Mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom

Lovretić, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MIKROUPRAVLJAČKI SUSTAV S JEDNIM TIPKALOM ZA NAVIGACIJU IZBORNIKOM

Završni rad

Luka Lovretić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 03.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Luka Lovretić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4234, 26.07.2018.
OIB studenta:	34962895614
Mentor:	Izv. prof. dr. sc. Davor Vinko
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom
Znanstvena grana rada:	**Elektronika (zn. polje elektrotehnika)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	03.09.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 24.09.2021.

Ime i prezime studenta:	Luka Lovretić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4234, 26.07.2018.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Davor Vinko

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ:

1. UVOD

Mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom je sklop realiziran pomoću Arduino Uno mikroupravljača. Svrha sklopa jeste reakcija ukoliko senzor sklopa očita nekakav objekt na određenoj udaljenosti. Sklop se sastoji od šest glavnih elemenata: Arduino Uno mikroupravljača, ultrazvučnog senzora, zvučnika(*buzzer-a*), RGB svjetleće diode, LCD zaslona i tipkala. Reakcija nakon očitanja senzora rezultirana je oglašavanjem zvučnika i svijetljenjem RGB svjetleće diode. Pomoću tipkala(jednog) određujemo koje će boje svijetliti RGB svjetleća dioda, kolikom glasnoćom će se zvučnik oglasiti, kolika će biti svjetlina LCD zaslona i na kojoj će udaljenosti sklop reagirati. Cijeli izbornik biti će prikazan na LCD zaslonu.

1.1. Zadatak završnog rada

Zadatak završnog rada je razvoj i izrada mikroupravljačkog sustava za navigaciju izbornikom korištenjem jednog tipkala. Zadatak sustava je da poduzme određene radnje ovisno o očitavanju senzora, dok pomoću glavnog izbornika i tipkala korisnik može definirati karakteristike sustava.

2. ARDUINO UNO I ARDUINO IDE

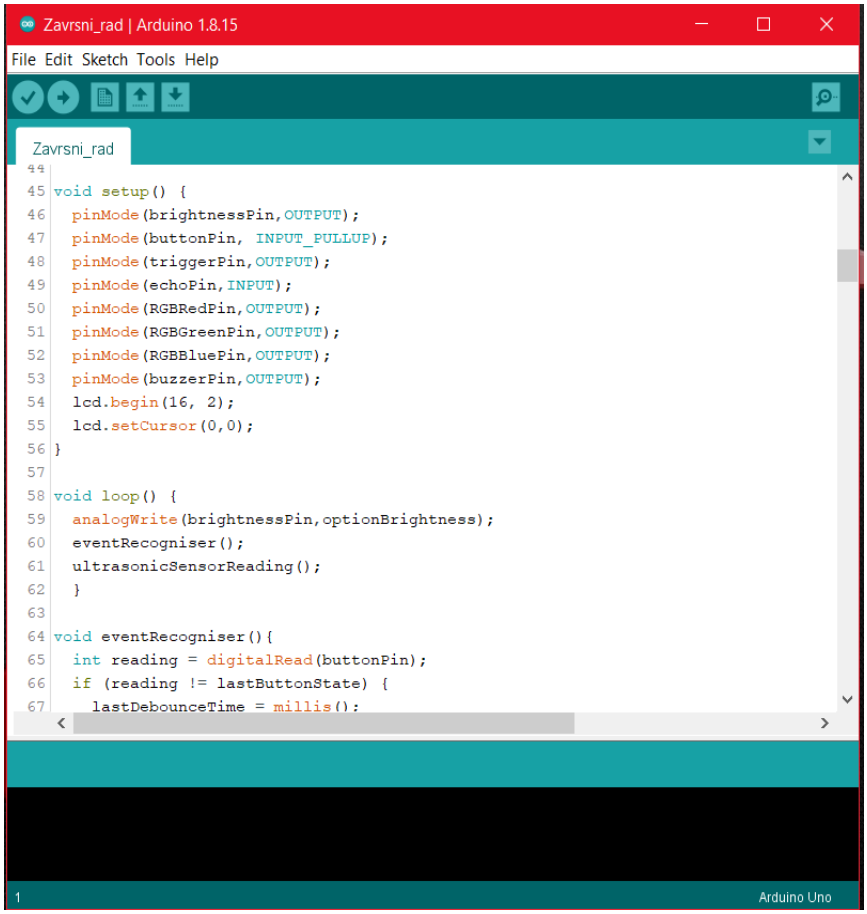
Arduino Uno je mikroupravljačka pločica zasnovana na Atmega328P. Sadrži dvanaest digitalnih *input*/*output* priključaka od kojih šest može biti korišteno kao PWM *output*-*i*. Sadrži također šest analognih priključaka, USB konekciju, priključak za napajanje, ICSP zaglavlje i gumb za resetiranje prema [1].

Sl. 2.1. Arduino Uno.

Tehničke specifikacije [2]:

- Radni napon: 5V
- Struja na pinovima: 40mA
- Struja na 3.3V izvoru: 50mA
- Memorija: 32kB
- SRAM: 2kB
- EEPROM: 1kB
- Frekvencija: 16MHz

Arduino pločice se uvijek mogu reprogramirati pomoću Arduino IDE(Arduino integrirano razvojno okruženje). Arduino IDE(*Integrated Development Environment*) je aplikacija napravljena u C i C++ programskom jeziku prema [3]. Služi za učitavanje programa na Arduino razvojne pločice. Besplatno je i javno dostupno korisničko sučelje za implementaciju i pisanje programske podrške u Arduino mikrokontrolere.



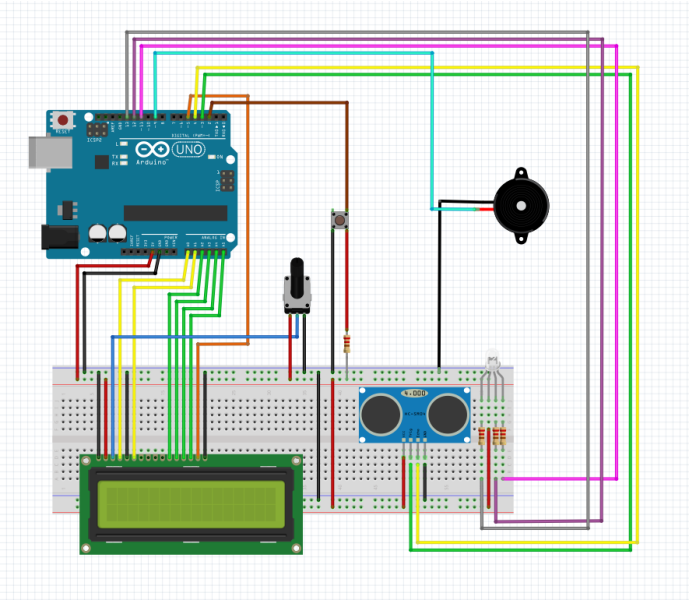
Sl. 2.2. Arduino IDE.

Programi napisani u Arduino IDE nazivaju se skice (engl. *Sketch*) i imaju ekstenziju .ino. Pritiskom na tipku *„Verify“* predvoditelj(engl. *Compiler*) pretvara kod iz programskog jezika u binarni strojni jezik. Nakon pritiska na *„Verify“*, pritiskom na „*Upload*“ program se prenosi na Arduino Uno pločicu, te nakon toga Arduino IDE daje povratnu informaciju o uspješnosti prijenosa podataka prema [4].

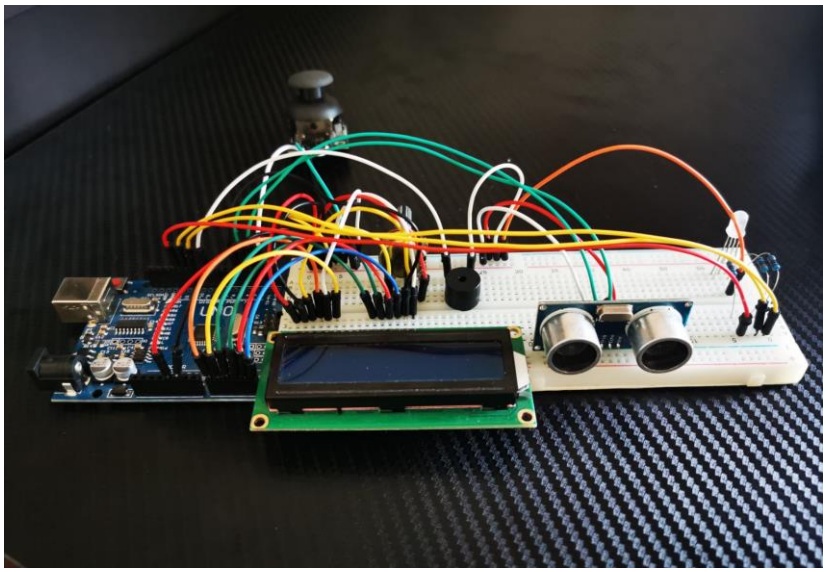
3. REALIZACIJA SUSTAVA

3.1. ELEKTRONIČKA SHEMA SPOJA

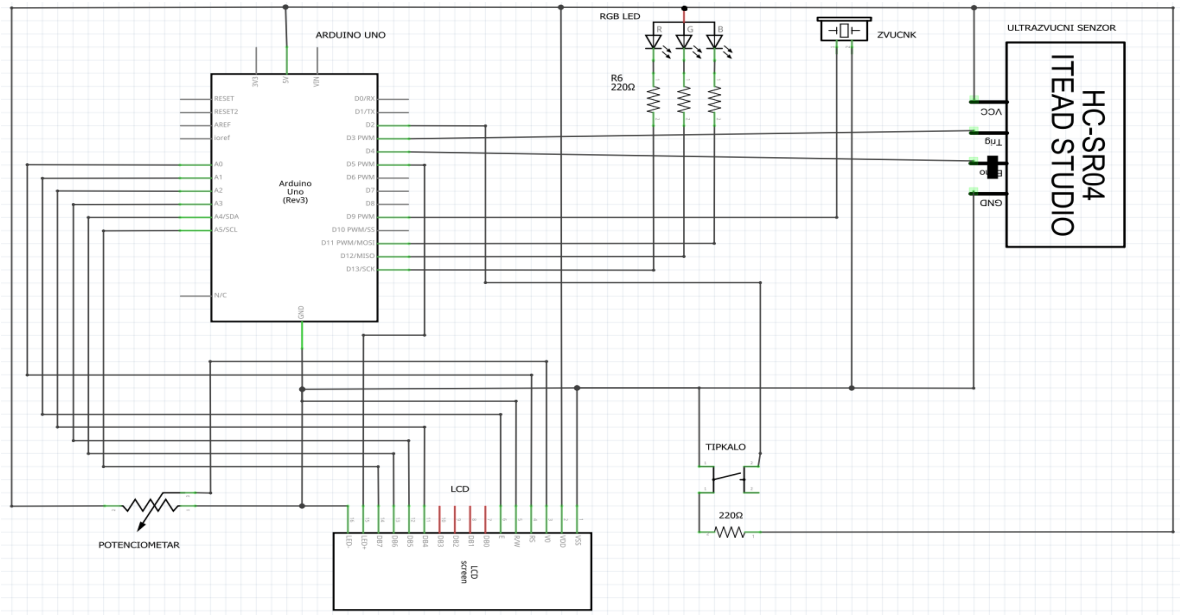
Shema spoja prikazuje točan raspored i spoj priključaka Arduino Uno razvojne pločice sa ostalim elementima sklopa. Također prikazan je izgled sheme uživo.



Sl. 3.1. Shema spoja (Fritzing).



Sl. 3.2. Stvarni izgled sklopa.



Sl. 3.3. Shema spoja.

Tablica 3.1. *Pinout* sklopa.

Arduino Uno priključak	Element (Priključak)
2	Tipkalo (SW)
3 (PWM)	UZ Senzor (TRIGGER)
4	UZ Senzor (ECHO)
5 (PWM)	LCD Zaslon (A)
9 (PWM)	Zvučnik (+)
11 (PWM)	RGB LED (B)
12	RGB LED (G)
13	RGB LED (R)
A0	LCD Zaslon (RS)
A1	LCD Zaslon (E)
A2	LCD Zaslon (D4)
A3	LCD Zaslon (D5)
A4	LCD Zaslon (D6)
A5	LCD Zaslon (D7)

3.2. *SETUP* I *LOOP* FUNKCIJE

Setup i *Loop* su funkcije unutar Arduino IDE. One su sastavni dijelovi Arduino kôda. Ono što je bitno napomenuti je to da se *Setup* funkcija izvršava samo jednom, a funkcija *Loop* se izvršava beskonačno mnogo puta, tj. ponaša se kao beskonačna petlja. U *Setup* funkciji se uglavnom definiraju varijable, započinju serijske komunikacije i definiraju se vrste priključaka, dok se u *Loop* funkciji definira ponašanje sklopa.

3.2.1. *Setup* Funkcija

```
Linija      Kôd
45:         void setup(){
46:            pinMode(brightnessPin,OUTPUT);
47:            pinMode(buttonPin,INPUT_PULLUP);
48:            pinMode(triggerPin,OUTPUT);
49:            pinMode(echoPin,INPUT);
50:            pinMode(RGBRedPin,OUTPUT);
51:            pinMode(RGBGreenPin,OUTPUT);
52:            pinMode(RGBBluePin,OUTPUT);
53:            pinMode(buzzerPin,OUTPUT);
54:            lcd.begin(16,2);
55:            lcd.setCursor(0,0); }
```

Sl. 3.4. Kôd *Setup* Funkcije.

PinMode je funkcija kojom se definira vrsta pina, može biti *INPUT* ,*OUTPUT* ili *INPUT_PULLUP*. Kao izlaze(engl. *output*) definirani su *brightnessPin*, *triggerPin*, *RGBRedPin*, *RGBBluePin*, *RGBGreenPin*, *buzzerPin*. Kao ulazi definirani su samo *buttonPin* koji je tipa *INPUT_PULLUP* i *echoPin* koji je tipa *INPUT*. Varijabla *lcd* je varijabla koja predstavlja LCD zaslon, pomoću nje može se upravljati što će biti ispisano na zaslon. Funkcija *begin* mora biti ispred svih funkcija vezanih za LCD zaslon, njom je definirano koliko ima redova i stupaca u LCD zaslonu. Funkcijom *setCursor* postavljen je pokaznik na nulti redak i nulti stupac. Sljedeća tablica prikazuje imena pin-ova na temelju na koji su priključak spojeni.

Tablica 3.2. *Pinout* sklopa sa imenima pinova u kôdu.

Arduino Uno priključak	Element (Priključak)	Naziv u Kôdu
2	Tipkalo (SW)	buttonPin
3 (PWM)	UZ Senzor (TRIGGER)	triggerPin
4	UZ Senzor (ECHO)	echoPin
5 (PWM)	LCD Zaslon (A)	brightnessPin
9 (PWM)	Zvučnik (+)	buzzerPin
11 (PWM)	RGB LED (B)	RGBBluePin
12	RGB LED (G)	RGBGreenPin
13	RGB LED (R)	RGBRedPin
A0	LCD Zaslon (RS)	registerSelectPin
A1	LCD Zaslon (E)	enablePin
A2	LCD Zaslon (D4)	digitalPin4
A3	LCD Zaslon (D5)	digitalPin5
A4	LCD Zaslon (D6)	digitalPin6
A5	LCD Zaslon (D7)	digitalPin7

3.2.2. *Loop* Funkcija

Linija ***Kôd***

```
58:        void loop(){
59:           analogWrite(brightnessPin,optionBrightness);
60:           eventRecogniser();
61:           ultrasonicSensorReading(); }
```

Sl. 3.5. Kôd *Loop* Funkcije.

Pomoću funkcije *analogWrite* pojačava/smanjuje se vrijednost svjetline LCD zaslona koja je spremljena u varijablu *optionBrightness*(može poprimiti vrijednosti 0-256). Funkcija *eventRecogniser* je jedna od dvije najvažnije funkcije ovog kôda. *EventRecogniser* je funkcija koja pri pritiskom na tipkalo prepoznaje radi li se o jednostrukom, dvostrukom, trostrukom ili

dugom pritisku. *UltrasonicSensorReading* je druga najvažnija funkcija ovog kôda. Njen zadatak je upravljanje nad ultrazvučnim senzorom. Pomoću *ultrasonicSensorReading* funkcije sklop zna na kojoj udaljenosti se nalazi nekakav objekt od ultrazvučog senzora, te na temelju te udaljenosti obavlja određene radnje.

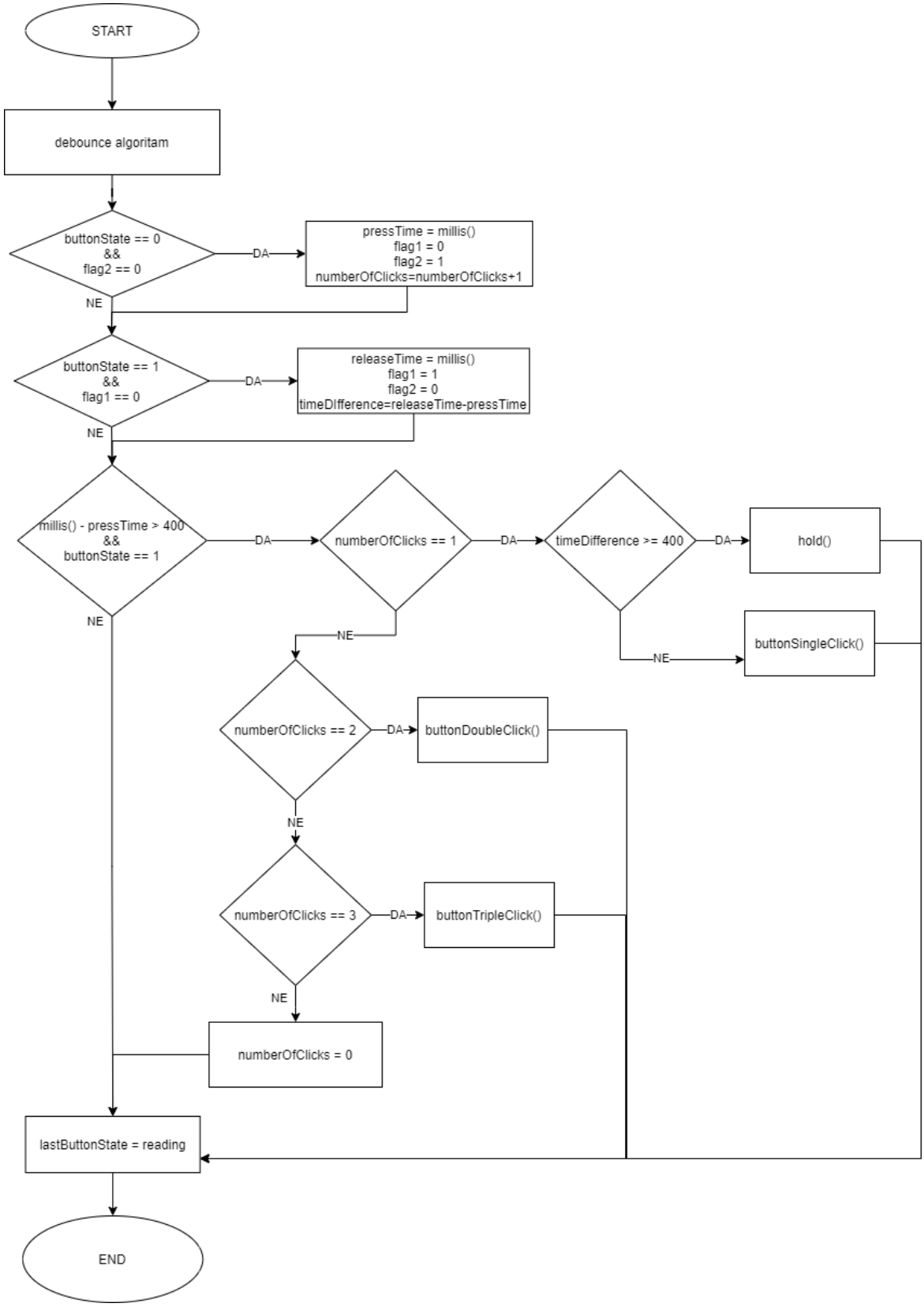
3.3. DETEKCIJA PRITISKA

Kao već ranije spomenuto, unutar *Loop* funkcije nalazi se funkcija koja se zove *eventRecogniser*. Njen cilj je detektirati radi li se o jednostrukom, dvostrukom, trostrukom ili dugom pritisku na tipkalo.

Linija ***Kôd***

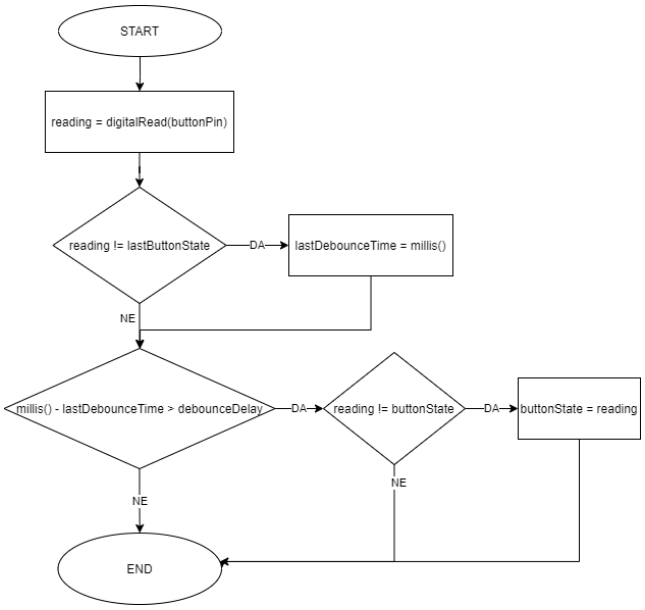
```
64:       void eventRecogniser(){
65:          int reading = digitalRead(buttonPin);
66:          if (reading != lastButtonState) {
67:             lastDebounceTime = millis(); }
68:          if ((millis() - lastDebounceTime) > debounceDelay) {
69:             if (reading != buttonState) {
70:                buttonState = reading; } }
71:          if (buttonState == 0 && flag2 == 0){
72:             pressTime = millis();
73:             flag1 = 0;
74:             flag2 = 1;
75:             numberOfClicks++; }
76:          if (buttonState == 1 && flag1 == 0){
77:             releaseTime = millis();
78:             flag1 = 1;
79:             flag2 = 0;
80:             timeDifference = releaseTime - pressTime; }
81:          if ((millis() - pressTime) > 400 && buttonState == 1){
82:             if (numberOfClicks == 1){
83:                if (timeDifference >= 400) {hold(); }
84:                else { buttonSingleClick(); } }
85:             else if (numberOfClicks == 2 ){buttonDoubleClick();}
86:             else if (numberOfClicks == 3) {buttonTripleClick();}
87:             numberOfClicks = 0; }
88:          lastButtonState = reading; }
```

Sl. 3.6. Kôd *eventRecogniser* funkcije.



Sl. 3.7. Dijagram toka *eventRecogniser* funkcije.

Debounce algoritam:



Sl. 3.8. Dijagram toka *debounce-a*.

Prvih sedam linija kôda predstavljaju klasični *debounce*. *Debounce* se nalazi u primjerima unutar Arduino IDE i cilj mu je da jedan pritisak ne očitava kao dva pritiska na tipkalo. Inače bez *debounce* algoritma program bi očitao pritisak na tipkalo kao prvi pritisak i puštanje tipkala kao drugi pritisak na tipkalo. Ostale linije kôda predstavljaju detekciju jednostrukog, dvostrukog, trostrukog i dugog pritiska na tipkalo. *ButtonState* je varijabla koja prikazuje trenutno stanje tipkala, je li pritisnuto ili ne, dok varijable *flag1* i *flag2* služe da bi se registriralo vrijeme pritiska i vrijeme otpuštanja tipkala. Ako je tipkalo pritisnuto i ako je *flag2* jednak nuli, u varijablu *pressTime* sprema se vrijeme u kojem je tipkalo pritisnuto i invertiraju se vrijednosti *flag1* i *flag2*, te se povećava vrijednost *numberOfClicks* za jedan. U drugom slučaju, ako je tipka otpuštena i *flag1* jednak nuli. To definira da je tipka definitivno otpuštena i u varijablu *releaseTime* se sprema vrijeme otpuštanja tipke i ponovno se invertiraju *flag1* i *flag2*, te se u varijablu *timeDifference* sprema razlika *pressTime* i *releaseTime* varijable. Pomoću *timeDifference* znati će se radi li se o jednostrukom ili dugom pritisku. U slučaju ako je otpušteno tipkalo i ako je trenutno vrijeme oduzeto sa vremenom pritiska veće od 400 milisekundi(vrijeme u kojem se sklop reagirati na pritisak), gleda se koliko se pritisaka dogodilo. Ukoliko je bio jedan pritisak i da je *timeDifference* manji od 400 milisekundi, zna se da se radi o jednostrukom pritisku, u suprotnom slučaju je dugi pritisak. Ako se pojavi dva ili tri pritiska na tipkalo, registrira se dvostruki i trostruki pritisak. Nakon svega, resetira se broj pritisaka na nulu i onda ponovno možemo pritiskati tipkalo.

3.4. FUNKCIJE UNUTAR *EVENTRECOGNISER* FUNKCIJE

Kao ranije spomenuto, unutar *eventRecogniser-a* imamo funkcije pomoću kojih definiramo kako će se sklop ponašati ukoliko se pojavi jednostruki, dvostruki, trostruki i dugi pritisak na tipkalo. Javljaju se funkcije *buttonSingleClick*, *buttonDoubleClick*, *buttonTripleClick* i *hold*.

3.4.1. ButtonSingleClick Funkcija

```
Linija      Kôd
103:        void buttonSingleClick{
105:           if(menuMode){
106:              switchMainElements();
107:              lcd.clear();
108:              lcd.setCursor(0,0)
109:              lcd.print(menuCurrentTargeted);}
111:           else{
112:              if(menuDistanceTargeted){
113:                 optionDistance+=1;
114:                 displayDistance();}
116:              else if(menuRGBColorTargeted){
117:                 RGBSwitchForward();
118:                 displayRGBColor();}
120:              else if(menuBrightnessTargeted){
121:                 if(optionBrightness<250){
122:                    optionBrightness+=25;
123:                    displayBrightness();}}
126:              else{
127:                 if(buzzerVolume<250){
128:                    buzzerVolume+=25;
129:                    displayVolume()}}}}
```

Sl. 3.9. Kôd *buttonSingleClick* funkcije.

ButtonSingleClick je funkcija unutar *eventRecogniser-a* koja se izvršava ukoliko se pojavi jednostruki pritisak na tipkalo. Pri pritisku na tipkalo, program najprije provjerava nalazi li se program u načinu rada glavnog izbornika ili u načinu rada namještanja (*menuMode* varijabla). Ukoliko je u načinu rada glavnog izbornika(*menuMode==true*), na LCD zaslonu ispisati će se opcija koju korisnik želi promijeniti. Ponuđene su četiri opcije: promjena udaljenosti na kojoj će sklop regirati(engl. *Change Distance*), promjena glasnoće zvučnika(engl. *Change Volume*), promjena boje RGB LED-ice (engl. *Change RGB Color*) i promjena svjetline LCD zaslona(engl. *Change Brightness*). Jednostrukim pritiskom na tipkalo, izmjenjuju se ponuđenje opcije. Ukoliko

se program nalazi u načinu rada namještanja(*menuMode==false*),znači da program nudi promjenu jedne od četiri ponuđene opcije. U način rada namještanja se ulazi tako da se napravi dvostruki pritisak na tipkalo dok je program u načinu rada glavnog izbornika. Kao već ranije spomenuto, moguće je mijenjati četiri varijable: svjetlinu zaslona, boju RGB LED-ice, udaljenost na kojoj će sklop reagirati i glasnoću zvučnika. Ukoliko je u načinu rada glavnog izbornika bila ponuđena promjena udaljenosti, dvostrukim pritiskom na tipkalo program je ušao u način rada namještanja za udaljenost na kojoj će sklop reagirati. Dok je program u načinu rada namještanja za udaljenosti, ukoliko se u tom trenutku pojavi jednostruki pritisak na tipkalo, varijabla *optionDistance*(varijabla koja predstavlja udaljenost na kojoj će sklop reagirati u centimetrima) se povećava za jedan centimetar. Slično vrijedi i za promjenu glasnoće zvučnika i promjenu svjetline zaslona. Ukoliko se program npr. nalazi u načinu rada namještanja za svjetlinu zaslona, jednostrukim pritiskom na tipkalo vrijednost varijable *optionBrightness* se povećava za 25. Maksimalna vrijednost varijable, tj. najjača svjetlina je 250, isto sve vrijedi i za promjenu jačine zvučnika, samo što se koristi varijabla *buzzerVolume*. Ako se program nalazi u načinu rada namještanja za boju RGB LED-ice, jednostrukim pritiskom na tipkalo varijabla *RGBState* mijenja svoje vrijednosti redoslijedom „crvena-zelena-plava“. *RGBState* je varijabla tipa *String* i poprima vrijednosti *„Red“* , *„Green“* ili *„Blue“*. Nakon svake promjene varijable pozivaju se funkcije koje na LCD zaslonu prikazuju trenutnu vrijednost određene varijable.

3.4.2. ButtonDoubleClick Funkcija

```
Linija    Kôd
136:      Void buttonDoubleClick{
138:         if(menuMode){
139:            menuMode=false;
140:            if(menuDistanceTargeted){displayDistance();}
141:            else if(menuRGBColoTargeted){displayRGBColor();}
142:            else if(menuLoudnessTargeted){displayVolume();}
143:            else{ displayBrightness();}}
145:         else{
146:            if(menuDistanceTargeted){
147:               if(optionDistance>1){
148:                  optionDistance-=1;}
150:               displayDistance();}
152:            else if(menuRGBColorTargeted){
153:               RGBSwitchBackward();
154:               displayRGBColor();}
156:            else if(menuBrightnessTargeted){
157:               if(optionBrightness>25){
158:                  optionBrightness-=25;
159:                  displayBrightness();}}
162:            else{
163:               if(buzzerVolume>0){
164:                  buzzerVolume-=25;
165:                  displayVolume();}}}}
```

Sl. 3.10. Kôd *buttonDoubleClick* funkcije

ButtonDoubleClick radi na sličan način kao i *buttonSingleClick* funkcija. Najprije provjerava nalazi li se program u načinu rada glavnog izbornika. Ako se nalazi(*menuMode==true*), automatski se prebacuje u način rada namještanja, te *menuMode* varijabla mijenja svoju vrijednost u *false*. Drugim riječima rečeno, program je prešao iz načina rada glavnog izbornika u način rada namještanja. Ukoliko je prije dvostrukog pritiska na tipkalo na izborniku bilo određeno namještanje udaljenosti(*Change Distance*), na LCD zaslonu će se ispisati udaljenost na kojoj će sklop reagirati. Isto vrijedi za promjenu boje RGB LED-ice, promjenu glasnoće zvučnika i promjenu svjetline LCD zaslona, samo što se prikazuju vrijednosti svjetline zaslona,jačine zvučnika i boje RGB LED-ice. U slučaju da pri dvostrukom pritisku na tipkalo program nije u načinu rada glavnog izbornika(*menuMode==false*), znači da je program već u načinu rada namještanja i da je promijenjena vrijednost jedne od varijabli koje se mogu mijenjati. Ukoliko se u načinu rada namještanja mijenja udaljenost na kojoj će sklop reagirati, dvostrukim pritiskom na tipkalo smanjuje se varijabla *optionDistance* za jedan centimetar(minimalna vrijednost je jedan centimetar) i ispisuje je nova vrijednost *optionDistance* varijable. Slično vrijedi za smanjivanje

varijable *buzzerVolume* i *optionBrightness*. Pri dvostrukim pritiskom na tipkalo smanjuju se za 25. Najmanja moguća vrijednost *buzzerVolume* varijable je nula(*MUTE*), dok je za *optionBrightness* dvadeset i pet(*MIN*). Što se tiče promjene boje RGB LED-ice, svakim dvostrukim pritiskom na tipkalo varijabla RGBState se mijenja u redoslijedu „crvena-plava-zelena“(suprotan smjer nego kod *buttonSingleClick*).

3.4.3. ButtonTripleClick Funkcija

Linija ***Kôd***

```
170:      void buttonTripleClick{
172:         if(!menuMode){
173:            lcd.clear();
174:            lcd.setCursor(0,0);
175:            lcd.print("Returning to");
176:            lcd.setCursor(0,1);
177:            lcd.print("   main menu...");
178:            delay(1500);
179:            lcd.clear();
180:            lcd.print(menuCurrentTargeted); }
182:         menuMode=true; }
```

Sl. 3.11. Kôd *buttonTripleClick* Funkcije.

ButtonTripleClick funkcija poziva se nakon trostrukog pritiska na tipkalo. Njen cilj je vratiti korisnika natrag u glavni izbornik, tj. prebaciti program iz načina rada namještanja u način rada glavnog izbornika. Bitno je napomenuti da se ova funkcija izvršava samo kada je program u načinu rada namještanja. Znači da, ukoliko se napravi trostruki pritisak na tipkalo u načinu rada glavnog izbornika, ništa se neće dogoditi. Funkcija najprije provjerava nalazi li se program u načinu rada namještanja. Ukoliko se nalazi,znači da je korisnik završio sa namještanjem jedne od četiri ponuđene vrijednosti. Nakon trostrukog pritiska na tipkalo, na LCD zaslonu se prikazuje tekst „*Returning to main menu...*“, što znači da se korisnik vraća na glavni izbornik. Funkcija također koristi *delay* funkciju kako bi ispisane riječi na LCD zaslonu stajale duže nego inače iz razloga da korisnik stigne pročitati što se dogodilo nakon trostrukog pritiska na tipkalo. Trajanje *delay-a* je 1500 milisekundi, što se sasvim dovoljno korisniku da pročita poruku na LCD zaslonu. Nakon *delay* funkcije, u glavnom izborniku se ponovo nudi namještanje ranije namještene vrijednosti. Nakon svih odrađenih radnji funkcije, varijabla *menuMode* poprima vrijednost *true*, što znači da se uključio način rada glavnog izbornika.

3.4.4. *Hold* Funkcija

Linija ***Kôd***

```
185:        void hold(){
186:           lcd.clear();
187:           lcd.setCursor(0,0);
188:           lcd.print("Reseting to");
189:           lcd.setCursor(0,1);
190:           lcd.print("default values...");
191:           delay(1500);
192:           lcd.clear();
193:           RGBstate=defaultRGBstate;
194:           optionDistance=defaultDistance;
195:           buzzerVolume=defaultBuzzerVolume;
196:           optionBrigthness=defaultBrightness;
197:           menuMode=true;
198:           lcd.print(menuCurrentTargeted); }
```

Sl. 3.12. Kôd *hold* Funkcije.

Hold funkcija izvršava se nakon dugog pritiska na tipkalo(ukoliko korisnik drži tipkalo 400 ms ili duže). Bitno je napomenuti da ova funkcija nije ovisna u kojem se načinu rada program nalazi, znači da je svejedno nalazi li se program u načinu rada glavnog izbornika ili u načinu rada namještanja, ova funkcija će jednako raditi. Dugim pritiskom na tipkalo sve vrijednosti resetiraju se na zadanu vrijednost. Funkcija prvo ispisuje poruku na LCD zaslon „*Reseting to default values*“, te nakon toga se varijable boje RGB LED-ice, zvučnika, LCD zaslona i ultrazvučnog senzora resetiraju na nekakvu predodređenu vrijednost. Zadana vrijednost za boju RGB LED-ice je „*Red*“(varijabla *defaultRGBState*), za jačinu zvučnika je 50 (varijabla *defaultBuzzerVolume*), za svjetlinu zaslona je 100 (varijabla *defaultBrightness*) i za očitavanje ultrazvučnog senzora je 7 centimetara(varijabla *defaultDistance*). Nakon resetiranja, program se nalazi u načinu rada glavnog izbornika.

3.4.5. Ostale Funkcije

Među ostalim funkcijama javljaju se jednostavne funkcije za navigaciju izbornikom i za ispisivanje trenutih vrijednost varijabli. Javljaju se funkcije *switchMainElements*, *RGBSwitchForward*, *RGBSwitchBackward*, *displayBrightness*, *displayVolume*, *displayDistance*, *displayRGBColor*.

3.4.6. SwitchMainElements Funkcija

Zadatak ove funkcije je da pri jednostrukom pritisku na tipkalo prelazi s jednog elementa glavnog izbornika na drugi u redoslijedu: Udaljenost(*Distance*)-RGB Boja(*RGB Color*) - Glasnoća(*Volume*) – Svjetlina (*Brightness*).

Linija ***Kôd***

```
211:          void switchMainElements(){
212:             if(menuDistanceTargeted){
213:                menuDistanceTargeted=false;
214:                menuRGBColorTargeted=true;
215:                menuCurrentTargeted="Change RGB Color"; }
217:             else if(menuRGBColorTargeted){
218:                menuRGBColorTargeted=false;
219:                menuLoudnessTargeted=true;
220:                menuCurrentTargeted="Change Volume";}
222:             else if(menuLoudnessTargeted){
223:                menuLoudnessTargeted=false;
224:                menuBrightnessTargeted=true;
225:                menuCurrentTargeted="Change Brightness"; }
227:             else{
228:                menuBrightnessTargeted=false;
229:                menuDistanceTargeted=true;
230:                menuCurrentTargeted="Change Distance"; }}
```

Sl. 3.13. Kôd *switchMainElements* Funkcije.

3.4.7. *RGBSwitchForward* i *RGBSwitchBackward* Funkcije

Zadatak ove dvije funkcije je promjena boje RGB LED-ice. Jedina razlika među njima smjer u kojem se boje mijenjaju. Pri *RGBSwitchForward* smjer promjene boja je „R-G-B“, dok je kod *RGBSwitchBackward* „R-B-G“. *RGBSwitchForward* se izvršava kada se pri modu namještanja boje RGB LED-ice dogodi jednostruki pritisak na tipkalo, dok za *RGBSwitchBackward* je potreban dvostruki pritisak na tipkalo.

Linija ***Kôd***

```
199:    void RGBSwitchForward(){
200:       if(RGBstate=="Red"){RGBstate="Green";}
201:       else if(RGBstate=="Green"){RGBstate="Blue";}
202:       else{ RGBstate="Red";}
205:    void RGBSwitchBackward(){
206:       if(RGBstate=="Red"){RGBstate="Blue";}
207:       else if(RGBstate=="Green"){RGBstate="Red";}
208:       else{ RGBstate="Green";}
```

Sl. 3.14. Kôd funkcija *RGBSwitchForward* i *RGBSwitchBackward*

3.4.8. DisplayRGBColor, DisplayDistance, DisplayVolume i DisplayBrightness Funkcije

Navedene funkcije služe za ispisivanje trenutnih vrijednosti varijabli boje RGB LED-ice, svjetline LCD zaslona, udaljenosti na kojoj će sklop reagirati i glasnoće zvučnika.

Linija ***Kôd***

```
309:      void displayRGBColor(){
310:         lcd.clear();
311:         lcd.setCursor(0,0);
312          lcd.print("Color: ");
313:         lcd.print(RGBstate); }
316:      void displayDistance(){
317:         lcd.clear();
318:         lcd.setCursor(0,0);
319:         lcd.print("Distance: ");
320:         lcd.print(optionDistance);
321:         lcd.print("cm"); }
324:      void displayVolume(){
325:         lcd.clear();
326:         lcd.setCursor(0,0);
327:         lcd.print("Volume: ");
328:         if(buzzerVolume==0){
329:            lcd.print("MUTE"); }
331:         else if(buzzerVolume==250){
332:            lcd.print("MAX"); }
334:         else{
335:            lcd.print(buzzerVolume); }}
339:      void displayBrightness(){
340:         lcd.clear();
341:         lcd.setCursor(0,0);
342:         lcd.print("Brightness: ");
343:         if(optionBrightness==25){
344:            lcd.print("MIN"); }
346:         else if(optionBrightness==250){
347:            lcd.print("MAX"); }
349:         else{
350:            lcd.print(optionBrightness); }}
```

Sl. 3.15. Kôd funkcija za ispisivanje na LCD zaslon.

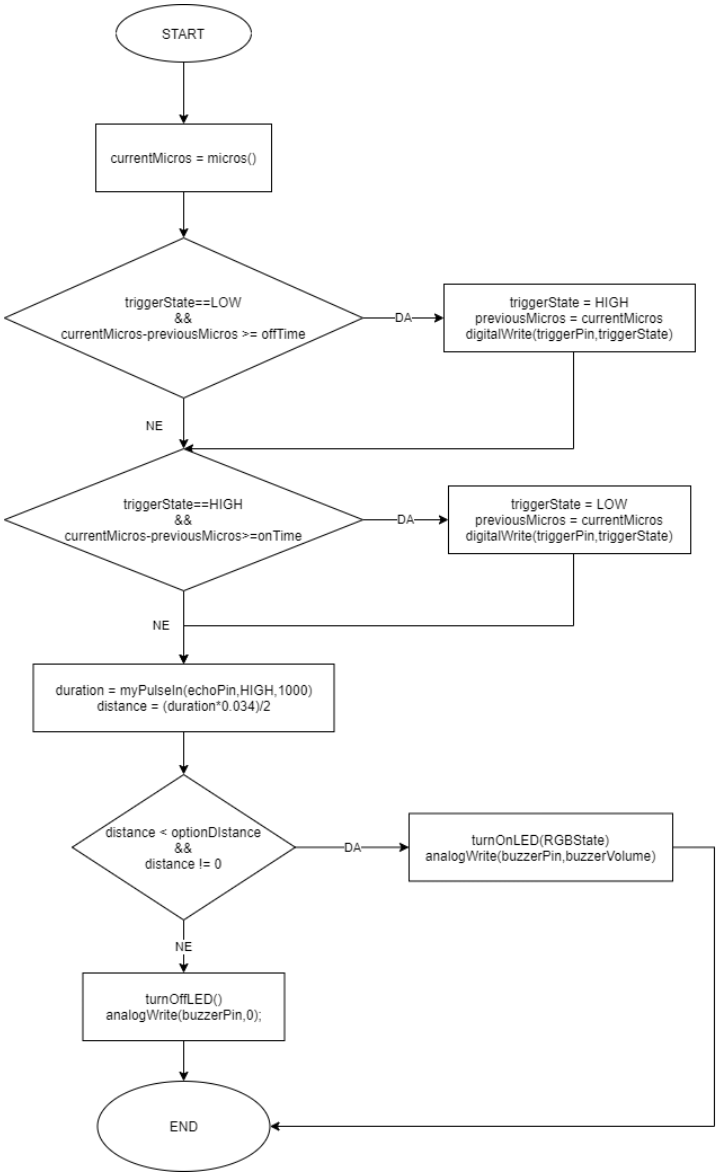
3.5. OČITAVANJE ULTRAZVUČNOG SENZORA

Unutar *Loop* funkcije se također nalazi i *ultrasonicSensorReading* funkcija koja očitava udaljenost nekog objekta od ultrazvučnog senzora. Ono što je bitno napomenuti je to da funkcija u sebi ne smije sadržavati nikakve *delay* funkcije jer bi poremetila rad *eventRecogniser* funkcije. Kôd funkcije na slici 3.16:

Linija ***Kôd***

```
234:    void ultrasonicSensorReading(){
235:       unsigned long currentMicros = micros();
236:       if(triggerState == LOW && currentMicros - previousMicros >=
        offTime){
237:          triggerState = HIGH;
238:          previousMicros = currentMicros;
239:          digitalWrite(triggerPin, triggerState);  }
240:       else if(triggerState == HIGH && currentMicros - previousMicros
        >= onTime){
241:          triggerState = LOW;
242:          previousMicros = currentMicros;
243:          digitalWrite(triggerPin, triggerState); }
244:       duration = myPulseIn(echoPin,HIGH,1000);
245:       distance = (duration*0.034)/2;
256:       if(distance < optionDistance && distance != 0){
247:          turnOnLED(RGBstate);
248:          analogWrite(buzzerPin,buzzerVolume); }
249:       else{
250:          turnOffLED();
251:          analogWrite(buzzerPin,0); }
```

Sl. 3.16. Kôd *ultrasonicSensorReading* funkcije.



Sl. 3.17. Dijagram toka ultrasonicSensorReading funkcije.

Ovo je malo kompliciraniji izraz od klasičnog korištenja ultrazvučnog senzora zbog toga što se ne smije u njemu imati nikakve *delay* funkcije, jer bi poremetile rad *eventRecogniser* funkcije. Najprije se uzima trenutno vrijeme u mikrosekundama, te ako je okidač(engl. *trigger*)ugašen i ako je razlika trenutnog vremena u mikrosekundama i prethodnog vremena u mikrosekundama veća ili jednaka od vremena gašenja okidača(što je dvije mikrosekunde), pali se okidač. Ovim dijelom kôda osigurano je da je okidač ugašen na dvije mikrosekunde. Ako je okidač upaljen i ako je razlika trenutnog i prethodnog vremena u mikrosekundama veća ili jednaka od 50, gasi se okidač. Na ovaj način je osigurano da je okidač upaljen pedeset mikrosekundi i da je ugašen nakon toga.

U varijablu *duration* sprema se vrijeme trajanja puta ultrazvučnog signala pomoću *myPulseIn* funkcije. *MyPulseIn* funkcija ima istu svrhu kao i *pulseIn* funkcija, samo što *myPulseIn* ne sadrži u sebi nikakve *delay* funkcije. Na temelju *duration* varijable i brzine zvuka, može se izračunati udaljenost od ultrazvučnog senzora do objekta, prema slici 3.16. ,linija kôda 245. Ako je udaljenost manja od opcionalne udaljenosti(udaljenost koju sam korisnik namješta preko izbornika), sklop će reagirati tako da će RGB svjetleća dioda svijetliti u određenoj boji (koju također korisnik namješta preko izbornika) i zvučnik će se oglasiti određenom glasnoćom. U suprotnom slučaju, RGB svjetleća dioda će se ugasiti, a zvučnik neće proizvoditi nikakav zvuk.

3.6. FUNKCIJE UNUTAR *ULTRASONICSENSORREADING* FUNKCIJE

Kao već ranije spomenuto, *ultrasonicSensorReading* funkcija služi za upravljanje ultrazvučnim senzorom. Unutar nje ne nalazi se mnogo funkcija, najbitnija za spomeniti je *myPulseIn* funkcija.

3.6.1. *MyPulseIn* Funkcija

MyPulseIn funkcija je posebna funkcija koja računa koliko je vremena prošlo od slanja ultrazvučnog signala do njegovog vraćanja. Posebna je po tome što nije kao obična *pulseIn* funkcija. Unutar nje se ne nalazi nikakav *delay* koji bi inače poremetio rad *eventRecogniser* funkcije.

Linija ***Kôd***

```
261:    unsigned long myPulseIn(int pin, int value, int timeout) {
262:       unsigned long currentMicros = micros();
263:       while(digitalRead(pin) == value) {
264:          if (micros() - currentMicros > (timeout*1000)) {
265:             return 0; }}
268:       currentMicros = micros();
269:       while (digitalRead(pin) != value) {
270:          if (micros() - currentMicros > (timeout*1000)) {
271:             return 0; }}
274:       currentMicros = micros();
275:       while (digitalRead(pin) == value) {
276:          if (micros() - currentMicros > (timeout*1000)) {
277:             return 0; }}
280:       return micros()-currentMicros; }
```

Sl. 3.18. Kôd *myPulseIn* funkcije.

3.6.2. *TurnOnLED* i *TurnOffLED* Funkcije

Dvije navedene funkcije služe za paljenje i gašenje RGB LED-ice. *TurnOnLED* je moguće pozvati sa parametrom tipa *String* koji može poprimiti vrijednosti „*Red*“, „*Green*“ i „*Blue*“.

Linija ***Kôd***

```
284:   void turnOnLED(String color){
285:      if(color=="Red"){
286:         digitalWrite(RGBRedPin,HIGH);
287:         digitalWrite(RGBGreenPin,LOW);
288:         digitalWrite(RGBBluePin,LOW); }
290:      else if(color=="Green"){
291:         digitalWrite(RGBGreenPin,HIGH);
292:         digitalWrite(RGBBluePin,LOW);
293:         digitalWrite(RGBRedPin,LOW); }
295:      else{
296:         digitalWrite(RGBBluePin,HIGH);
297:         digitalWrite(RGBRedPin,LOW);
298:         digitalWrite(RGBGreenPin,LOW); }}
302:   void turnOffLED(){
303:      digitalWrite(RGBRedPin,LOW);
304:      digitalWrite(RGBGreenPin,LOW)
305:      digitalWrite(RGBBluePin,LOW); }
```

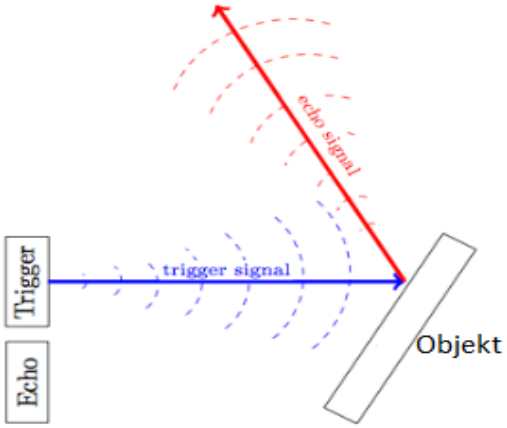
Sl. 3.19. Kôd *turnOnLED* i *turnOffLED* funkcije.

4. MOGUĆI PROBLEMI I POTEŠKOĆE

Kao i pri svakom kreiranju nekakvog mikroupravljačkog sklopa, uvijek se mogu dogoditi nekakve poteškoće i problemi, pa tako i na ovome. Problemi mogu biti programski ili izvedbeni.

4.1.1. Očitavanje udaljenosti od objekta pod kutom

Ultrazvučni senzor ima na sebi dva elementa, jedan koji šalje ultrazvučni signal i jedan koji ga očitava nakon što se signal odbije od objekt(prepreku). Problem je u tome što je ultrazvučni senzor dizajniran tako da može očitavati udaljenosti tek kada je površina objekta okomita na smjer gibanja ultrazvučnog vala.Sljedeća slika prikazuje što se događa kada je objekt pod nekim kutom naspram smjera signala(Slika 4.1.).



Sl. 4.1. Odbijanje signala od objekt.

Kao što je prikazano, signal se odbija u drugom smjeru jer se površina prepreke ne nalazi okomito naspram trigger signala. U ovim slučajevima signal obično putuje predugo, pa vrijednost varijable koju očitava *pulseIn* (*myPulseIn*) funkcija bude netočno. Ovakva situacija obično rezultira netočnim očitavanjem ili ne očitavanjem vrijednosti udaljenosti prema [5].

4.1.2. Materijal objekta(prepreke)

Mora se uzeti u obzir i materijal objekta. Postoje materijali koji apsorbiraju signal okidača. Korištenje takvih objekata rezultira netočnim očitavanjem senzora prema [5].

4.1.3. Minimalna udaljenost objekta od ultrazvučnog senzora

Ultrazvučni senzor ima svoju minimalnu udaljenost koju može očitati. Minimalna udaljenost je dva centimetra. U području manjem od dva centimetra udaljenosti od okidača, javlja se mrtva zona(engl. *Dead zone*). Mrtva zona odnosi se na područje neposredno ispred okidača(engl. *Trigger*) signala na kojem senzor ne može pouzdano mjeriti. Zbog kuta mjerenja od 15° i frekvencije ultrazvučnog vala prema [6] ,senzor nije u mogućnosti očitati odbijeni signal na udaljenostima manjim od dva centimetra.

4.1.4. Maksimalna udaljenost objekta od ultrazvučnog senzora

Kao što ima minimalnu, tako ultrazvučni senzor ima i maksimalnu udaljenost na kojoj može mjeriti. Što je viša frekvencija rada, manje su valne duljine, što rezultira manjim dometom prema [7]. Za HC-SR04 (40kHz) koji je korišten u sklopu, maksimalni domet je četiri metra.

4.1.5. Korištenje *delay* funkcije

U ovom programu nisu dopuštene nikakve funkcije koje odgađaju programske operacije na nekakvo određeno vrijeme. Uglavnom, ovdje govorimo o *delay* funkciji koja u sebi kao parametar sadrži vrijeme odgode operacija u milisekundama. Iznimka je delay od 1500 milisekundi koji koriste funkcije *hold* i *buttonTripleClick*. U njima se javlja *delay* iz razloga da bi korisnik mogao stići pročitati poruke ispisane na LCD zaslonu. Te *delay* funkcije zapravo remete rad sustava, ali s obzirom da se u to vrijeme korisnik bavi čitanjem poruke sa LCD zaslona, nije toliko značajan problem.

5. ZAKLJUČAK

U ovom završnom radu napravljen je mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom. Nakon razvijanja ideje kako bi ovaj sustav trebao raditi, izrađena je elektronička shema, te su nabavljene sve potrebne komponente za izradu sklopa. Prvi korak realizacije sklopa bilo je postavljanje svih komponenti na *breadboard* na temelju osmišljene elektroničke sheme. Spajanjem mikroupravljača sa komponentama na *breadboard-u* ,dobili smo jednu elektroničku cjelinu kojoj je bio potreban program koji bi definirao njeno ponašanje. Drugi dio realizacije sustava temeljio se isključivo na programiranju. Koristeći različite funkcije i algoritme u programskom kôdu i prenošenjem istog na mikroupravljač, postignuto je da sklop uspješno izvršava sve očekivane zadatke. Sklop se može koristiti kao nekakav alarm kojeg je moguće prilagoditi korisnikovim željama.

LITERATURA

[1] „Arduino Uno Pinout“, dostupno na: https://content.arduino.cc/assets/Pinout-UNOrev3_latest.pdf [31.8.2021.]

[2] „Arduino Uno Specs“, dostupno na: https://components101.com/microcontrollers/arduinouno [31.8.2021.]

[3] „Arduino IDE - Wikipedia“, dostupno na: https://en.wikipedia.org/wiki/Arduino_IDE [31.8.2021.]

[4] „Arduino IDE“ , dostupno na: https://www.arduino.cc/en/guide/environment [31.8.2021.]

[5] „HC-SR04: ultrasonic sensor for Arduino“, dostupno na: https://macduino.blogspot.com/2013/11/HC-SR04-part1.html [31.8.2021.]

[6] „Detection range of ultrasonic sensors“, dostupno na: https://www.sensorpartners.com/en/knowledge-base/detection-range-of-an-ultrasonic-sensor/ [1.9.2021.]

[7] „Wavelenght“, dostupno na: https://scied.ucar.edu/learning-zone/atmosphere/wavelength [1.9.2021.]

ŽIVOTOPIS

Autor ovog završnog rada, Luka Lovretić ,student je treće godine Sveučilišnog preddiplomskog studija Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek(FERIT). Prije upisivanja na FERIT, bio je učenik Tehičke škole Ruđera Boškovića u Vinkovcima smjera Tehničar za mehatroniku.

_____

Potpis autora

SAŽETAK

Naslov: Mikroupravljački sustav s jednim tipkalom za navigaciju izbornikom

Cilj završnog rada bio je razvoj i izrada mikroupravljačkog sustava s jednim tipkalom za navigaciju izbornikom. Sustav je baziran na Arduino Uno mikroupravljaču i Arduino integriranom razvojnom okruženju. Sklop se sastoji od šest glavnih komponenti: Arduino Uno mikroupravljača, tipkala, LCD zaslona, RGB svjetleće diode, zvučnika(*buzzer-a*) i ultrazvučnog senzora. Kombinacijom navedenih komponenti sklopljen je sustav koji se ponaša kao alarm kojeg je moguće prilagoditi korisnikovim željama. Sklop će reagirati svjetlosnim i zvučnim signalom ukoliko se nekakav objekt nalazi na određenoj udaljenosti od senzora. Korisnik je u mogućnosti pomoću glavnog izbornika na LCD zaslonu mijenjati karakteristike sklopa prema svojim željama kao što su: udaljenost na kojoj će sklop reagirati, boja svjetlosnog signala, jačina zvučnog signala i svjetlina LCD zaslona. Navigacija po izborniku odvija se isključivo preko jednog jedinog tipkala. Definiranjem različitih vrsta pritisaka na tipkalo, korisnik je u mogućnosti kretati se izbornikom i mijenjati vrijednosti sustava.

Ključne riječi: Arduino, izbornik, LCD zaslon, tipkalo, ultrazvučni senzor

ABSTRACT

Title: Microcontroller system with one button for menu navigation

The main goal of this final thesis was development and making of a microcontroller system with a single button for menu navigation. The system is based on a Arduino Uno microcontroller and Arduino integrated development environment. The system is made of six main components: Arduino Uno microcontroller, pushbutton, LC display, RGB light emitting diode, buzzer and ultrasonic sensor. With a combination of these components we made a system which behaves like an alarm which can be adjusted depending on user preferences. The system will react with a light and sound signal if any object is located at a certain distance from the ultrasonic sensor. The user is able to change the characteristics of the system according to their preferences using the main menu on the LCD screen like: the distance on which the system will react, the color of the light signal, the volume of the sound signal and the brightness of the LCD screen. Navigation through the menu takes place exclusively via a single button. By defining different types of button-presses, the user is able to navigate the menu and change system values.

Keywords: Arduino, menu, LC display, button, ultrasonic sensor

PRILOZI

P1: Cijeli Arduino Kôd

```
#include <LiquidCrystal.h>
#include <String.h>
#define buttonPin 2
#define triggerPin 3
#define echoPin 4
#define brightnessPin 5
#define buzzerPin 9
#define RGBRedPin 13
#define RGBGreenPin 12
#define RGBBluePin 11
#define registerSelectPin A0
#define enablePin A1
#define digitalPin4 A2
#define digitalPin5 A3
#define digitalPin6 A4
#define digitalPin7 A5
LiquidCrystal lcd(registerSelectPin, enablePin, digitalPin4, digitalPin5,
digitalPin6, digitalPin7);

//varijable za ocitavanje klika
boolean lastButtonState = HIGH, buttonState=HIGH, flag1, flag2;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;
byte numberOfClicks;
int timeDifference;
long double pressTime, releaseTime;

//varijable glavnog izbornika
boolean menuDistanceTargeted=true, menuBrightnessTargeted=false,
menuRGBColorTargeted=false, menuLoudnessTargeted=false;
boolean menuMode=true;
String menuCurrentTargeted="Change Distance";

//varijable za ultrazvucni senzor
boolean triggerState = LOW;
int offTime = 2,onTime = 10,distance;
unsigned long previousMicros = 0;
long duration;

//varijable za glasnocu,svjetnlinu,glasnocu i stanje RBG LED-ice
String RGBstate="Red",defaultRGBstate="Red";
int buzzerVolume=50,defaultBuzzerVolume=50;
int optionDistance=7,defaultDistance=7;
int optionBrightness=100,defaultBrightness=100;


void setup() {
  pinMode(brightnessPin,OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(triggerPin,OUTPUT);
  pinMode(echoPin,INPUT);
  pinMode(RGBRedPin,OUTPUT);
  pinMode(RGBGreenPin,OUTPUT);
  pinMode(RGBBluePin,OUTPUT);
```

```
  pinMode(buzzerPin,OUTPUT);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
}

void loop() {
  analogWrite(brightnessPin,optionBrightness);
  eventRecogniser();
  ultrasonicSensorReading();
  }

void eventRecogniser(){
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
    }
  }
  if (buttonState == 0 && flag2 == 0)
  {
    pressTime = millis();
    flag1 = 0;
    flag2 = 1;
    numberOfClicks++;
  }
  if (buttonState == 1 && flag1 == 0)
  {
    releaseTime = millis();
    flag1 = 1;
    flag2 = 0;
    timeDifference = releaseTime - pressTime;
  }

  if ((millis() - pressTime) > 400 && buttonState == 1)
  {
    if (numberOfClicks == 1)
    {
      if (timeDifference >= 400) {hold(); }
      else { buttonSingleClick(); }
    }
    else if (numberOfClicks == 2 ){buttonDoubleClick();}
    else if (numberOfClicks == 3) {buttonTripleClick();}
    numberOfClicks = 0;
    }
  lastButtonState = reading;
  }

void buttonSingleClick()
{
  if(menuMode){
    switchMainElements();
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(menuCurrentTargeted);
  }
  else{
    if(menuDistanceTargeted){
```

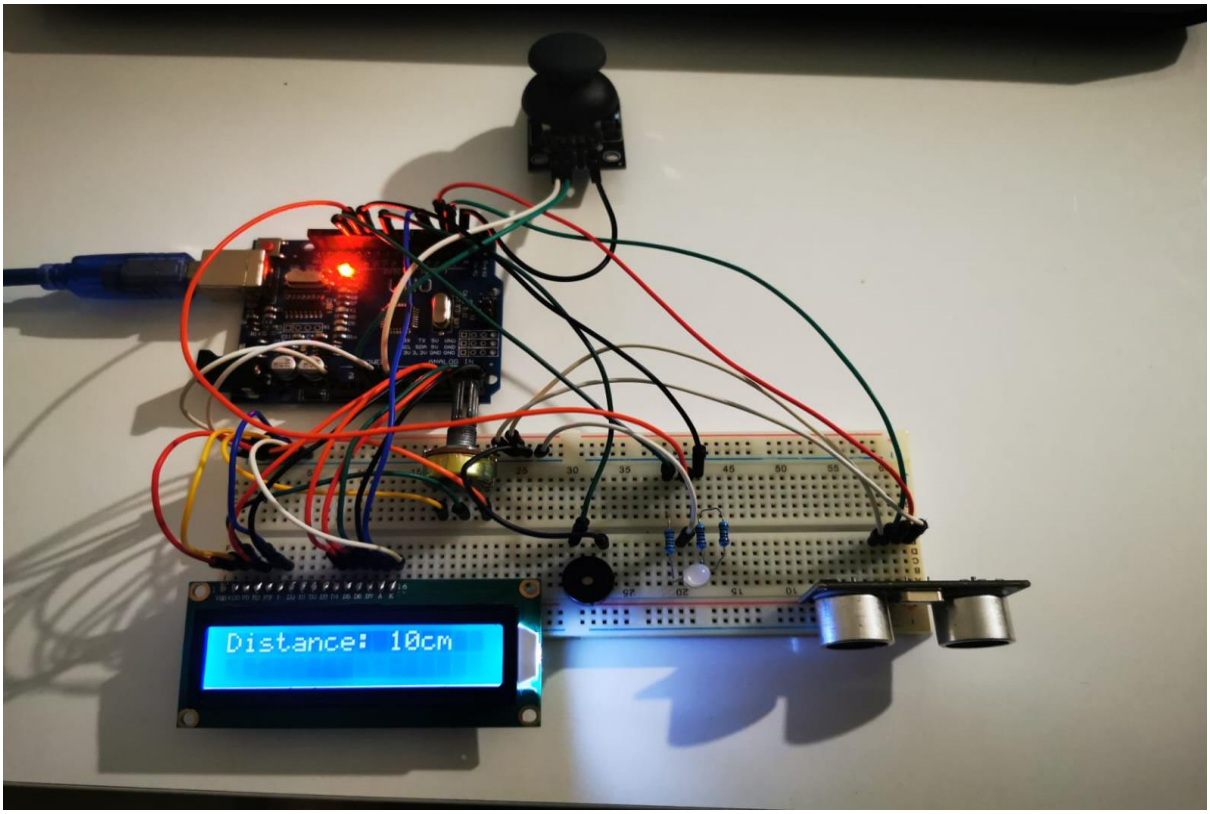
```
      optionDistance+=1;
      displayDistance();
       }
    else if(menuRGBColorTargeted){
      RGBSwitchForward();
      displayRGBColor();
      }
    else if(menuBrightnessTargeted){
      if(optionBrightness<250){
        optionBrightness+=25;
        displayBrightness();
      }
    }
    else{
      if(buzzerVolume<250){
        buzzerVolume+=25;
        displayVolume();
      }
    }
  }
}


void buttonDoubleClick()
{
  if(menuMode){
    menuMode=false;
    if(menuDistanceTargeted){displayDistance();}
    else if(menuRGBColorTargeted){displayRGBColor();}
    else if(menuLoudnessTargeted){displayVolume();}
    else{ displayBrightness();}
    }
    else{
      if(menuDistanceTargeted){
        if(optionDistance>1){
          optionDistance-=1;
        }
        displayDistance();
        }
      else if(menuRGBColorTargeted){
        RGBSwitchBackward();
        displayRGBColor();
        }
      else if(menuBrightnessTargeted){
        if(optionBrightness>25){
          optionBrightness-=25;
          displayBrightness();
        }
      }
      else{
        if(buzzerVolume>0){
          buzzerVolume-=25;
          displayVolume();
        }
      }
    }
}
void buttonTripleClick()
{
  if(!menuMode){
```

```
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Returning to");
    lcd.setCursor(0,1);
    lcd.print("   main menu...");
    delay(1500);
    lcd.clear();
    lcd.print(menuCurrentTargeted);
    }
  menuMode=true;
}

void hold(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Reseting to");
  lcd.setCursor(0,1);
  lcd.print("default values...");
  delay(1500);
  lcd.clear();
  RGBstate=defaultRGBstate;
  optionDistance=defaultDistance;
  buzzerVolume=defaultBuzzerVolume;
  optionBrightness=defaultBrightness;
  menuMode=true;
  lcd.print(menuCurrentTargeted);
}

void RGBSwitchForward(){
  if(RGBstate=="Red"){RGBstate="Green";}
  else if(RGBstate=="Green"){RGBstate="Blue";}
  else{ RGBstate="Red";}
}

void RGBSwitchBackward(){
  if(RGBstate=="Red"){RGBstate="Blue";}
  else if(RGBstate=="Green"){RGBstate="Red";}
  else{ RGBstate="Green";}
}

void switchMainElements(){
    if(menuDistanceTargeted){
      menuDistanceTargeted=false;
      menuRGBColorTargeted=true;
      menuCurrentTargeted="Change RGB Color";
    }
    else if(menuRGBColorTargeted){
      menuRGBColorTargeted=false;
      menuLoudnessTargeted=true;
      menuCurrentTargeted="Change Volume";
    }
    else if(menuLoudnessTargeted){
      menuLoudnessTargeted=false;
      menuBrightnessTargeted=true;
      menuCurrentTargeted="Change Brightness";
    }
    else{
      menuBrightnessTargeted=false;
      menuDistanceTargeted=true;
      menuCurrentTargeted="Change Distance";
```

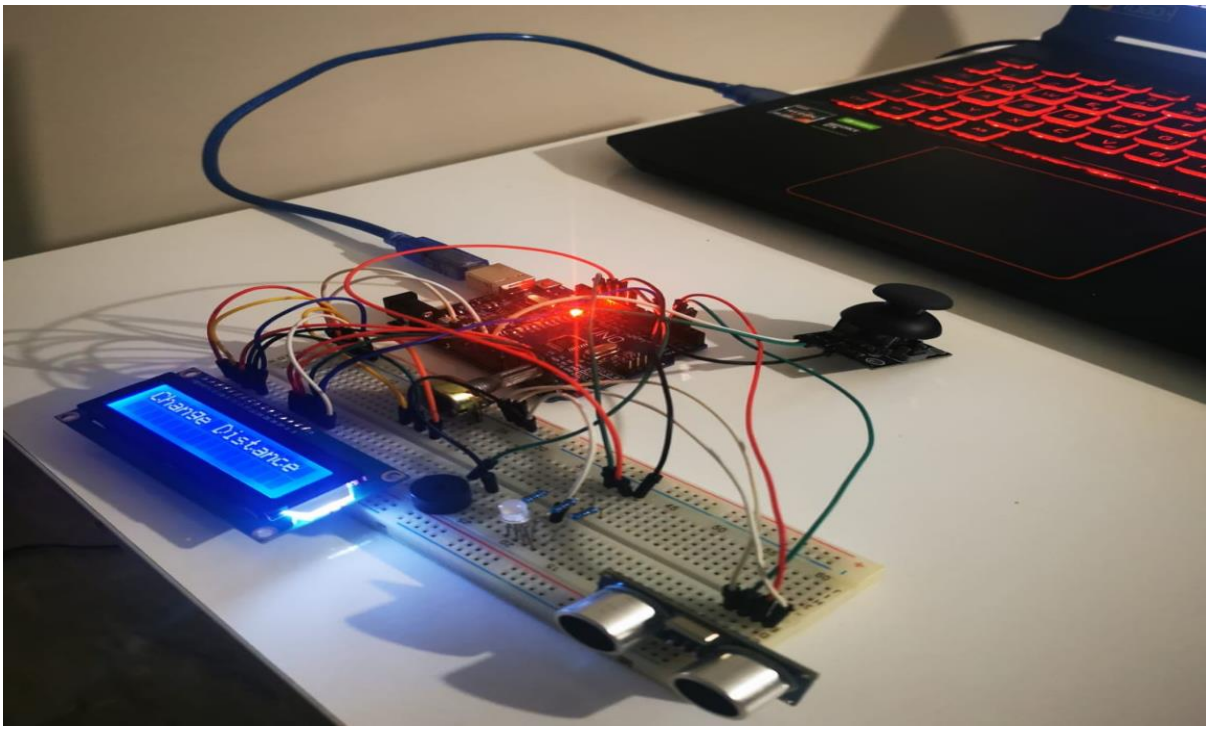
```
    }
}

void ultrasonicSensorReading(){
    unsigned long currentMicros = micros();
    if(triggerState == LOW && currentMicros - previousMicros >= offTime)
     {
       triggerState = HIGH;
       previousMicros = currentMicros;
       digitalWrite(triggerPin, triggerState);
     }
     else if(triggerState == HIGH && currentMicros - previousMicros >=
onTime)
     {
       triggerState = LOW;
       previousMicros = currentMicros;
       digitalWrite(triggerPin, triggerState);
     }
    duration = myPulseIn(echoPin,HIGH,1000);
    distance = ((duration*0.034)/2);
    if(distance < optionDistance && distance != 0){
      turnOnLED(RGBstate);
      analogWrite(buzzerPin,buzzerVolume);
    }
    else{
      turnOffLED();
      analogWrite(buzzerPin,0);
    }
}


unsigned long myPulseIn(int pin, int value, int timeout) {
  unsigned long currentMicros = micros();
  while(digitalRead(pin) == value) {
    if (micros() - currentMicros > (timeout*1000)) {
      return 0;
      }
  }
  currentMicros = micros();
  while (digitalRead(pin) != value) {
    if (micros() - currentMicros > (timeout*1000)) {
      return 0;
    }
  }
  currentMicros = micros();
  while (digitalRead(pin) == value) {
    if (micros() - currentMicros > (timeout*1000)) {
      return 0;
    }
  }
  return micros()-currentMicros;
}


void turnOnLED(String color){
  if(color=="Red"){
     digitalWrite(RGBRedPin,HIGH);
     digitalWrite(RGBGreenPin,LOW);
     digitalWrite(RGBBluePin,LOW);
  }
```

```
  else if(color=="Green"){
    digitalWrite(RGBGreenPin,HIGH);
    digitalWrite(RGBBluePin,LOW);
    digitalWrite(RGBRedPin,LOW);
  }
  else{
    digitalWrite(RGBBluePin,HIGH);
    digitalWrite(RGBRedPin,LOW);
    digitalWrite(RGBGreenPin,LOW);
  }
}

void turnOffLED(){
  digitalWrite(RGBRedPin,LOW);
  digitalWrite(RGBGreenPin,LOW);
  digitalWrite(RGBBluePin,LOW);
}


void displayRGBColor(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Color: ");
  lcd.print(RGBstate);
}

void displayDistance(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Distance: ");
  lcd.print(optionDistance);
  lcd.print("cm");
}

void displayVolume(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Volume: ");
  if(buzzerVolume==0){
    lcd.print("MUTE");
  }
  else if(buzzerVolume==250){
    lcd.print("MAX");
  }
  else{
    lcd.print(buzzerVolume);
  }
}

void displayBrightness(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Brightness: ");
  if(optionBrightness==25){
    lcd.print("MIN");
  }
  else if(optionBrightness==250){
    lcd.print("MAX");
  }
else{ lcd.print(optionBrightness); }}
```

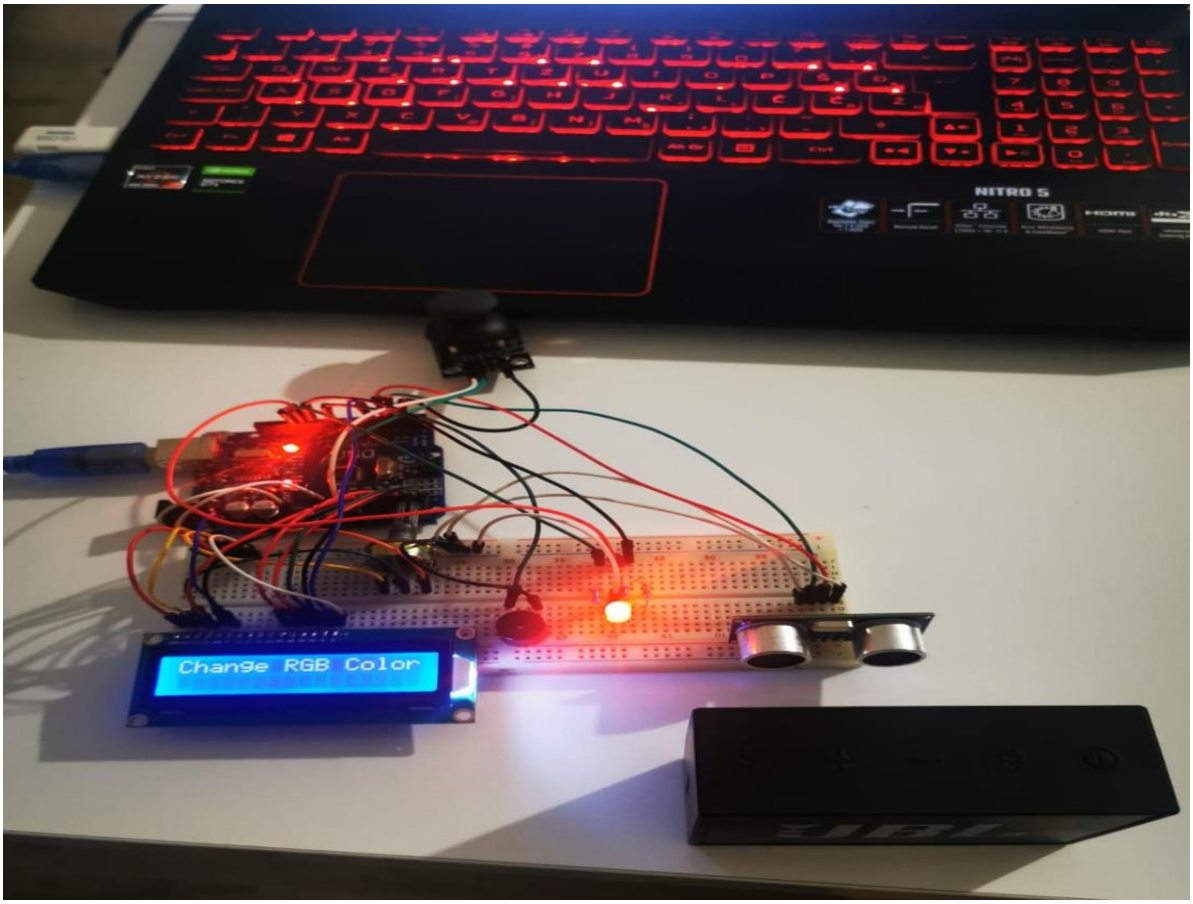
P2: Fotografije gotovog sustava



P 2.1. Sklop u načinu rada namještanja za udaljenost.



P 2.2. Sklop u načinu rada glavnog izbornika za udaljenost.



P 2.3. Reakcija sklopa na objekt u blizini senzora.