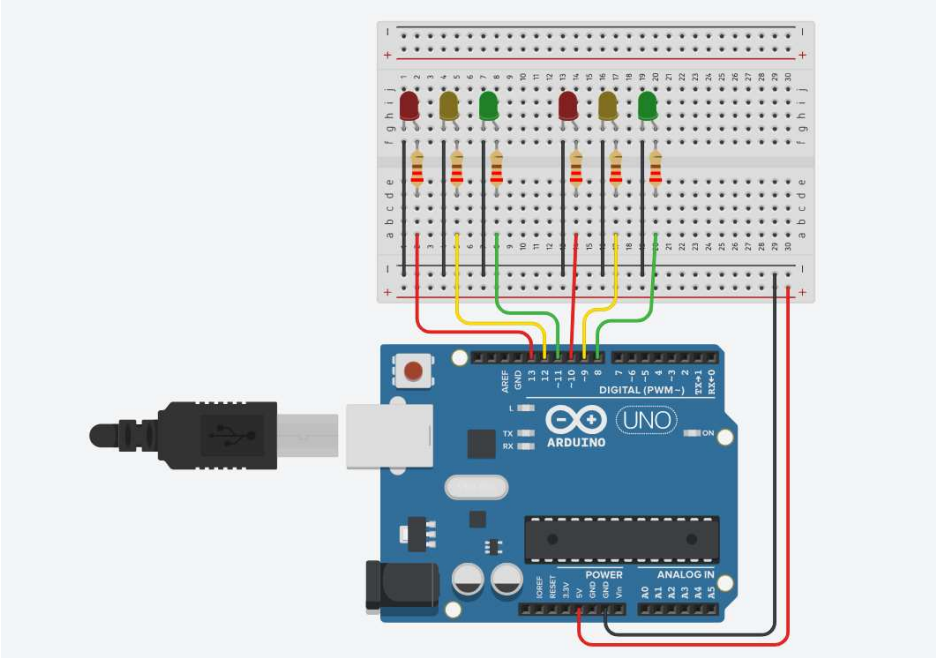
U funkciji *jednostavniSemafor()* pišemo kod koji simulira rad semafora počevši od paljenja crvenog svjetla. Pozivamo funkcije *upaliSvjetlo()* i *ugasiSvjetlo()* redom na LED diodama te na odgovarajućim mjestima funkciju *delay()*. U *loop()* funkciji pozivamo funkciju *jednostavniSemafor()* kako bi se svjetla semafora neprekidno izmjenjivala.

Zadatak 2:
Napravi program koji simulira rad raskrižja s dva cestovna semafora. Semafori trebaju naizmjenično propuštati promet tako da im u isto vrijeme bude upaljeno žuto svjetlo koje jednom semaforu služi kao najava crvenog, a drugom kao najava zelenog svjetla.

Potreban materijal:

- Arduino UNO
- matična ploča
- 6 LED dioda (dvije crvene, dvije žute i dvije zelene)
- 6 otpornika otpora 220 Ω
- spojne žice
- USB kabel

Rješenje:
Na matičnu ploču postavljenu za prethodni zadatak treba dodati još tri LED diode te ih pomoću tri dodatna otpornika i spojne žice na isti način povezati na digitalne izvode pločice označene brojevima 13, 12 i 11.



Slika 3.8

Program:

```
// Prvi semafor
int crvena1 = 10;
int zuta1 = 9;
int zelena1 = 8;

// Drugi semafor
int crvena2 = 13;
int zuta2 = 12;
int zelena2 = 11;

void setup() {
  pinMode(crvena1, OUTPUT);
  pinMode(zuta1, OUTPUT);
  pinMode(zelena1, OUTPUT);

  pinMode(crvena2, OUTPUT);
  pinMode(zuta2, OUTPUT);
  pinMode(zelena2, OUTPUT);
```

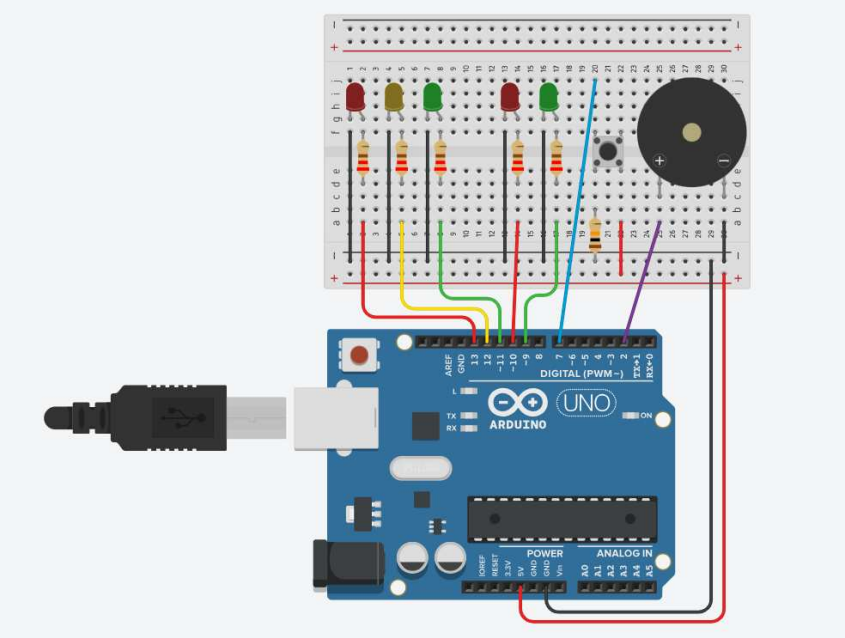
```
}

void loop() {
  raskrizjeSemafor();
}

void ugasiSvjetlo(int boja){
  digitalWrite(boja, LOW);
}

void upaliSvjetlo(int boja){
  digitalWrite(boja, HIGH);
}

void raskrizjeSemafor () {
  upaliSvjetlo(crvena1);
  upaliSvjetlo(zelena2);
  delay(5000);

  upaliSvjetlo(zuta1);
  ugasiSvjetlo(zelena2);
  upaliSvjetlo(zuta2);
  delay(2000);

  ugasiSvjetlo(crvena1);
  ugasiSvjetlo(zuta1);
  upaliSvjetlo(zelena1);
  ugasiSvjetlo(zuta2);
  upaliSvjetlo(crvena2);
  delay(5000);

  ugasiSvjetlo(zelena1);
  upaliSvjetlo(zuta1);
  upaliSvjetlo(zuta2);
  delay(3000);

  ugasiSvjetlo(zuta1);
  ugasiSvjetlo(zuta2);
  ugasiSvjetlo(crvena2);
}
```

Kao i u prethodnom zadatku, na početku programa inicijaliziramo varijable s nazivima boja LED dioda i vrijednostima digitalnih izvoda kojima pripadaju, samo što sada imamo dva semafora sa šest LED dioda pa ukupno inicijaliziramo šest varijabli. Koristimo funkcije *upaliSvjetlo()* i *ugasiSvjetlo()* iz prošlog primjera, ali ovaj put u *loop()* funkciji pozivamo novu funkciju *raskrizjeSemafor().*
U funkciji *raskrizjeSemafor()* pišemo kod koji simulira rad dvaju cestovnih semafora na raskrižju počevši od paljenja crvenog svjetla na prvom i zelenog svjetla na drugom semaforu. Ponovno izmjenjujemo svjetla semafora pozivima funkcija *upaliSvjetlo()*, *ugasiSvjetlo()* i *delay()*. Program završava gašenjem svih upaljenih svjetala, žutog na prvom semaforu te žutog i crvenog na drugom semaforu jer pri sljedećem pozivu funkcija ponovno kreće od paljenja svjetala.

Primjer 3:
Napravi program koji simulira rad cestovnog i pješačkog semafora na pješačkom prijelazu. U izvedbi programa potrebno je koristiti tipku i zujalicu. Na cestovnom semaforu treba stalno biti upaljeno zeleno svjetlo koje se gasi na određeno vrijeme samo pritiskom tipke. Tipka simulira gumb na pješačkom semaforu koji pješaci stisnu kada žele prijeći cestu. Nakon pritiska tipke, potrebno je na određeno vrijeme upaliti zeleno svjetlo na pješačkom semaforu i pomoću zujalice emitirati zvučni signal koji osobama s oštećenjima vida signalizira da mogu prijeći cestu.

Potreban materijal:
- Arduino UNO
- matična ploča
- 5 LED dioda (dvije crvene, jedna žuta i dvije zelene)
- 5 otpornika otpora 220 Ω i 1 otpornik otpora 10 kΩ
- tipka
- zujalica
- spojne žice
- USB kabel

Rješenje:
S matične ploče postavljene za prethodni zadatak uklanjamo žutu LED diodu i na njeno mjesto stavljamo zelenu LED diodu s izvoda pločice pod brojem 8. Dodajemo tipku i jednu njenu žicu pomoću otpornika i spojnih žica povezujemo s izvodom pločice pod brojem 7. Pomoću otpornika povezujemo njoj nasuprotnu žicu za uzemljenje, a njoj susjednu za izvod 5V. Zujalicu povezujemo za uzemljenje i digitalni izvod označen brojem 2.



Slika 3.8

Program:

```
int crvena = 13;
int zuta = 12;
int zelena = 11;

int crvenaPjesacki = 10;
int zelenaPjesacki = 9;

int tipka = 7;
int zujalica = 2;

void setup() {
  pinMode(crvena, OUTPUT);
  pinMode(zuta, OUTPUT);
  pinMode(zelena, OUTPUT);
  pinMode(crvenaPjesacki, OUTPUT);
  pinMode(zelenaPjesacki, OUTPUT);

  pinMode(tipka, INPUT);
  pinMode(zujalica, OUTPUT);

  upaliSvjetlo(zelena);
  upaliSvjetlo(crvenaPjesacki);
}
```

```
void loop() {
  if (digitalRead(tipka) == HIGH) {
      delay(500);
      pjesackiSemafor(10000);
  }
}

void ugasiSvjetlo(int boja){
  digitalWrite(boja, LOW);
}

void upaliSvjetlo(int boja){
  digitalWrite(boja, HIGH);
}

void zvucniSignal(int trajanje){
  tone(zujalica, 500);
  delay(trajanje);
  noTone(zujalica);
}

void pjesackiSemafor(int trajanjeZelenogPjesackog) {
ugasiSvjetlo(zelena);
upaliSvjetlo(zuta);
delay(3000);

ugasiSvjetlo(zuta);
upaliSvjetlo(crvena);
ugasiSvjetlo(crvenaPjesacki);
upaliSvjetlo(zelenaPjesacki);
zvucniSignal(trajanjeZelenogPjesackog);

upaliSvjetlo(zuta);
ugasiSvjetlo(zelenaPjesacki);
upaliSvjetlo(crvenaPjesacki);
delay(2000);

ugasiSvjetlo(zuta);
ugasiSvjetlo(crvena);
upaliSvjetlo(zelena);
delay(3000);
}
```

Na početku programa inicijaliziramo varijable za komponente koje koristimo. U ovom je programu potrebno sedam varijabli, tri za LED diode cestovnog semafora, dvije za LED diode pješačkog semafora, jedna za tipku te jedna za zujalicu. Budući da na cestovnom semaforu stalno treba biti upaljeno zeleno svjetlo do pritiska tipke, unutar funkcije *setup()* ujedno postavljamo početna svjetla semafora pozivima funkcije *upaliSvjetlo()* na zelenom svjetlu cestovnog semafora i crvenom svjetlu pješačkog semafora.
U ovom programu uz funkcije *upaliSvjetlo()* i *ugasiSvjetlo()*, koristimo i novu funkciju *zvucniSignal()* s parametrom *trajanje*. Pozivom te funkcije na zujalici se emitira zvučni signal onoliko milisekundi koliko je proslijeđeno u parametru *trajanje*. Pozivamo postojeću funkciju *tone()* s parametrima digitalnog izvoda zujalice i frekvencijom zvuka u hercima. Kako bi se funkcija *zvucniSignal()* izvršavala onoliko dugo koliko se treba emitirati zvučni signal, pozivamo funkciju *delay()* s parametrom *trajanje*, zatim gasimo zujalicu pozivom postojeće funkcije *noTone()*. Logiku rada programa pišemo u funkciji *pjesackiSemafor()* s parametrom *trajanjeZelenogPjesacki* koji se kasnije prosljeđuje u poziv funkcije *zvucniSignal()*. Funkcija se treba izvršiti u slučaju da je potrebno upaliti zeleno svjetlo na pješačkom semaforu, stoga u njoj pozivamo funkcije *upaliSvjetlo()* i *ugasiSvjetlo()* kako bismo izmijenili svjetla semafora, s tim da nakon paljenja zelenog svjetla na pješačkom semaforu pozivamo i *zvucniSignal(trajanjeZelenogPjesackog)*. Funkcija završava tako da vrati svjetla u početno stanje, tj. upali zeleno svjetlo cestovnog i crveno svjetlo pješačkog semafora. U funkciji *loop()* želimo stalno iznova provjeravati je li tipka pritisnuta što činimo pozivom *digitalRead()* funkcije. Ako funkcija na digitalnom izvodu tipke pročita vrijednost napona kao HIGH, znači da je tipka pritisnuta pa nakon kratkog čekanja inicira izmjenu svjetala pozivom funkcije *pjesackiSemafor()*.

3.3 Digitalni klavir

Opis aktivnosti: Cilj ove aktivnosti je vježbanje rada s tipkama i zujalicom u svrhu izrade jednostavnog digitalnog klavira. Aktivnost je zbog svoje jednostavnosti prikladna za samostalni rad. Ako su učenici uspješno odradili prethodne aktivnosti, ova im aktivnost neće predstavljati problem pa mogu u obliku grupnog rada povezati nekoliko matičnih ploča i tako osigurati veći broj tipki klavira. Aktivnost je primjerena za 1. i 2. razred srednje škole.

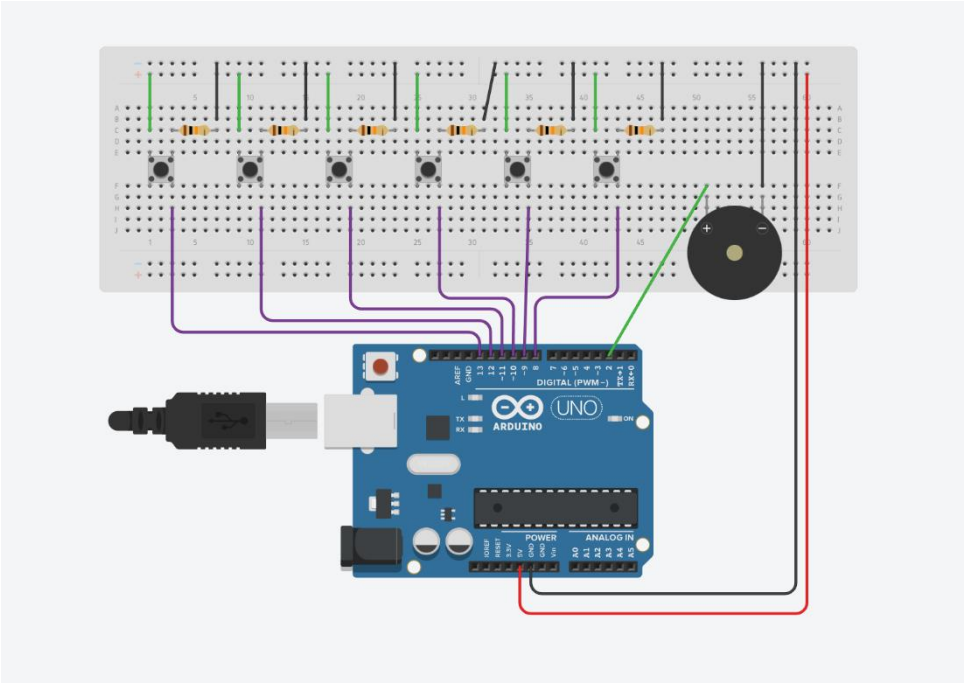
Ishodi aktivnosti:
C.1.1 pronalazi podatke i informacije, odabire prikladne izvore informacija te uređuje, stvara i objavljuje/dijeli svoje digitalne sadržaje.
C.1.3 u online okruženju surađuje i radi na projektu.

Potreban materijal:
- Arduino UNO
- matična ploča

- 6 tipki
- 6 otpornika otpora 10 kΩ
- zujalica
- spojne žice
- USB kabel

Zadatak:
Napravi program koji simulira rad digitalnog klavira s najmanje šest tipki. Istraži koje note klavir treba moći odsvirati za pjesmu „Blistaj, blistaj zvijezdo mala“ pa je pokušaj i odsvirati.

Rješenje:
Za ovaj program koristimo veliku matičnu ploču kako bi na nju moglo stati svih šest tipki i zujalica. Matičnu ploču povezujemo s Arduino UNO pločicom kao i u prethodnim primjerima. Već smo vidjeli i povezivanje tipki s pločicom pa isti princip primjenjujemo i ovdje – svaku tipku povezujemo s digitalnim izvodom, s izvodom 5V te uzemljenjem putem otpornika. Tipke su povezane s digitalnim izvodima pločice označenim brojevima 8, 9, 10, 11, 12 i 13. Zujalicu kao i ranije povezujemo s uzemljenjem i digitalnim izvodom pod brojem 2.



Slika 3.9

Program:

```
// frekvencije nota
int notaA = 440;
```

```
int notaC = 261;
int notaD = 290;
int notaE = 330;
int notaF = 349;
int notaG = 392;

// digitalni izvodi tipki koje sviraju odgovarajuće note
int tipkaA = 13;
int tipkaC = 8;
int tipkaD = 9;
int tipkaE = 10;
int tipkaF = 11;
int tipkaG = 12;
int zujalica = 2;

void setup()
{
  pinMode(zujalica, OUTPUT);
  pinMode(tipkaC, INPUT);
  pinMode(tipkaD, INPUT);
  pinMode(tipkaE, INPUT);
  pinMode(tipkaF, INPUT);
  pinMode(tipkaG, INPUT);
  pinMode(tipkaA, INPUT);
}

void loop()
{
  if (digitalRead(tipkaC) == HIGH){
    tone(zujalica, notaC, 500);
    delay(500);
  }
  if (digitalRead(tipkaD) == HIGH){
    tone(zujalica, notaD, 500);
    delay(500);
  }
  if (digitalRead(tipkaE) == HIGH){
    tone(zujalica, notaE, 500);
    delay(500);
  }
  if (digitalRead(tipkaF) == HIGH){
    tone(zujalica, notaF, 500);
```

```
    delay(500);
  }
  if (digitalRead(tipkaG) == HIGH){
    tone(zujalica, notaG, 500);
    delay(500);
    }
  if (digitalRead(tipkaA) == HIGH){
    tone(zujalica, notaA, 500);
    delay(500);
    }
}
```

Pjesma „Blistaj, blistaj zvijezdo mala“ može se odsvirati pomoću nota A, C, D, E, F i G. Na početku programa inicijaliziramo dvije vrste globalnih varijabli, jednima pridružujemo odgovarajuće frekvencije pripadnih nota, a drugima digitalne izvode tipki koje trebaju moći svirati te note. Potrebna je i varijabla za izvod zujalice kojom će se emitirati odsvirane note. U *setup()* funkciji konfiguriramo izvode pa u *loop()* funkciji čitamo vrijednost svakog digitalnog izvoda na koji smo povezali tipku. Ukoliko je očitana vrijednost *HIGH*, zujalica treba odsvirati tu notu. U ovom programu pozivamo funkciju *tone(pin, frequency, duration)* s trećim parametrom *duration* koji predstavlja vrijeme emitiranja zvučnog signala u milisekundama. To se vrijeme odnosi samo na zujalicu i ne utječe na daljnje izvođenje programa pa ćemo ujedno i zaustaviti program pozivom funkcije *delay()* kako bi se nota mogla odsvirati do kraja prije nego li je neka druga nota prekine.

3.4 Protuprovalni alarmni sustav

Opis aktivnosti: Učenici će otkriti nove komponente za rad s Arduino UNO pločicom, PIR senzor pokreta i RGB LED diodu. Aktivnost je primjerena za 2. i 3. razred srednje škole.

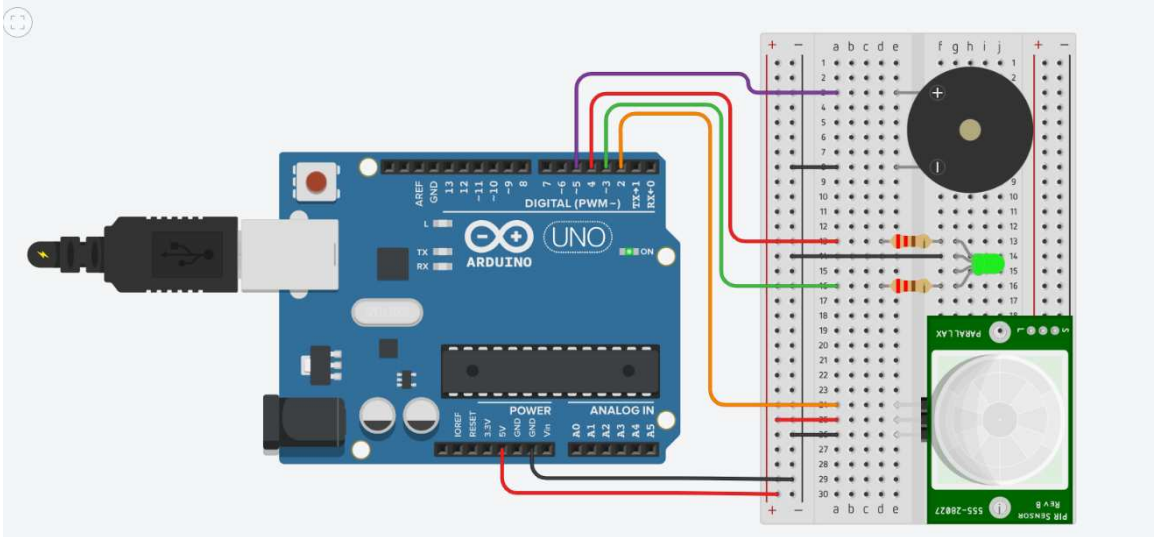
Ishodi aktivnosti:
B.3.5 definira problem iz stvarnoga života i stvara programsko rješenje prolazeći sve faze programiranja. Predstavlja programsko rješenje i vrednuje ga.

Potreban materijal:
- Arduino UNO
- matična ploča
- PIR senzor pokreta
- RGB LED dioda
- zujalica

- 2 otpornika otpora 220 Ω
- spojne žice
- USB kabel

Zadatak:
Napravi program koji simulira rad protuprovalnog alarmnog sustava. Pri početnom pokretanju programa treba biti upaljeno zeleno svjetlo kao znak da nema kretnji. Ako senzor pokreta osjeti kretanje, treba pokrenuti alarmni sustav tako da se emitira zvučni signal na zujalici te promijeni svjetlo u crveno.

Rješenje:
Za izradu programa koristit ćemo PIR senzor pokreta. Senzor ima tri žice, jednu povezujemo s uzemljenjem matične ploče, drugu s izvodom 5V, a treću s digitalnim izvodom na Arduino UNO pločici označenim brojem 2. Od novih komponenti također koristimo RGB LED diodu koja ima ukupno četiri žice. Jedna žica se spaja na uzemljenje matične ploče, a preostale tri na digitalne izvode pločice. Ova se dioda sastoji od tri LED diode (crvena, zelena i plava) pa svaka žica predstavlja jednu od te tri LED diode. Mi ćemo za ovaj program koristiti samo digitalne izvode pločice 3 i 4 za zelenu i crvenu LED diodu, iako je s ovom diodom moguće kombinirati sve tri osnovne boje te tako dobiti bilo koju drugu. Kao i ostale LED diode, treba paziti da prilikom spajanja s izvodima Arduino pločice koristimo otpornike. Preostalo je još povezati zujalicu s uzemljenjem matične ploče i digitalnim izvodom pločice pod brojem 5.



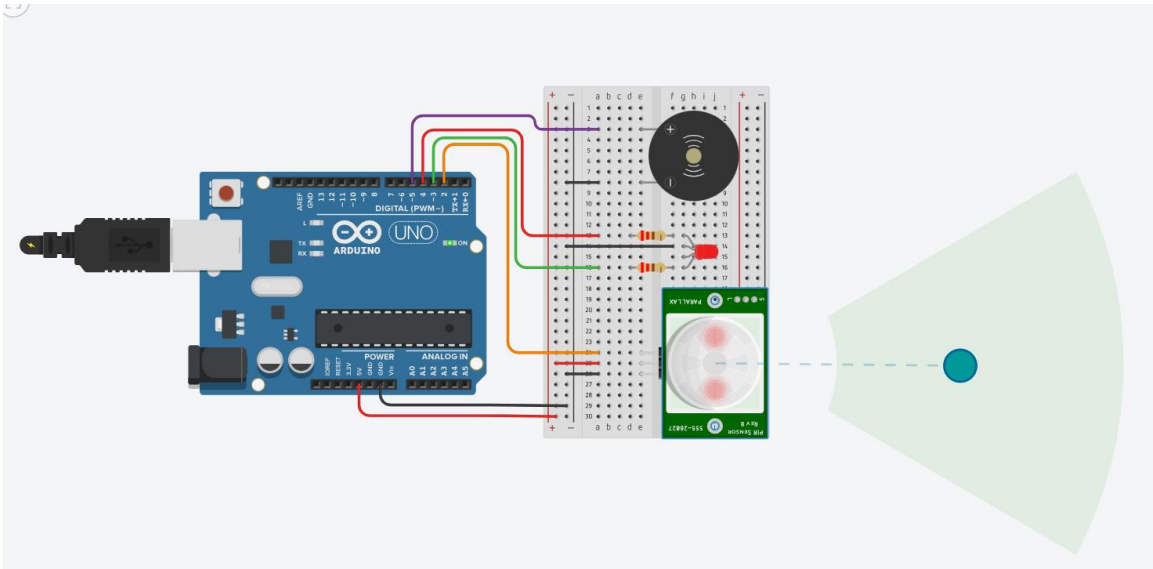
Slika 3.10

Program:

```
int senzorPokreta = 2;
int ledZelena = 3;
int ledCrvena = 4;
```

```
int zujalica = 5;
int senzorVrijednost = 0;

void setup() {
  pinMode(senzorPokreta, INPUT);
  pinMode(ledZelena, OUTPUT);
  pinMode(ledCrvena, OUTPUT);
  pinMode(zujalica, OUTPUT);

  digitalWrite(ledZelena, HIGH); // na početku programa na diodi treba
biti upaljeno zeleno svjetlo
}

void loop(){
  senzorVrijednost = digitalRead(senzorPokreta);
  delay(50);
  if (senzorVrijednost == HIGH)
  {
    digitalWrite(ledCrvena, HIGH);
    digitalWrite(ledZelena, LOW);
    tone(zujalica, 300);
  }
  else if (senzorVrijednost == LOW)
  {
    digitalWrite(ledCrvena, LOW);
    digitalWrite(ledZelena, HIGH);
    noTone(zujalica);
  }
}
```

Na početku programa inicijaliziramo varijable s vrijednostima digitalnih izvoda komponenti spojenih na Arduino UNO te ih u *setup()* funkciji konfiguriramo. Izvod senzora pokreta konfiguriramo kao ulaznu jedinicu. U funkciji *loop()* trebamo stalno čitati vrijednosti sa senzora pokreta uz korištenje funkcije *delay(50)* radi stabilizacije signala. Ako senzor očita kretanje (šalje vrijednost *HIGH* ili 1), potrebno je ugasiti zeleno te upaliti crveno svjetlo. Uz to, pozivom funkcije *tone(pin, frequency)* emitiramo zvučni signal sve dok vrijednost očitana na senzoru pokreta ne zadovolji sljedeći uvjet (bude *LOW*). U slučaju da je vrijednost senzora 0 ili *LOW*, ne detektira se kretanje pa je potrebno ponovno upaliti zeleno svjetlo diode te ugasiti zujalicu pozivom funkcije *noTone(pin)*.



Slika 3.11: Upaljen protuprovalni alarmni sustav

3.5 Automatska klizna vrata

Opis aktivnosti: U ovoj se aktivnosti simulira rad automatskih vrata pomoću dvije nove komponente, ultrazvučnog senzora i servo motora. Iako program nije zahtjevan, za rad sa servo motorom potrebno je poznavanje osnova objektno-orijentiranog programiranja, stoga je ova aktivnost primjerena za učenike 4. razreda prirodoslovno-matematičkih gimnazija koji su s lakoćom savladali prethodne aktivnosti.

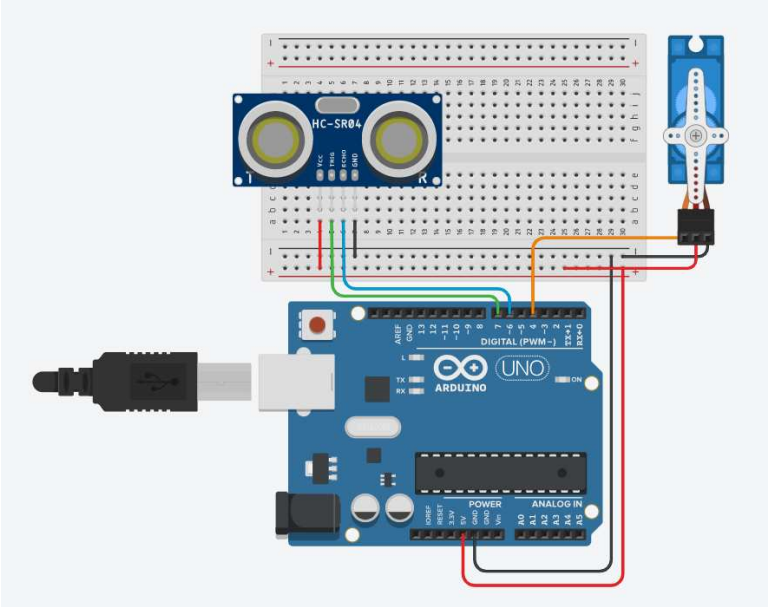
Ishodi aktivnosti:
B.4.4 definira problem iz stvarnoga života i stvara programsko rješenje prolazeći sve faze programiranja. Predstavlja programsko rješenje i vrednuje ga.

Potreban materijal:
- Arduino UNO
- matična ploča
- ultrazvučni senzor
- servo motor
- spojne žice
- USB kabel

Zadatak: Napravi program koji simulira rad automatskih kliznih vrata. Ako se objekt približi vratima (senzoru) na manje od 60 cm, potrebno je otvoriti vrata i držati ih

otvorenima (tj. zakrenuti motor). Kad je objekt udaljen od senzora više od 60 cm, vrata se trebaju zatvoriti (motor se vraća u početno stanje).

Rješenje:
Ultrazvučni senzor ima ukupno četiri žice. Jednu krajnju žicu povezujemo s uzemljenjem, a drugu s naponom 5V. Preostale dvije žice su *echo* i *trigger* i njih povezujemo s digitalnim izvodima Arduino pločice (*echo* povezujemo na izvod 6, a *trigger* na izvod 7). Pomoću izvoda *trigger* šalju se ultrazvučni valovi iz odašiljača, a na izvodu *echo* se osluškuje reflektirani signal. Ultrazvučni senzor radi tako da emitira ultrazvuk koji putuje zrakom i u slučaju da naiđe na prepreku ili neki objekt odbija se natrag do modula. Servo motor ima ukupno tri žice, dvije koje povezujemo s uzemljenjem i naponom i posljednju za povezivanje s digitalnim izvodom pločice (u našem slučaju izvod pod brojem 4).



Slika 3.12

Program:

```
#include <Servo.h>

Servo servoMotor; // deklaracija novog objekta klase Servo koji omogućuje rad s motorom
int servoMotorIzvod = 4;
int echo = 6;
int trigger = 7;
float ocitanaUdaljenost;
float vrijeme;
```

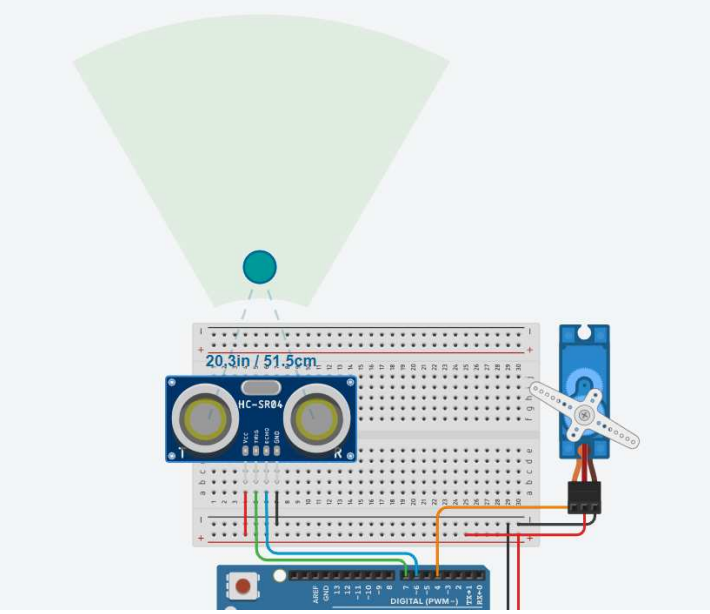
```
void setup()
{
  servoMotor.attach(servoMotorIzvod); // pridruživanje digitalnog izvoda
servo motoru
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
}

void loop()
{
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  // čita izvod echo i vraća vrijeme trajanja zvučnog vala u
mikrosekundama
  vrijeme = pulseIn(echo, HIGH);
  ocitanaUdaljenost = vrijeme * 0.034 / 2;

  if(ocitanaUdaljenost<60)
  {
    servoMotor.write(180);
    delay(3000);
  }
   else{
     servoMotor.write(0);
     delay(50);
   }
}
```

Za rad sa servo motorom potrebno je uključiti u program odgovarajuću biblioteku te deklarirati novi objekt klase Servo. Na objektu *servoMotor* se poziva metoda *attach()* kako bi se postavio odgovarajući digitalni izvod.
Da bi se mogao generirati ultrazvuk potrebno se prvo osigurati da je izvod *triggera LOW*, zatim se postavlja na *HIGH* kako bi otpustio val iz odašiljača, koji se onda odbije od objekta i zabilježi u izvodu *echo*.
Da bismo mogli izračunati udaljenost objekta od senzora, potrebno je pozvati funkciju *pulseln()* koja „sluša“ dani izvod (*echo*) i čeka da poprimi vrijednost HIGH. U tom trenutku počinje mjeriti vrijeme dok se stanje izvoda ne promijeni u LOW.
Koristimo formulu $s = v \cdot t$ kako bismo izračunali udaljenost objekta. Brzina v jednaka je brzini zvuka, otprilike 340 m/s, a vrijeme t je zabilježeno u varijabli *vrijeme* nakon

poziva funkcije *pulseln()*. Vrijeme je dano u mikrosekundama pa je potrebno i brzinu zvuka prilagoditi istoj mjernoj jedinici. Budući da zvuk putuje do objekta i natrag, potrebno je još udaljenost izračunatu primjenom ove formule podijeliti s 2:

```
ocitanaUdaljenost = vrijeme * 0.034 / 2;
```

Unutar *loop()* funkcije još provjeravamo je li očitana udaljenost manja od 60 cm pa u tom slučaju okrećemo kazaljku motora, a u suprotnom je postavljamo na početnu poziciju.



Slika 3.13: Aktiviran rad motora zbog očitane vrijednosti manje od 60 cm

Literatura

[1] Arduino, dostupno na: https://www.arduino.cc (svibanj 2023.).

[2] Arduino – Control LED Brightness With a Potentiometer, dostupno na: https://roboticsbackend.com/arduino-control-led-brightness-with-a-potentiometer/ (rujan 2023.)

[3] Arduino Programming for Beginners: Traffic Light Controller Project Tutorial, dostupno na: https://www.makeuseof.com/tag/arduino-traffic-light-controller/ (svibanj 2023.).

[4] Arduino – Turn LED ON and OFF With Button, dostupno na: https://roboticsbackend.com/arduino-turn-led-on-and-off-with-button/ (rujan 2023.).

[5] Arduino Tutorial 28: Using a Pushbutton as a Toggle Switch, dostupno na: https://www.youtube.com/watch?v=aMato4olzi8 (rujan 2023.)

[6] Automatic Gate open and close Using ultrasonic sensor, dostupno na: https://www.hackster.io/Techatronic/automatic-gate-open-and-close-using-ultrasonic-sensor-ac5579 (rujan 2023.)

[7] Getting Started with the HC-SR04 Ultrasonic sensor, dostupno na: https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1 (rujan 2023.)

[8] How to Make Burglar Alarm Using Arduino Uno, dostupno na: https://linuxhint.com/make-burglar-alarm-arduino-uno/ (rujan 2023.)

[9] Ministarstvo znanosti i obrazovanja, Kurikulum nastavnoga predmeta Informatika za osnovne i srednje škole, dostupno na: https://mzo.gov.hr/UserDocsImages//dokumenti/Obrazovanje/NacionalniKurikulum/PredmetniKurikulumi//Informatika,%20ožujak%202018.%20-%20informatika-6-3-2018.pdf (svibanj 2023.).

[10] S. Fitzgerald, M. Shiloh, The Arduino Projects Book, Torino, 2012.

[11] What is an Arduino?, dostupno na: https://learn.sparkfun.com/tutorials/what-is-an-arduino/whats-on-the-board (rujan 2023.)

Sve slike korištene u trećem poglavlju izrađene su u programu Tinkercad, dostupno na: https://www.tinkercad.com/dashboard (rujan 2023.)

Sažetak

U ovom se radu istražuje mogućnost izrade programa iz svakodnevnog života u nastavi informatike, s ciljem unaprjeđenja vještina računalnog razmišljanja i programiranja uz usklađivanje s ishodima Kurikuluma nastavnoga predmeta Informatika za osnovne i srednje škole. Za realizaciju programa odabrana je Arduino platforma jer je bogata hardverskim komponentama, što potiče kreativnost u izradi projekata. Također, platforma ima veliku podršku online zajednice, čime se uvelike olakšava rad sa svim njenim komponentama. Programi su rađeni na Arduino UNO pločici, koristeći besplatne softverske alate Arduino IDE i Tinkercad.

Glavni dio ovog rada i razrada programa obuhvaćeni su u pet različitih aktivnosti primjerenih za učenike srednjih škola. Aktivnosti obuhvaćaju upravljanje svjetlima, simulaciju semafora, digitalnog klavira, protuprovalnog alarmnog sustava te automatskih kliznih vrata. Rad s Arduino pločicama zahtijeva razumijevanje osnovnih koncepata iz fizike poput strujnih krugova i Ohmovog zakona, stoga ove aktivnosti pružaju i priliku za poveznicama između predmeta Fizike i Informatike u izvedbi nastave.

Ovaj rad naglašava vrijednost učenja putem samostalne izrade programa u nastavi informatike. Sudjelovanjem u ovim aktivnostima, učenici ne samo da usavršavaju svoje vještine programiranja, već stječu i neprocjenjivo iskustvo zbog fizičke izgradnje programa. Promatranjem vizualnih rezultata svog rada, produbljuje se razumijevanje učenika i potiče kritičko razmišljanje, ali i razvoj analitičkih vještina u planiranju izrade.

Summary

This thesis explores the possibility of integrating everyday programs into computer science education, aiming to enhance students' computational thinking and programming skills while aligning with the objectives of the Computer Science Curriculum for primary and secondary schools. The Arduino platform was chosen for this exploration due to it being rich in hardware components, which increase creativity in project development. Additionally, the platform is supported by a strong online community, thus greatly facilitating working with all of its components. The programs were developed on the Arduino UNO board, using free software tools Arduino IDE and Tinkercad.

The core of this thesis and the elaboration of programs are covered in five distinct activities suitable for high school students. These activities include controlling lights, simulating traffic lights, a digital piano, a burglar alarm system and automatic sliding doors. Working with Arduino boards requires an understanding of basic physics concepts, such as electrical circuits and Ohm's law, thereby providing opportunities for interdisciplinary connections between Physics and Computer Science in teaching.

This thesis emphasizes the value of learning through hands-on program creation in computer science education. By engaging in these activities, students not only refine their programming skills but also gain invaluable experience due to the physical construction of programs. Observing visual results of their work deepens students' understanding and promotes critical thinking, as well as the development of analytical skills in project planning.

Životopis

Rođena sam 29. lipnja 1995. godine u Dubrovniku i tamo sam započela svoje školovanje 2001. godine u Osnovnoj školi Marina Držića. S obitelji sam se 2004. godine preselila u Samobor i nastavila obrazovanje u Osnovnoj školi Samobor. Nakon završene osnovne škole, 2009. godine sam upisala prirodoslovno-matematički smjer u Gimnaziji Lucijana Vranjanina u Zagrebu. Po završetku srednje škole upisala sam preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Prebacujem se na preddiplomski sveučilišni studij Matematika; smjer: nastavnički 2016. godine te ga završavam 2019. godine. Iste godine upisujem diplomski sveučilišni studij Matematika i informatika; smjer: nastavnički te 2021. godine dobivam Dekanovu nagradu za izuzetan uspjeh u tom studiju. Za vrijeme diplomskog studija radila sam u informatičkoj tvrtki i osnovnoj školi.