

SVEČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5590

**Programska podrška za muzički
sekvencer**

Matko Martinić

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 14. ožujka 2018.

ZAVRŠNI ZADATAK br. 5590

Pristupnik: **Matko Martinić (0036497019)**
Studij: Elektrotehnika i informacijska tehnologija
Modul: Elektroničko i računalno inženjerstvo

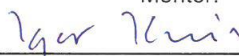
Zadatak: **Programska podrška za muzički sekvencer**

Opis zadatka:


Korištenjem razvojne ploče Raspberry Pi 3 model B te programskog jezika Python potrebno je napisati i testirati programsku podršku za muzički sekvencer. Programska podrška treba upravljati poljem LED-ova te detekcijom odabira željene melodije od strane korisnika. Na temeniju odabira potrebno je na izlazno audio pojačalo poslati odgovarajuću melodiju.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

Mentor:


Izv. prof. dr. sc. Igor Krois

Djelovođa:


Prof. dr. sc. Dražen Jurišić

Predsjednik odbora za
završni rad modula:

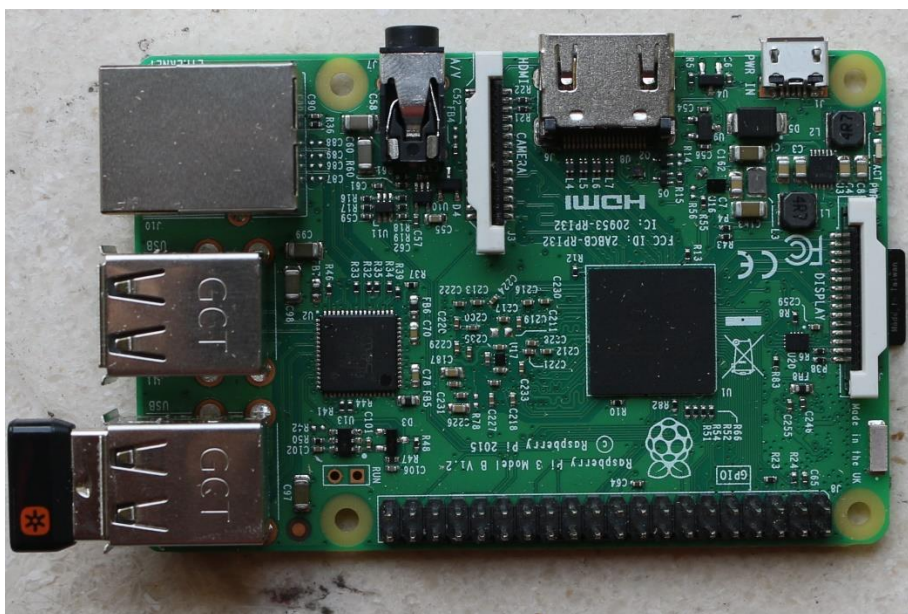

Prof. dr. sc. Mladen Vučić

SADRŽAJ

1. Uvod	1
2. Opis uređaja	3
3. Upravljanje LED diodama	4
3.1 Prvi program	6
4. Programiranje senzora	10
4.1 Prvi program	11
4.2 Drugi program	13
4.3 Treći program	15
4.4 Četvrti program	17
4.5 Završni program	20
5. Zaključak	26
Literatura	27

1. Uvod

Muzički sekvencer je uređaj koji služi glazbenicima za stvaranje glazbe. Glazbenici ga koriste kako bi brzo i efikasno isprobali određene ritmove i tonove. Svaki red odgovara jednom tonu koji se unaprijed dodijelio, a svaki stupac se pali određenom brzinom. Uređaj ima polje LED dioda (naš primjerak ima polje od 8×8 LED-ica), čiji stupci se pale jedan za drugim, može se reći kao „val“, i ovisno o tome gdje u tome polju, odnosno stupcu, je postavljen senzor uređaj proizvede određeni zvuk. Prikladno zahtjevima zadatka izabrao sam Raspberry Pi 3 model B (slika 1) mikrokontroler. Raspberry Pi je mikrokontroler koji sadrži GPIO pinove, pinove koji se koriste za napajanje sklopovlja, USB priključke, audio priključnicu, HDMI priključnicu za spajanje na monitor te mikro USB za vlastito napajanje. Od navedenog se koristi: priključak za napajanje mikrokontrolera, USB i audio priključak na koje su spojeni zvučnici, 13 GPIO pinova spojeni na sklopovlje i pin za napajanje od 5 volti i masa. U daljnjem tekstu ću opisivati razvoj programa kojeg sam podijelio u dva koraka. Prvi korak je bio napisati program koji upravlja LED-icama, a drugi korak je bio napisati program koji će upravljati senzorima, odnosno ovisno o redcima u kojima je stavljen senzor pustiti odgovarajući zvuk.



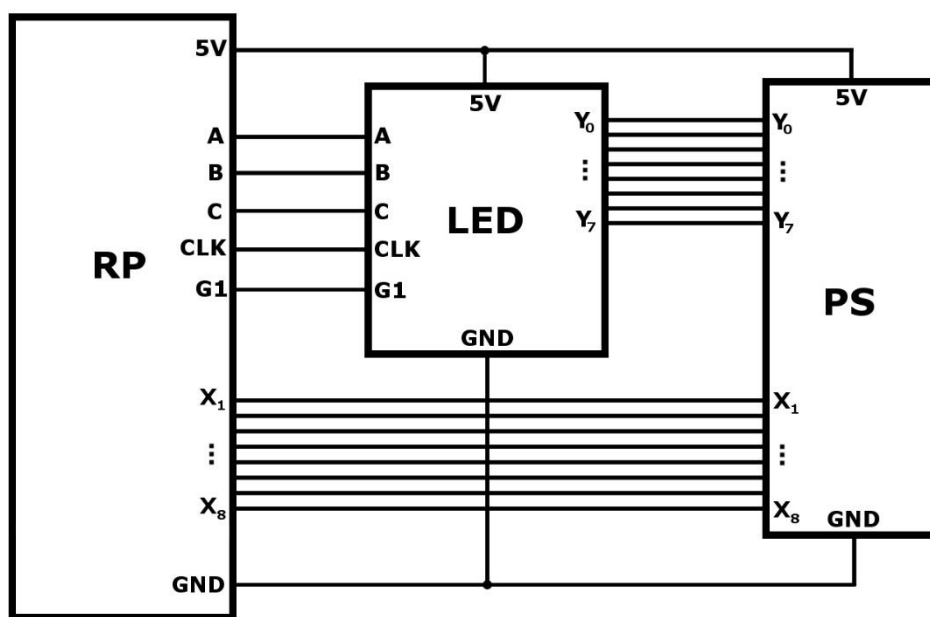
Slika 1
Raspberry Pi 3 model B

Specifikacije mikrokontrolera:

- Broadcom procesor (ARMv8) 64-bit 1.4GHz
- 1GB RAM-a
- Wifi i Bluetooth modul
- 40 pinova od kojih 17 GPIO pina (general-purpose input/output)
- HDMI priključak
- 4 USB 2.0 priključnice
- Priključak za kameru
- 3,5 mm audio priključak
- Micro SD čitač
- 5V/2.5A istosmjerno napajanje

2. Opis uređaja

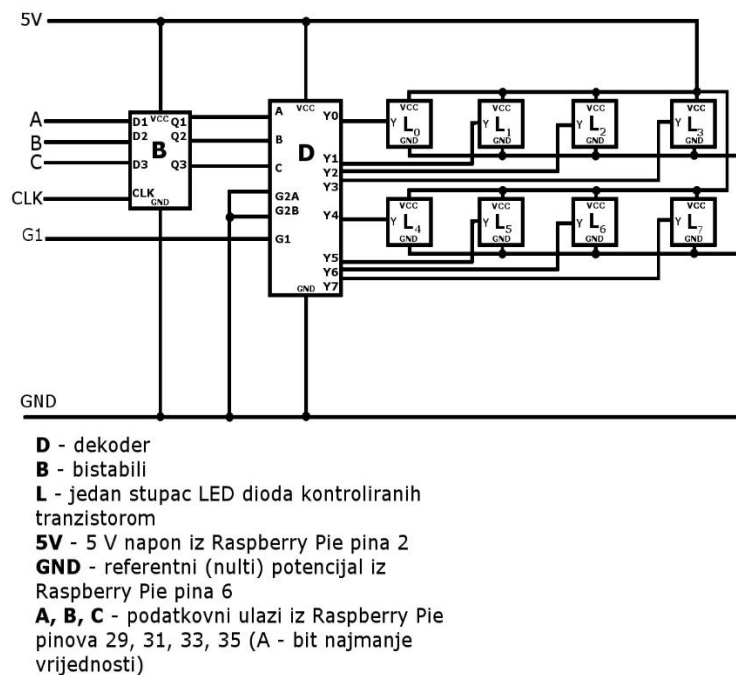
Uređaj sačinjavaju 3 funkcijska dijela (slika 2). Blok na slici s nazivom „RS“ odnosi se na Raspberry Pi. Kao što je na slici vidljivo s mikrokontrolera se dovodi napajanje na polje LED-ica (blok s nazivom „LED“) i na polje senzora (blok s nazivom „PS“). Upravljanje poljem LED-ica je realizirano preko pet GPIO pina. Tri su potrebna za adresiranje, jedan za takt i jedan za resetiranje polja. U prvom koraku se trebalo programski osigurati adresiranje stupaca u pravilnim vremenskim razmacima, kako bi LED-ice svijetlile kao „val“ čija brzina se može regulirat. U drugom koraku je trebalo realizirati reprodukciju zvuka u ovisnosti o položaju senzora. Zbog toga dovodimo sve redove polja senzora na GPIO pinove mikrokontrolera. Ulazi trebaju u određenom trenutku očitati da li je visoka ili niska razina i ovisno o tome reproducirati zvuk. Sklopovski je izvedeno da se dogodi pad napona od 2.8 volti kada je senzor uključen što označava logičku jedinicu.



Slika 2
Shema uređaja

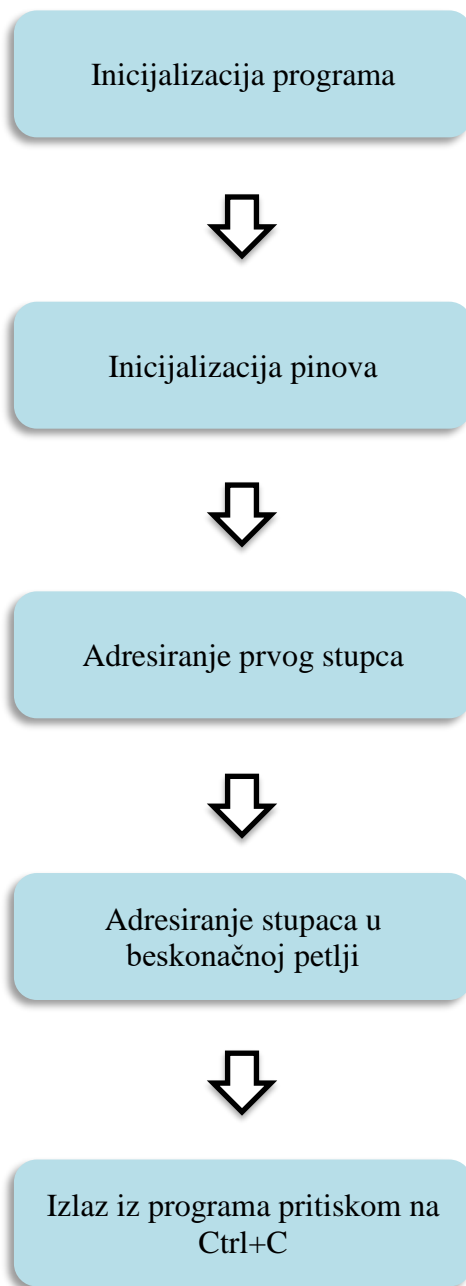
3. Upravljanje LED diodama

Prvi korak je bio napisati program koji upravlja s LED diodama, gdje se mikrokontroler koristi za napajanje pločice i za adresiranje stupaca. Tri GPIO pina se koriste kao izlaz te su spojeni na ulaze u bistabile koji su upravljani taktom. Izlazi bistabila su spojeni na dekodler (3 u 8) čiji izlazi su spojeni na stupce LED-ica. Program treba realizirati tako da se u beskonačnoj petlji, pomoću tri GPIO pina, šalje adresa stupca koja će se, prelaskom takta iz niske u visoku razinu, propagirati na ulaze u dekodler i time upaliti adresirani stupac LED-ica. Dekoder je invertiran što znači da ako na ulaz dovedemo „000“ izlaz s adresom „000“ će biti u niskoj razini dok će svi ostali biti u visokoj. Zbog toga izlaze dekodera dovodimo na invertore. Na početku programa inicijaliziramo program, zatim inicijaliziramo četiri pina kao izlazne, adresiramo prvi stupac prije ulaska u beskonačnu petlju te zatim ulazimo u beskonačnu petlju u kojoj adresiramo od broja „000“ do broja „111“.



Slika 3
Sklopovska izvedba

Dijagram toka:



3.1 Prvi program

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
```

Slika 4
Inicijalizacija programa

Na početku programa (slika 4) potrebno je inicijalizirati biblioteke u kojima su zapisane funkcije koje će se koristiti kasnije.

Prvim retkom ("RPi.GPIO as GPIO") sve pinove na pločici referenciram kao GPIO odnosno (general pin input output). Time se ne mora navoditi točno ime pina koje procesor „vidi“ nego vlastite oznake (odnosno oznake zapisane u toj biblioteci). Naredba "import time" služi da se može koristiti vrijeme kao varijabla (kasnije će biti objašnjena upotreba). Jako bitna naredba u ovom odlomku je „GPIO.setmode(GPIO.BOARD)“ u kojem se određuje kojem pinu će se dodijeliti koji broj. U ovome programu sam koristio najčešću podjelu GPIO.BOARD u kojem su pinovima brojevi dodijeljeni po redu. Tako je prvi pin onaj pin koji na drugoj strani pločice ima četvrtastu završetak, a pin u istom redu do njega se dodjeljuje broj 2 i tako redom.

Nadalje, u prvom odlomku (slika 5) se dodjeljuju imena određenim pinovima kako bih ih lakše kasnije koristio. CLK je normalan GPIO pin samo što sam mu ja ovdje opredijelio naziv CLK jer služi kao takt sklopu (kasnije će se objasniti kako je to izvedeno). Pin 35, kojemu sam opredijelio naziv B0, je najmanje značajan bit podatka koji se šalje na sklop muzičkog sekvencera. Tako je B1 značajniji od B0 dok je B2 najznačajniji. Ovdje još inicijaliziram varijablu „t“ koja označava vrijeme odnosno brzinu paljenja i gašenja led lampica.

```
CLK=29
B2=31
B1=33
B0=35
t=0.25
GPIO.setup(CLK,GPIO.OUT)
GPIO.setup(B2,GPIO.OUT)
GPIO.setup(B1,GPIO.OUT)
GPIO.setup(B0,GPIO.OUT)

GPIO.output(B0, False)
GPIO.output(B1, False)
GPIO.output(B2, False)
time.sleep(0.001)
GPIO.output(CLK, True)
time.sleep(0.001)
GPIO.output(CLK, False)
```

Slika 5
Inicijalizacija pinova

Naredba „GPIO.setup(...)“ određuje koji pin se koristi na koji način (kao ulazni ili kao izlazni). U ovom programu se koriste svi pinovi kao izlazni. U sljedećem odlomku referenciram se na te bitove i namještam ih na nisku naponsku razinu. Ovaj odlomak je stavljen tu kao sigurnost da program svaki puta kad se pokrene inicijalizira pinove na nisku naponsku razinu, da ne bi ostali u visokoj zbog prijašnjeg programa ili nečeg drugog. Bitna naredba je i „time.sleep(...)“ kojom određujemo koliko će Raspberry Pi pričekati (u sekundama) prije izvođenja sljedeće naredbe.

U ovom bloku naredba (slika 6) na početku koristim naredbu „try“ kako bi mogao osigurati kasnije ispravan prekid jer s naredbom „while 1“ ulazim u beskonačnu petlju. Prvi podatak koji šaljem na pločicu muzičkog sekvencera je „000“ jer je to adresa prvog stupca led lampica.

```
try:
    while 1:
        GPIO.output(B0, False) #nula
        GPIO.output(B1, False)
        GPIO.output(B2, False)
        time.sleep(0.01)
        GPIO.output(CLK, True)
        time.sleep(0.1)
        GPIO.output(CLK, False)
        time.sleep(t)
        GPIO.output(B0, True) #jedan
        GPIO.output(B1, False)
        GPIO.output(B2, False)
        time.sleep(0.01)
        GPIO.output(CLK, True)
        time.sleep(0.1)
        GPIO.output(CLK, False)
        time.sleep(t)
        GPIO.output(B0, False) #dva
```

Slika 6
Slanje signala na izlaz

Nakon toga se pričekava neko vrijeme kako bi se osiguralo postavljanje pinova u odgovarajuću naponsku razinu. Stavljam CLK u jedinicu kako bi se na pločici podaci zapisali u bistabile (okidani su bridom) te isto tako pričekam da budem siguran da se sve ispravno zapisalo. Nakon toga spuštam CLK na nisku naponsku razinu i puštam da led lampice svijetle „t“ sekundi. U ovom programu to je 0.25 sekundi. Nakon toga inicijaliziram postavljam pinove da daju na izlaz broj „001“ (to je drugi stupac led lampica), a ostalo je isto kao i u koraku prije. I tako napravim za svaki stupac odnosno sve do broja „111“.

Na kraju sam morao dodati sigurni izlaz iz beskonačne petlje. To je učinjeno naredbom na slici 7. Izlaz iz programa se postiže pritiskom u isto vrijeme na „Ctrl + C“. Time „očistimo“ pinove i oni postanu spremni za izvođenje sljedećeg programa.

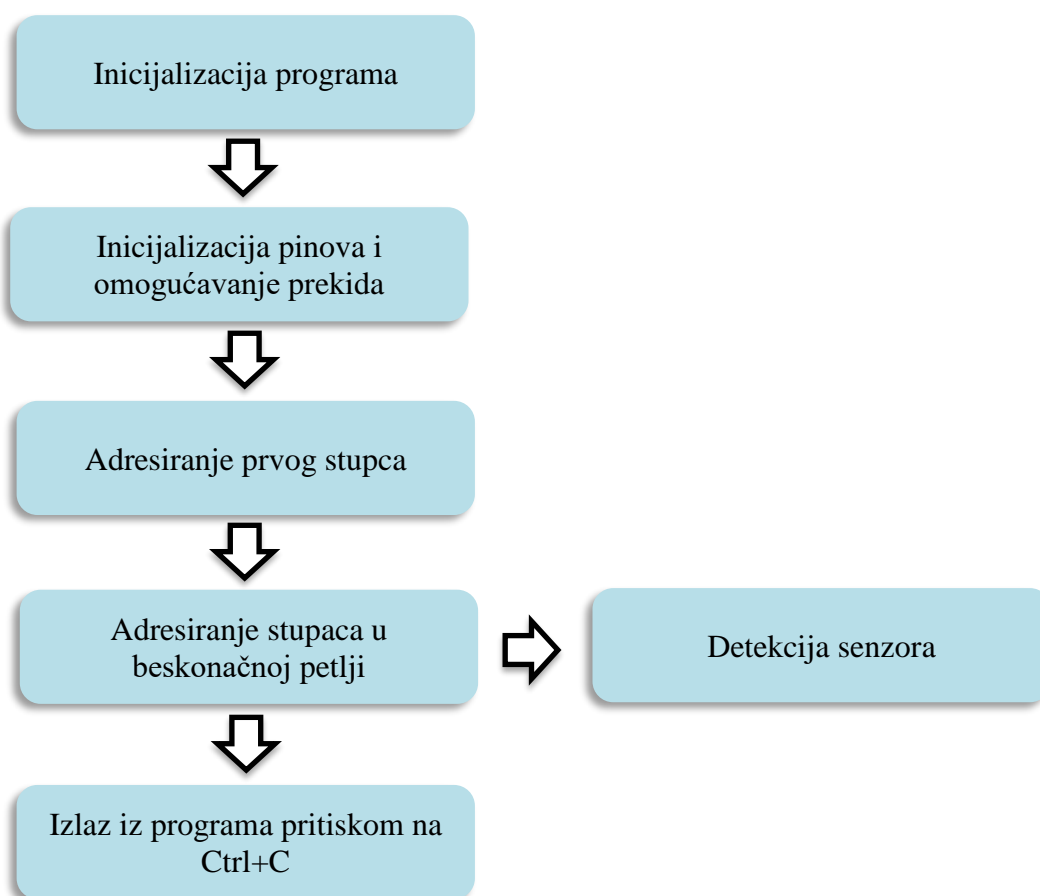
```
except KeyboardInterrupt:  
    GPIO.cleanup()
```

Slika 7
Izlaz iz programa

4 Programiranje senzora

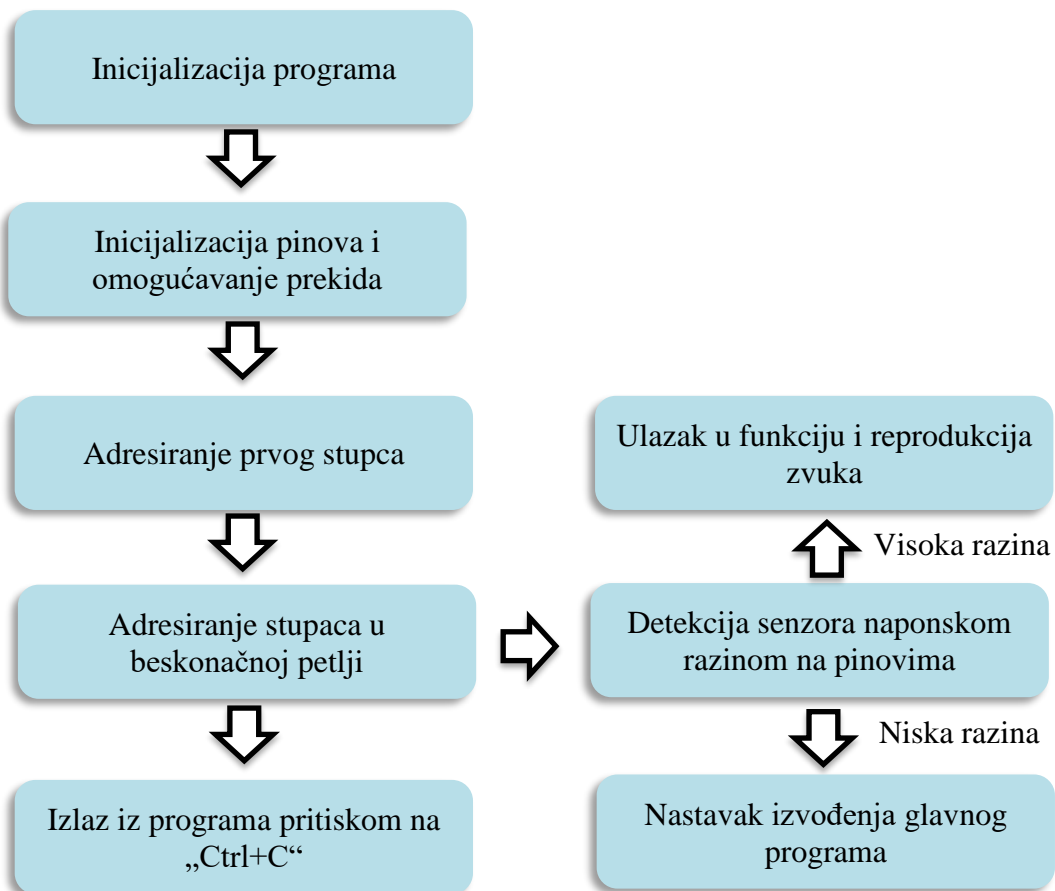
Drugi korak je bio napisati program koji će prepoznati na koji od redova je postavljen senzor i sukladno s time pustiti određeni zvuk kada se upali određeni stučac. Svaki red je spojen na jedan GPIO pin na mikrokontroleru. Naime, sklopovski je izvedeno tako da u trenutku paljenja stupca LED dioda, dolazi do pada napona na otporniku u iznosu od 2.8 volti što je dovoljno da GPIO pin prepozna kao jedinicu i pokrene određeni zvuk. Prvi problem koji tu nastaje je taj da se svih (u najgorem slučaju) osam zvukova moraju reproducirati u istom trenutku (ili dovoljno brzo jedan iza drugoga). Poželjno bi bilo da se glavni program oslobodi reprodukcije zvukova pošto u njemu upravljamo LED-icama, ali ne i nužno. U nastavku su programi koji predstavljaju put ka konačnom rješenju u detekciji senzora. U svakome od njih izostavio sam program iz prvog koraka zbog bolje preglednosti.

Dijagram toka programa:



4.1 Prvi program

U prvom programu detekcija senzora je realizirana u glavnom programu provjeravajući naponsku razinu na pinovima tijekom svijetljenja određenog stupca. Ako je naponska razina visoka poziva se funkcija u kojoj se reproducira zvuk, a ako je niska nastavlja se sa izvođenjem glavnog programa.



Čitajući literaturu naišao sam na biblioteku „gpiozero“ u kojoj postoje definirani gumbi. Pinovi koji se odrede da su gumbi, kao što sam napravio u retku šest i sedam gdje sam varijabli „jelly“ opredijelio pin 3, a varijabli „btn2“ pin 4, mogu se kasnije s opcijom „when_pressed“ ispitati da li su pritisnuti ili ne. U redcima dva i redcima od devet do trinaest, inicijaliziram zvukove koji će svirati ovisno o pinu, odnosno gumbu koji je „pritisnut“.

```
1  from gpiozero import Button
2  import pygame
3  from time import sleep
4  import RPi.GPIO as GPIO
5
6  jelly= Button(3)
7  btn2=Button(4)
8
9  pygame.init()
10 pygame.mixer.init()
11
12 burp = pygame.mixer.Sound("Front_Right.wav")
13 drugi = pygame.mixer.Sound("Noise.wav")
14
15 def hello():
16     burp.play
17     print ('Hello')
18
19 def hello2():
20     drugi.play
21     print ('drugi')
22
23
24
25 try:
26     while True:
27         jelly.when_pressed=hello
28         btn2.when_pressed=hello2
29
30 except KeyboardInterrupt:
31     GPIO.cleanup()
```

Slika 8
Prvi program

Pritiskom na određeni gumb poziva se funkcija (poziva se naredbom „ime_funkcije“ u redu 27 i 28) u kojoj se pušta zvuk određen tim gumbom (zvukovi se dodjeljuju varijablama u redcima 12 i 13). Ovdje pritisnut gumb ne znači ništa više nego da na ulazni pin dođe visoka naponska razina (u našem slučaju 2.8 volti) kao što je to sklopovski dizajnirano. Problem koji nastaje tijekom izvođenja ovog programa je taj da glavni program čeka na pritisak prvog gumba kako bi mogao preći na čekanje pritiska drugog. Time nije omogućeno preskakanje gumbova, nego se moraju pritiskati redoslijedom kako je napisano u glavnom programu.

4.2 Drugi program

U ovom programu se detekcija senzora realizira u glavnom programu gdje se odmah i reproducira zvuk. Dijagram toka ovog programa je jako sličan prvom programu. Jedina razlika je što se ovdje ne koriste inicijalizacija pinova kao gumba nego se njihova razina ispituje u glavnom programu pomoću *if* slučaja u kojima se reproduciraju zvukovi.

Ovdje prvi put uvodim (što ću kasnije imati u svakom programu) *pull up* i *pull down* otpornike. Naime, ti otpornici se programski mogu namjestiti kao što je učinjeno u retku osam i devet. Naredbom „GPIO.PUD_DOWN“ se postavlja *pull down*, a naredbom „GPIO.PUD_UP“ se postavlja *pull up* otpornik. Ti otpornici su jako korisni jer služe tome da pin ne „visi u zraku“ nego da je pričvršćen na neki potencijal.

```
1  from gpiozero import Button
2  import pygame
3  from time import sleep
4  import RPi.GPIO as GPIO
5
6  jelly=5
7  belly=7
8  GPIO.setup(jelly,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
9  GPIO.setup(belly,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
10
11  pygame.init()
12  pygame.mixer.init()
13
14  burp = pygame.mixer.Sound("Front_Right.wav")
15  drugi = pygame.mixer.Sound("Noise.wav")
16
17  while True:
18      if GPIO.input (jelly)==1:
19          burp.play()
20          sleep(2)
21      if GPIO.input (belly)==1:
22          drugi.play()
23          sleep(2)
```

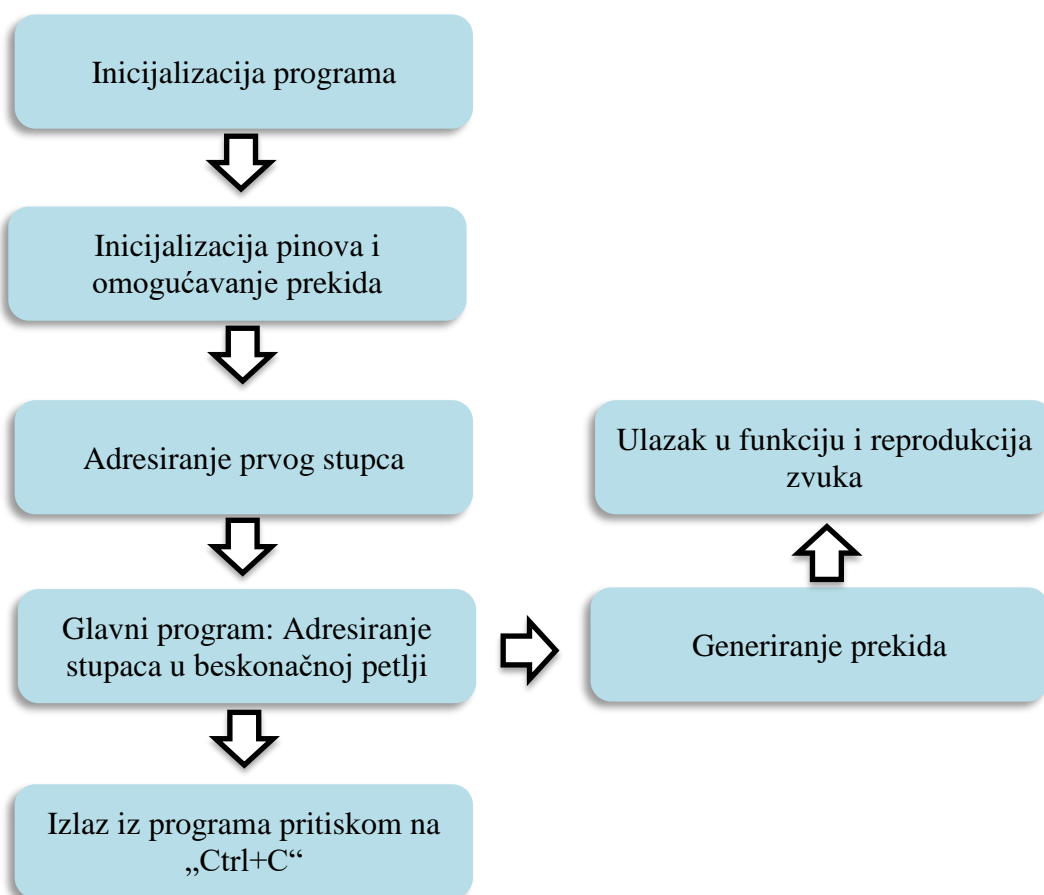
Slika 9
Drugi program

Time uvelike smanjujemo utjecaj smetnji. Ako pin „zaključamo“ otpornikom *pull down*, kao što je u ovom programu, onda pin očitava vrijednost jedan kada je visoka naponska razina dok „zaključavanjem“ sa *pull up* otpornikom vrijedi obrnuta logika (očitava jedinicu kada je naponska razina niska). Prvih četiri retka su već standardna inicijalizacija pinova, dok su 11. i 12. redak inicijalizacija za sviranje zvuka, a 14. i 15.

redak su dodjela određenog zvuka varijablama. U glavnom programu je beskonačna petlja s *if* slučajevima. Svaki slučaj će biti zadovoljen ako je na određenom pinu visoka naponska razina i tako će se pustiti određeni ton. Ovaj program ne rješava problem istitravanja. Naime, ako se ploča pomakne zatim prekine pa ponovno uspostavi kontakt dok se pušta zvuk određenog stupca, isti zvuk bi se pustio dva ili više puta. Pokušao sam sa naredbom „sleep“ onemogućiti ulazak u isti *if* slučaj, ali ta naredba zaustavlja izvođenje cijelog programa na određeni broj sekundi.

4.3 Treći program

Ovim programom sam detekciju senzora realizirao detekcijom promjene bridova na pinovima iz niske u visoku razinu, što uzrokuje generiranje prekida zbog kojeg se onda poziva funkcija i ispušta se određeni zvuk. Ovime sam oslobodio glavni program od stalne provjere naponskih razina na ulazu pinova.



U programu (slika 10) sam detektirao rastući brid (redak 25 i 26). Dokle god red nije aktivan (red koji nema postavljen senzor u pojedinom stupcu koji je trenutno aktivan) na ulazu u pripadajući pin je niska naponska razina. Kada se aktivira sljedeći stupac i ako u tome istome redu postoji senzor, naponska razina prelazi u visoku i time pusti određeni zvuk. Isto tako, na ovaj način se oslobađa glavni program od provjere naponskih razina na ulazu pinova.

```

2  import pygame
3  from time import sleep
4  import RPi.GPIO as GPIO
5
6  jelly=5
7  belly=7
8  GPIO.setup(jelly,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
9  GPIO.setup(belly,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
10
11 pygame.init()
12 pygame.mixer.init()
13
14 burp = pygame.mixer.Sound("Front_Right.wav")
15 drugi = pygame.mixer.Sound("Noise.wav")
16
17 def Input1(jelly):
18     burp.play
19     print ('Front')
20
21 def Input2(belly):
22     drugi.play
23     print ('sum')
24
25 GPIO.add_event_detect(jelly, GPIO.RISING, callback=Input1)
26 GPIO.add_event_detect(belly, GPIO.RISING, callback=Input2)
27
28 try:
29     while True:
30         if GPIO.input(jelly):
31             print('Input was HIGH')
32             sleep(2)
33         else:
34             print('Input was LOW')
35             sleep(2)
36
37 except KeyboardInterrupt:
38     GPIO.cleanup()

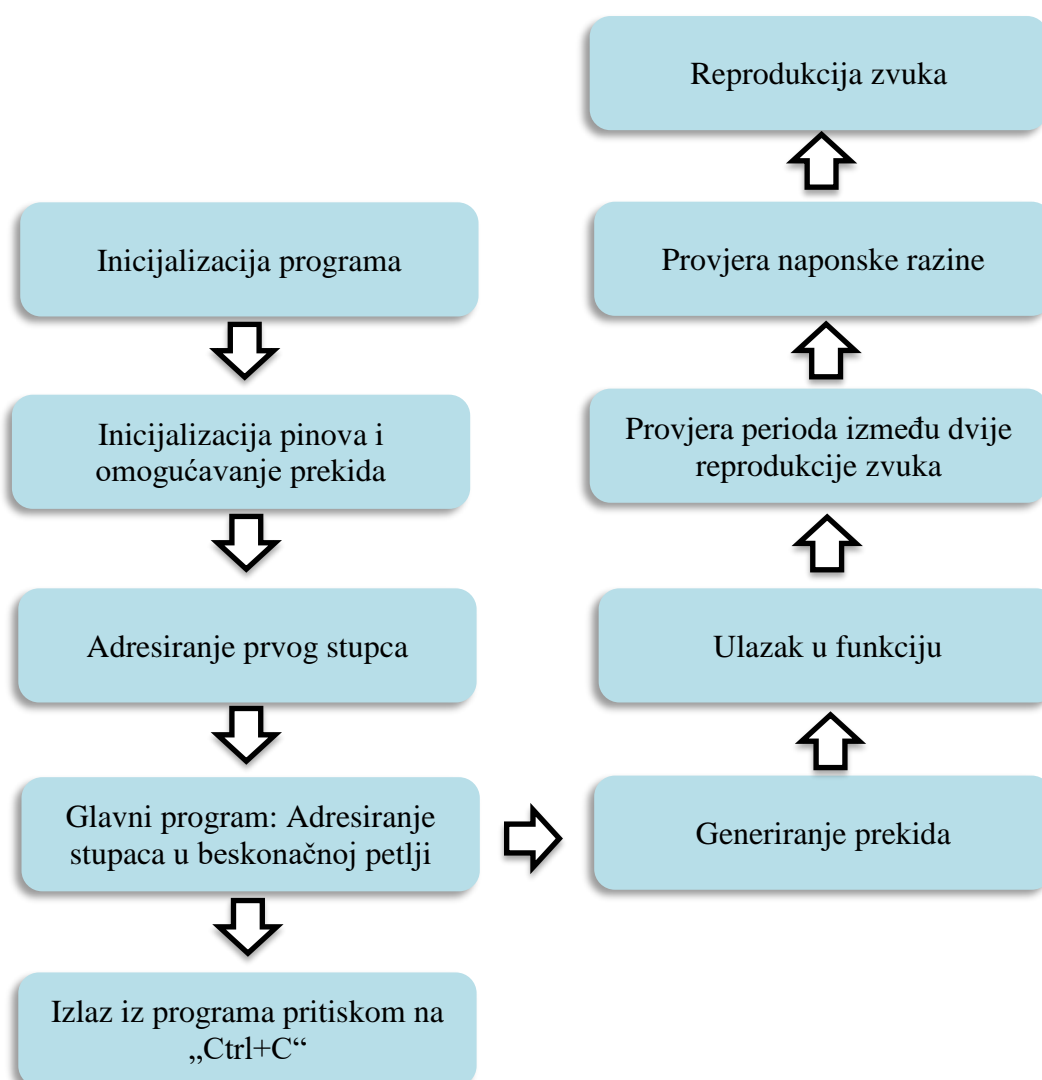
```

Slika 10
Treći program

Na taj način se može izvoditi program koji upravlja paljenjem stupaca LED dioda. Nedostatak ovog programa se opet javlja u istitravanju, odnosno ulaska u istu funkciju više puta za vrijeme aktivnosti jednog stupca. Isto tako, drugi nedostatak je loša detekcija rastućeg brida. Zbog činjenice da su pinovi podložni smetnjama gradske mreže, jako male oscilacije na ulazu okidaju detekciju promjene i time dolazi do puštanja određenog zvuka iako na ulazu se ništa nije promijenilo. Iz tog razloga u glavnom programu se provjerava da li je naponska razina visoka ili niska kada se detektirala promjena. Glavni program služi kao provjera detekcije promjene naponskih razina.

4.4 Četvrti program

Četvrti program (slika 11) je kombinacija drugog i četvrtog programa. Funkcije se pozivaju promjenom naponske razine, dok se u njima provjerava je li promjena uzrokovana smetnjom ili ne. Isto tako istitravanje je realizirano unutar same funkcije. Ovime je prekid izazvan rastućim i padajućim bridom te se ulazi u funkciju svaki put, ali se zvuk ne reproducira ako nije visoka razina i ako nije prošlo dovoljno vremena od zadnje reprodukcije istog.



Ovdje je, jer se pokazalo da dolazi do neželjenih detekcija promjena naponskih razina, stavljeno da se funkcije pozivaju na padajući i rastući brid. To se postiglo s naredbom „GPIO.BOTH“.

```

1  import RPi.GPIO as GPIO
2  import time
3  GPIO.setmode(GPIO.BOARD)
4  import pygame
5
6  pygame.init()
7  pygame.mixer.init()
8
9  burp = pygame.mixer.Sound("Front_Right.wav")
10 noise = pygame.mixer.Sound("Noise.wav")
11
12 GPIO.setup(37, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
13 GPIO.setup(35, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
14 time_stamp1=time.time()
15 time_stamp2=time.time()
16 def prvi(channel):
17     global time_stamp1
18     time_now=time.time()
19     if(time_now-time_stamp1)>=1:
20         print ("pritisnuo si 35")
21         if GPIO.input(35):
22             print ("Rising")
23             burp.play()
24         else:
25             print("Falling")
26     time_stamp1=time.time()
27
28
29 def drugi(channel):
30     global time_stamp2
31     time_now=time.time()
32     if(time_now-time_stamp2)>=1:
33         print ("pritisnuo si 37")
34         if GPIO.input(37):
35             print ("Rising")
36             noise.play()
37         else:
38             print("Falling")
39     time_stamp2=time.time()
40
41 GPIO.add_event_detect(35, GPIO.BOTH, callback=prvi)
42 GPIO.add_event_detect(37, GPIO.BOTH, callback=drugi)
43
44 try:
45     while(1):
46         time.sleep(100)
47 except KeyboardInterrupt:
48     GPIO.cleanup()
49

```

Slika 11
Četvrti program

Kada se pozove određena funkcija u njoj se ispituje je li došlo do željene detekcije. To se provjerava u 21. retku prve funkcije i 34. retku druge funkcije s jednim *if* slučajem. Naime, ako je pin koji je zabilježio promjenu i dalje u visokoj razini, znači da se treba

pustiti određeni zvuk (redovi 23 i 36). Zbog nepouzdanosti naredbe „bouncetime“ iz prošlog programa, ovdje se provjerava koliko je vremena prošlo od prošlog poziva funkcije. U redu 14 i 15 sam definirao varijable „time_stamp1“ i „time_stamp2“. U te varijable sam spremio trenutno vrijeme na početku programa. Ulaskom u funkciju (kao primjer proći ćemo kroz funkciju „prvi“) prvo definiramo (redak 17) da nam je varijabla „time_stamp1“ globalna. U drugu varijablu „time_now1“ pohranjujemo trenutno vrijeme te zatim u *if* slučaju oduzimamo varijablu „time_now1“ s varijablom „time_stamp1“. Ako je razlika veća od 1, što znači da je prošla jedna sekunda od prošlog poziva funkcije, ispisat će se „pritisnuo si 35“. Poslije toga se ispituje naponska razina odgovarajućeg pina te ako je ta ista razina visoka ispisuje se „Rising“ i pušta zvuk dok u drugom slučaju se samo ispisuje „Falling“. Na kraju se u varijablu „time_stamp1“ ponovno upisuje trenutno vrijeme u sekundama kako bi se prilikom ponovnog ulaska u funkciju usporedila s trenutnim vremenom.

4.5 Završni program

Završni program (slika 12) je kombinacija programa koji upravlja stupcima LED dioda iz prvog koraka i četvrtog programa koji upravlja prekidima odnosno senzorima. U glavnom programu se još poziva funkcija „reset“ koja postavlja naponske razine na ulazima pinova senzora na nisku razinu nakon svakog stupca. Time je omogućeno sviranje istog tona dva puta za redom.



U nastavku ću objasniti završni program na samo par senzora i stupaca LED dioda. Cijeli program funkcijski izgleda isto samo su još naredbe proširene za preostale stupce i senzore. Cijeli završni program priložit ću na kraju završnog rada.

Prva promjena se može uočiti u retku 6 u kojem se inicijaliziraju, po redu, karakteristike puštanja zvuka: frekvencija reprodukcije zvukova, broj korištenih bitova, broj kanala (kanal 1 je za puštanje preko HDMI kabla dok kanal 2 je za puštanje preko audio priključka), veličinu međuspremnika (*buffer*).

```
1 import RPi.GPIO as GPIO
2 import time
3 GPIO.setmode(GPIO.BOARD)
4 import pygame
5
6 pygame.mixer.pre_init(44100, -16, 2, 512)
7 pygame.mixer.init()
8 pygame.init()
9
10 CLK=29
11 B2=31
12 B1=33
13 B0=35
14 E=12
15 t=0.35
16 z=t
17
18 GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
19 GPIO.setup(13, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
20
21 jedan = pygame.mixer.Sound("Zvuk1.wav")
22 dva = pygame.mixer.Sound("Zvuk2.wav")
23
24 GPIO.setup(CLK, GPIO.OUT)
25 GPIO.setup(B2, GPIO.OUT)
26 GPIO.setup(B1, GPIO.OUT)
27 GPIO.setup(B0, GPIO.OUT)
28 GPIO.setup(E, GPIO.OUT)
29
30 time_stamp1=time.time()
31 time_stamp2=time.time()
```

Slika 12
Završni program

Poteškoća koja se pojavila je bila ta da se zvuk puštao prekasno, odnosno pustio bi se tek kada bi zasvijetlio stupac poslije stupca u kojem su se nalazili senzori. To kašnjenje se uvelike smanjilo sa smanjenjem međuspremnika. Veličina međuspremnika mora biti potencija broja 2, ali ako se stavi na mali broj, npr. 64, dolazi do velikog izobličenja zvuka. Tako je brojem 512 postignuta željena kvaliteta i brzina puštanja zvuka. Dodana je još jedna varijabla, varijabla „z“, čija je vrijednost jednaka varijabli „t“. Varijabla „t“ određuje (kasnije u programu) koliko će se brzo propagirati svjetlosni „val“ po stupcima, dok varijabla „z“ određuje vrijeme koje treba proći prije ispuštanja istog

zvuka. U redcima 21 i 22 su varijablama „jedan“ i „dva“ pridijeljeni tonovi „Zvuk1.wav“ i „Zvuk2.wav“. Prvo učitavanje tonova je bilo neuspješno jer tonovi nisu bili 16 bitni kako je definirano u šestom retku. Problem je riješen formatiranjem svih tonova. Uz već prije definirane izlazne pinove (35, 33, 31) kojima se adresira određeni stupac i pin 29 koji služi kao takt, definira se pin 12, isto kao izlazni, koji služi za resetiranje svih stupaca (niti jedan stupac ne svijetli). Kasnije ću objasniti koji je razlog tome.

U ovom isječku programa (slika 14) definirane su funkcije u koje se ulazi prilikom prekida izazvanog promjenom stanja na sensorima. Ovdje, za razliku od programa sa stranice 10, koristimo varijablu „z“ kojom se namješta vrijeme između puštanja istog zvuka. To vrijeme je jednako brzini promjene stupaca jer u slučaju da je manje, prilikom pomaka senzora, koje iskapčaju pa ukapčaju strujni krug i time generiraju prekide, dolazi do puštanja istog zvuka više puta za vrijeme svijetljenja jednog stupca. Ovime se onemogućuje puštanje zvuka dokle god je stupac aktivan.

```
33 def prvi(channel):
34     global time_stamp1
35     time_now=time.time()
36     if(time_now-time_stamp1)>=z:
37         if GPIO.input(11):
38             print ("Prvi red")
39             jedan.play()
40             time_stamp1=time.time()
41         else:
42             print ("prvi red")
43
44 def drugi(channel):
45     global time_stamp2
46     time_now=time.time()
47     if(time_now-time_stamp2)>=z:
48         if GPIO.input(13):
49             print ("Drugi red")
50             dva.play()
51             time_stamp2=time.time()
52         else:
53             print ("drugi red")
```

Slika 13
Završni program

Tek kada sljedeći stupac postane aktivan može doći do ponovnog ispuštanja zvuka. Također, ako stavimo da je varijabla „z“ veća od varijable „t“ onemogućuje se puštanje istog tona dva puta za redom. Iz tih razloga su te dvije varijable jednake. U prijašnjem programu (slika 12) u varijablu „time_stampX“ (X označava red) se sprema trenutno

vrijeme prilikom svakog prolaska kroz funkciju. Time je bilo onemogućeno puštanje zvukova za vrijeme jednako varijabli „z“ iako je prijašnji ulazak u funkciju uzrokovala smetnja. To smo riješili tako da smo varijablu „time_stampX“ ubacili u *if* slučaj koja sad sprema vrijeme samo kada se pusti ton. Sada se, prilikom prekida i ulaskom u funkciju, ton pušta samo ako je prošlo vrijeme „z“ od zadnjeg puštenog tona. Pojedina ispisivanja na ekranu su služila isključivo za provjeru.

Funkcijom „reset“ (slika 15) postavljamo sve stupce u nisku razinu. Prilikom testiranja ustanovljeno je da se ne reproduciraju zvukovi dva puta za redom iako su varijable „z“ i „t“ bile namještene tako da je „z“ manji ili jednak od „t“.

```
56 def reset():
57     GPIO.output(E, False)
58     GPIO.output(CLK, True)
59     time.sleep(0.001)
60     GPIO.output(CLK, False)
61     GPIO.output(E, True)
```

Slika 14
Završni program

Naime, kako je isti red u svim stupcima povezan s istom žicom čiji potencijal se onda dijeli na dva otpornika i to dijelilo se dovodi na ulaz u mikrokontroler, prilikom postavljanja uzastopnih senzora u istom redu, ne dolazi do promjene naponske razine na pinu. Zbog toga ne dolazi do prekida pa tako ni do reprodukcije zvukova. Iz tog razloga se pin 12 spaja na bistabil čiji izlaz je spojen na dekodler. Kada je izlaz iz bistabila na visokoj razini sklop radi normalno, kada se on dovede u nisku



Slika 15
Signal na ulazu u pin s dva senzora u istom redu susjednih stupaca

razinu, dekodir sve svoje izlaze postavi u visoku razinu koji onda preko invertora upravljaju stupcima i senzorima. Dakle, kada se na pin 12 dovede niska razina i takt prijeđe iz niske u visoku razinu, svi stupci prestaju svijetli, čime se na ulaz pinova senzora dovodi niska razina. Time je dodan (slika 17), u trenutku prijelaza s jednog stupca na drugi, prijelaz napona u nisku razinu kako bi se u sljedećem stupcu dogodio prijelaz iz niske u visoku razinu i time se generirao prekid. Funkcija „reset“ kratko traje kako ljudsko oko ne bi zamijetilo da niti jedan stupac ne svijetli.



Slika 16
Signal na ulazu u pin s dva senzora u istom redu susjednih stupaca
s funkcijom „reset“

Prije glavnog programa se definiraju prekidi i funkcije koje se pozivaju, postavljaju se pinovi, koji su spojeni na bistabile, na nisku razinu i pin za „reset“ u visoku i to se popraća s promjenom takta iz niske u visoku pa opet u nisku razinu kako bi se ti izlazi pinova pojavili na izlazima bistabila. U glavnom programu postoji beskonačna petlja koja upravlja stupcima kako je opisano na stranici 5.

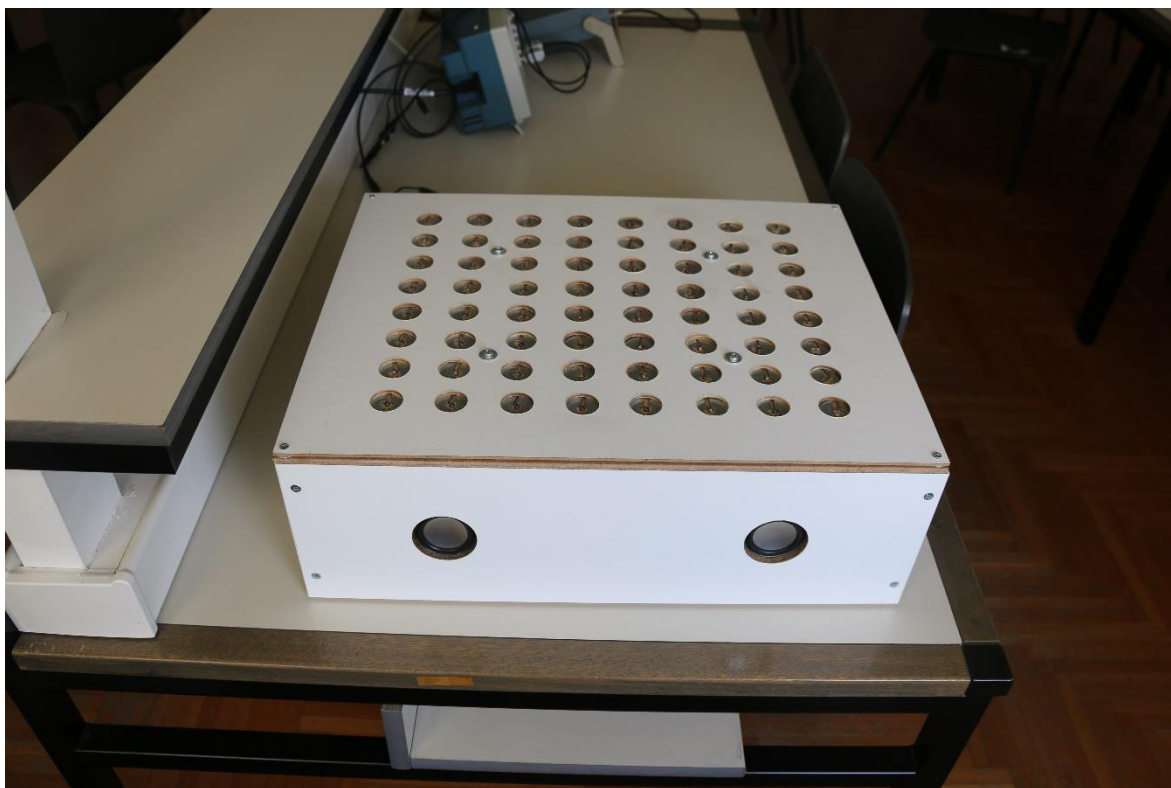
```
63 GPIO.add_event_detect(11, GPIO.BOTH, callback=prvi)
64 GPIO.add_event_detect(13, GPIO.BOTH, callback=drugi)
65
66 GPIO.output(B0, False)
67 GPIO.output(B1, False)
68 GPIO.output(B2, False)
69 GPIO.output(E, True)
70 GPIO.output(CLK, False)
71 time.sleep(0.001)
72 GPIO.output(CLK, True)
73 time.sleep(0.001)
74 GPIO.output(CLK, False)
75
76 try:
77     while(1):
78         GPIO.output(B0, False) #nula
79         GPIO.output(B1, False)
80         GPIO.output(B2, False)
81         time.sleep(0.001)
82         GPIO.output(CLK, True)
83         time.sleep(0.001)
84         GPIO.output(CLK, False)
85         time.sleep(t)
86         reset()
87
88         GPIO.output(B0, True) #jedan
89         GPIO.output(B1, False)
90         GPIO.output(B2, False)
91         time.sleep(0.001)
92         GPIO.output(CLK, True)
93         time.sleep(0.001)
94         GPIO.output(CLK, False)
95         time.sleep(t)
96         reset()
97
98         GPIO.output(B0, False) #dva
99         GPIO.output(B1, True)
100        GPIO.output(B2, False)
101        time.sleep(0.001)
102        GPIO.output(CLK, True)
103        time.sleep(0.001)
104        GPIO.output(CLK, False)
105        time.sleep(t)
106        reset()
107
108 except KeyboardInterrupt:
109     GPIO.cleanup()
```

Slika 17
Završni program

Promijenjeno je vrijeme između postavljanja pinova na visoku odnosno nisku razinu i dodan je poziv funkcije „reset“ poslije svakog binarnog broja, odnosno stupca. Izlaz programa je opet omogućen pritiskom na „Ctrl+C“ čime se „čiste“ pinovi i na siguran način izlazi iz programa.

Zaključak

U ovom radu sam opisao realizaciju programske podrške za muzički sekvencer. Kao mikrokontroler izabran je Raspberry Pi 3 model B. Konačni program izgleda poprilično dugačak. Razlog tome je veliki broj LED-ica, ali dovoljno je shvatiti princip rada jednog stupca i jednog senzora jer se ista logika primjenjuje na sve ostale. Program je prilagođen sklopovskoj izvedbi muzičkog sekvencera zbog čega preporučujem prvo proučiti sklopovski dio dokumentacije. Omogućio sam izvođenje programa na „boot up-u“ zbog čega nema više potrebe za monitorom, već će se program početi izvoditi čim se spoji napajanje mikrokontrolera. Pripomogao sam i sklopovskoj izvedbi uređaja kako idejno tako i u postavljanju. Preporučljivo je što manje baratati sa uređajem zbog mogućnosti odlemljivanja.



Slika 18
Muzički sekvencer

LITERATURA

https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM_1p0.pdf
<https://projects.raspberrypi.org/en/projects/burping-jelly-baby/7>
[https://projects.raspberrypi.org/en/projects?technologies\[\]=python](https://projects.raspberrypi.org/en/projects?technologies[]=python)
<https://projects.raspberrypi.org/en/projects/gpio-music-box>
<https://www.raspberrypi.org/documentation/usage/audio/README.md>
<https://github.com/raspberrypilearning/burping-jelly-baby/blob/master/worksheet.md>
<https://gpiozero.readthedocs.io/en/stable/recipes.html>
https://gpiozero.readthedocs.io/en/stable/recipes_advanced.html
<https://www.raspberrypi.org/forums/viewtopic.php?t=148428>
<http://marjan.fesb.hr/~imarin/prsklprim.htm>
<https://www.raspberrypi.org/forums/viewtopic.php?t=7934>
<https://www.raspberrypi.org/forums/viewtopic.php?t=43216>

Programska podrška za muzički sekvencer

Sažetak

Program namijenjen za izvođenje na Raspberry Pi-u koji upravlja stupcima LED dioda i senzorima te zbog određenog položaja senzora na ploči pušta određeni zvuk, pisan u programskom jeziku Python.

Ključne riječi: Program, Raspberry Pi, LED diode, muzički sekvencer, Python.

Software for music sequencer

Abstract

Program for Raspberry Pi microcontroller which controls LED diodes and sensors that plays sounds depending on sensor placement, written in language Python.

Keywords: Program, Raspberry Pi, LED diodes, music sequencer, software, Python.

Cijeli program:

```
1  import RPi.GPIO as GPIO
2  import time
3  GPIO.setmode(GPIO.BOARD)
4  import pygame
5
6  pygame.mixer.pre_init(44100, -16, 2, 512)
7  pygame.mixer.init()
8  pygame.init()
9
10 CLK=29
11 B2=31
12 B1=33
13 B0=35
14 E=12
15 t=0.35
16 z=t
17
18 GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
19 GPIO.setup(13, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
20 GPIO.setup(15, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
21 GPIO.setup(16, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
22 GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
23 GPIO.setup(36, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
24 GPIO.setup(38, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
25 GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
26
27 jedan = pygame.mixer.Sound("Zvuk1.wav")
28 dva = pygame.mixer.Sound("Zvuk2.wav")
29 tri = pygame.mixer.Sound("Zvuk3.wav")
30 cetiri = pygame.mixer.Sound("Zvuk4.wav")
31 pet = pygame.mixer.Sound("Zvuk5.wav")
32 sest = pygame.mixer.Sound("Zvuk6.wav")
33 sedam = pygame.mixer.Sound("Zvuk7.wav")
34 osam = pygame.mixer.Sound("Zvuk8.wav")
35
36 GPIO.setup(CLK, GPIO.OUT)
37 GPIO.setup(B2, GPIO.OUT)
38 GPIO.setup(B1, GPIO.OUT)
39 GPIO.setup(B0, GPIO.OUT)
40 GPIO.setup(E, GPIO.OUT)
41
42 time_stamp1=time.time()
43 time_stamp2=time.time()
44 time_stamp3=time.time()
45 time_stamp4=time.time()
46 time_stamp5=time.time()
47 time_stamp6=time.time()
48 time_stamp7=time.time()
49 time_stamp8=time.time()
```

Slika 19
Cijeli program


```

50
51 def prvi(channel):
52     global time_stamp1
53     time_now=time.time()
54     if(time_now-time_stamp1)>=z:
55         if GPIO.input(11):
56             print ("Prvi red")
57             jedan.play()
58             time_stamp1=time.time()
59         else:
60             print ("prvi red")
61
62 def drugi(channel):
63     global time_stamp2
64     time_now=time.time()
65     if(time_now-time_stamp2)>=z:
66         if GPIO.input(13):
67             print ("Drugi red")
68             dva.play()
69             time_stamp2=time.time()
70         else:
71             print ("drugi red")
72
73 def treci(channel):
74     global time_stamp3
75     time_now=time.time()
76     if(time_now-time_stamp3)>=z:
77         if GPIO.input(15):
78             print ("Treci red")
79             tri.play()
80             time_stamp3=time.time()
81         else:
82             print("treci red")
83
84 def cetvrti(channel):
85     global time_stamp4
86     time_now4=time.time()
87     if(time_now4-time_stamp4)>=z:
88         if GPIO.input(16):
89             print ("Cetvrti red")
90             cetiri.play()
91             time_stamp4=time.time()
92         else:
93             print("cetvrti red")
94

```

Slika 20
Cijeli program

```

95 def peti(channel):
96     global time_stamp5
97     time_now=time.time()
98     if(time_now-time_stamp5)>=z:
99         if GPIO.input(18):
100             print ("Peti red")
101             pet.play()
102             time_stamp5=time.time()
103         else:
104             print("peti red")
105
106 def sestti(channel):
107     global time_stamp6
108     time_now=time.time()
109     if(time_now-time_stamp6)>=z:
110         if GPIO.input(36):
111             print ("Sesti red")
112             sest.play()
113             time_stamp6=time.time()
114         else:
115             print("sesti red")
116
117 def sedmi(channel):
118     global time_stamp7
119     time_now=time.time()
120     if(time_now-time_stamp7)>=z:
121         if GPIO.input(38):
122             print ("Sedmi red")
123             sedam.play()
124             time_stamp7=time.time()
125         else:
126             print("sedmi red")
127
128 def osmi(channel):
129     global time_stamp8
130     time_now=time.time()
131     if(time_now-time_stamp8)>=z:
132         if GPIO.input(40):
133             print ("Osmi red")
134             osam.play()
135             time_stamp8=time.time()
136         else:
137             print("osmi red")
138
139 def reset():
140     GPIO.output(E, False)
141     GPIO.output(CLK, True)
142     time.sleep(0.001)
143     GPIO.output(CLK, False)
144     GPIO.output(E, True)

```

Slika 21
Cijeli program

```

145
146 GPIO.add_event_detect(11, GPIO.BOTH, callback=prvi)
147 GPIO.add_event_detect(13, GPIO.BOTH, callback=drugi)
148 GPIO.add_event_detect(15, GPIO.BOTH, callback=treci)
149 GPIO.add_event_detect(16, GPIO.BOTH, callback=cetvrti)
150 GPIO.add_event_detect(18, GPIO.BOTH, callback=peti)
151 GPIO.add_event_detect(36, GPIO.BOTH, callback=sesti)
152 GPIO.add_event_detect(38, GPIO.BOTH, callback=sedmi)
153 GPIO.add_event_detect(40, GPIO.BOTH, callback=osmi)
154
155 GPIO.output(B0, False)
156 GPIO.output(B1, False)
157 GPIO.output(B2, False)
158 GPIO.output(E, True)
159 time.sleep(0.001)
160 GPIO.output(CLK, True)
161 time.sleep(0.001)
162 GPIO.output(CLK, False)
163
164 try:
165     while(1):
166         GPIO.output(B0, False) #nula
167         GPIO.output(B1, False)
168         GPIO.output(B2, False)
169         time.sleep(0.001)
170         GPIO.output(CLK, True)
171         time.sleep(0.001)
172         GPIO.output(CLK, False)
173         time.sleep(t)
174         reset()
175
176         GPIO.output(B0, True) #jedan
177         GPIO.output(B1, False)
178         GPIO.output(B2, False)
179         time.sleep(0.001)
180         GPIO.output(CLK, True)
181         time.sleep(0.001)
182         GPIO.output(CLK, False)
183         time.sleep(t)
184         reset()
185
186         GPIO.output(B0, False) #dva
187         GPIO.output(B1, True)
188         GPIO.output(B2, False)
189         time.sleep(0.001)
190         GPIO.output(CLK, True)
191         time.sleep(0.001)
192         GPIO.output(CLK, False)
193         time.sleep(t)
194         reset()

```

Slika 22
Cijeli program

```

195
196     GPIO.output(B0, True) #tri
197     GPIO.output(B1, True)
198     GPIO.output(B2, False)
199     time.sleep(0.001)
200     GPIO.output(CLK, True)
201     time.sleep(0.001)
202     GPIO.output(CLK, False)
203     time.sleep(t)
204     reset()
205     GPIO.output(B0, False) #cetiri
206     GPIO.output(B1, False)
207     GPIO.output(B2, True)
208     time.sleep(0.001)
209     GPIO.output(CLK, True)
210     time.sleep(0.001)
211     GPIO.output(CLK, False)
212     time.sleep(t)
213     reset()
214
215     GPIO.output(B0, True) #pet
216     GPIO.output(B1, False)
217     GPIO.output(B2, True)
218     time.sleep(0.001)
219     GPIO.output(CLK, True)
220     time.sleep(0.001)
221     GPIO.output(CLK, False)
222     time.sleep(t)
223     reset()
224     GPIO.output(B0, False) #sest
225     GPIO.output(B1, True)
226     GPIO.output(B2, True)
227     time.sleep(0.001)
228     GPIO.output(CLK, True)
229     time.sleep(0.001)
230     GPIO.output(CLK, False)
231     time.sleep(t)
232     reset()
233
234     GPIO.output(B0, True) #sedam
235     GPIO.output(B1, True)
236     GPIO.output(B2, True)
237     time.sleep(0.001)
238     GPIO.output(CLK, True)
239     time.sleep(0.001)
240     GPIO.output(CLK, False)
241     time.sleep(t)
242     reset()
243
244     except KeyboardInterrupt:
245         GPIO.cleanup()
246

```

Slika 23
Cijeli program