ŠOLA N. H. MAKSA PEČARJA

Črnuška cesta 9, 1231 Ljubljana Črnuče

# Senzor zvoka in različni primeri uporabe modula ESP8266

Raziskovalna naloga s področja tehnike

Avtor: Matic Potočnik, 9. d

Mentor: Milan Gaberšek

Ljubljana, marec 2020

## *Zahvala*

*Za navdih, usmerjanje in podporo pri izdelavi raziskovalne naloge se iskreno zahvaljujem svojemu mentorju Milanu Gaberšku. Hvaležen sem mu, ker me je v treh letih svojega poučevanja tehnike znal navdušiti nad elektroniko in me iz popolnega začetnika pripeljal do stopnje, ko sem bil sposoben izdelati raziskovalno nalogo.*

*Iskreno se zahvaljujem za podporo tudi staršem, predvsem pa očetu, ki mi je ob zapletih vedno znal pravilno svetovati.*

*Najlepše se zahvaljujem tudi gospe knjižničarki Mateji Miljković, ki si je vzela čas in mi lektorirala besedilo raziskovalne naloge.*

*Nenazadnje se zahvaljujem tudi vsem učencem in učiteljem, ki so si vzeli čas in rešili mojo anketo.*

# Povzetek

Namen raziskovalne naloge je bil ugotoviti možnosti uporabe senzorja zvoka v kombinaciji s krmilnikom ESP8266, ki ima možnost WiFi povezave. V nalogi je podrobneje prikazana izvedba senzorja zvoka s pomočjo operacijskega ojačevalnika in elektret mikrofona. V nadaljevanju je prikazana rešitev nadzora pozicije živali z uporabo mreže senzorjev zvoka in krmilnikom, ki prek WiFi pošilja podatke v medmrežje in se povezuje z IOT platformo Blynk. Podatke o položaju živali lahko pregledujemo na Android napravi, v aplikaciji Blynk, ki jo lahko izdelamo sami. Prikazane so tudi druge možnosti uporabe krmilnika in senzorja zvoka, na primer krmiljenje peči za centralno kurjavo prek telefona ter odpiranje garažnih vrat prek telefona. Na koncu naloge so prikazani tudi rezultati kratke ankete, ki sem jo izvedel med učenci in učitelji svoje šole. Z njo sem si želel razjasniti mnenje drugih o uporabi telefona za nadziranje naprav in uporabi zvočnega senzorja za nadzor živali ter njihovo splošno mnenje o nadzoru javnih površin.

**Ključne besede**: senzor zvoka, spremljanje živali, mikrokrmilnik ESP8266, Arduino, Blynk

# Kazalo

# Kazalo slik

# 1 Uvod

Lansko poletje sem se udeležil Poletnega tabora inovativnih tehnologij, ki je potekal konec avgusta na Fakulteti za elektrotehniko v Ljubljani. Tam sem obiskoval delavnico »Svetloba z WiFi«, kjer smo izdelali LED svetilko, ki smo jo krmilili s telefonom prek WiFi z Aplikacijo Blynk. Spoznal sem se z IOT ploščico ESP2866, ki je kompatibilna z Arduino razvojnim okoljem ter ima WiFi modul. Ker sem v preteklem šolskem letu že naredil detektor intenzivnosti hrupa, ki je imel analogen prikaz z LED diodami na tiskanem vezju, sem prišel na idejo, da je ta modul ravno prava stvar, da svoje delo nadgradim in povežem s telefonom oziroma tabličnim računalnikom, ki ima večji prikazovalnik (displej). Ko sem začel delati na tem, sem prišel še na mnogo idej, kako lahko modul ESP8266 koristno uporabimo za upravljanje in nadzor v vsakdanjem življenju. V raziskovalni nalogi bom opisal nekaj primerov uporabe, ki sem jih že naredil oziroma jih še nameravam narediti.

## 1.1 Hipoteze

**1.HIPOTEZA**: Obstaja drug način nadzora površin, ki je alternativen video nadzoru.

**2. HIPOTEZA**: Omenjene rešitve bi lahko služile tudi drugim namenom.

**3. HIPOTEZA**: Omenjena rešitev bi bila zanimiva in sprejemljiva za sovrstnike.

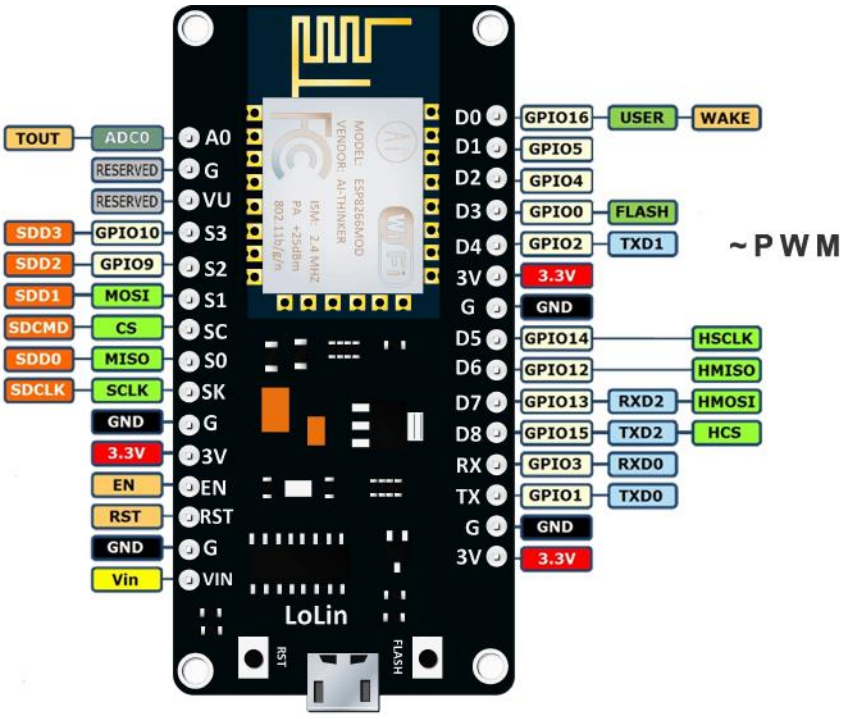
# 2 Predstavitev materialne in programske opreme, teoretična izhodišča

## 2.1 Krmilnik ESP8266 NodeMCU

NodeMCU je razvojna odprtokodna procesorska ploščica, ki temelji na krmilniku ESP8266 in ima veliko splošno uporabnih vhodno-izhodnih vrat (General Purpose IO Ports GPIO), en analogni vhod, WiFi modul ter USB serijski vmesnik [1]. Fizični izgled je prikazan na sliki (Slika 1).

*Slika 1 - NodeMCU*

Na naslednji sliki so prikazani izhodi omenjene ploščice.



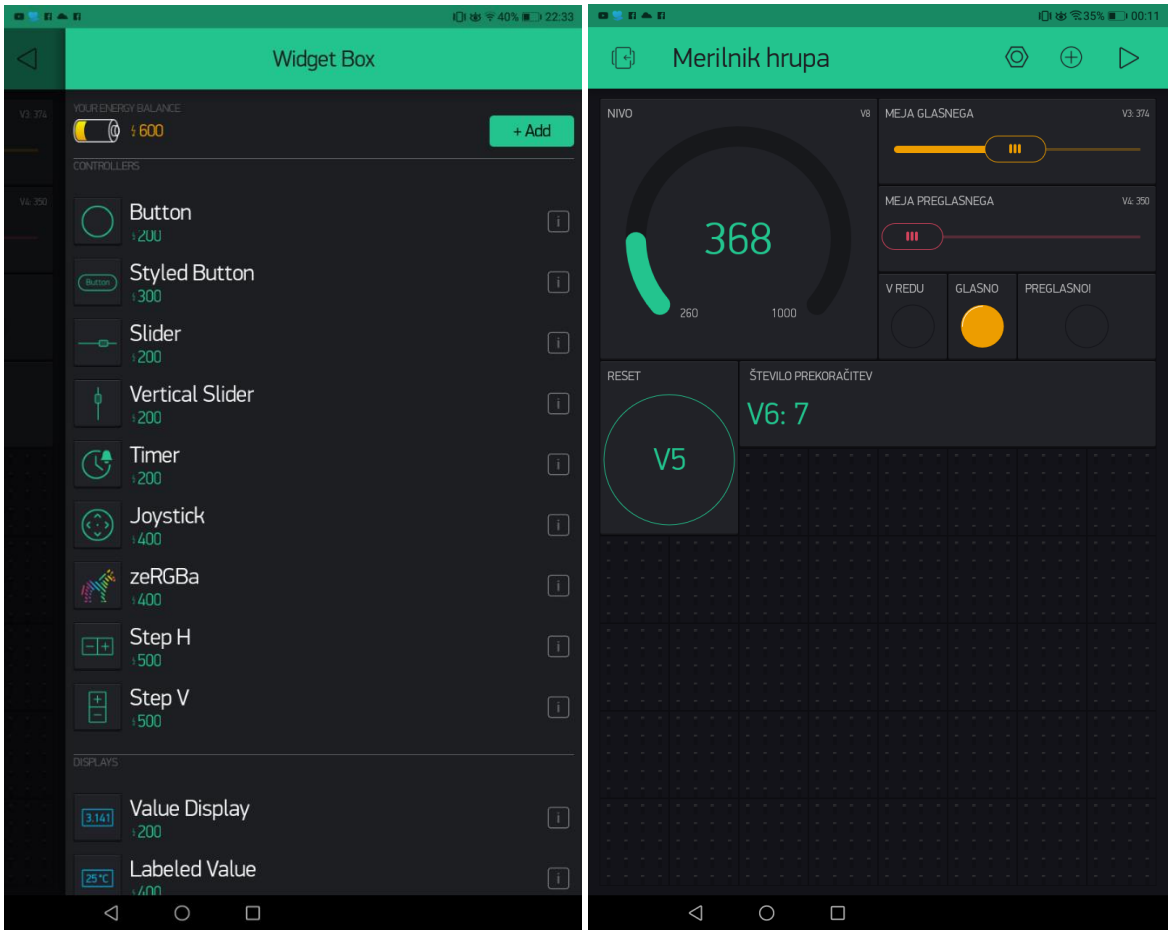

*Slika 2 - NodeMCU priključek (pinout)*

Iz slike vidimo (Slika 2), da ima dotična ploščica 9 digitalnih vrat D0-D8, ki jih lahko uporabimo kot vhode ali izhode, en analogni vhod A0, serijski vmesnik, ki je povezan prek mikro USB priključka ter možnost 5 V ali 3,3 V napajanja. V bistvu je modul 3,3 V naprava, ki ima svoj regulator napetosti iz 5 V na 3,3 V. Zaradi tega tudi vsi vhodi in izhodi delujejo na 3,3 V nivojih. Ploščica ima 32-bitni procesor in 1 MB pomnilnika in se jo lahko programira z Arduino razvojnim okoljem.

## 2.2 Aplikacija Blynk

Blynk je IOT platforma, ki ponuja Android aplikacijo, s pomočjo katere si zelo enostavno v nekaj minutah naredimo grafični vmesnik, s katerim potem krmilimo svojo napravo [2]. Brezplačno je mogoče uporabljati omejeno število grafičnih elementov. Ob obvezni registraciji dobi vsak uporabnik 2500 energijskih točk, ki jih lahko potem poljubno uporablja za želene grafične elemente. Pri tem ima vsak element svojo ceno, kot je razvidno na sliki (Slika 3). Za enostavno aplikacijo potrebujemo le nekaj gumbov, drsnikov in LED indikatorjev, za kar 2500 točk zadostuje in še nekaj malega ostane.

Aplikacija deluje tako, da vsakemu grafičnemu elementu ustreza nek fizični vhod ali izhod na ploščici, lahko pa takemu elementu priredimo tudi virtualna vrata, ki jih potem obdelujemo v aplikaciji na sami ploščici.




*Slika 3 - Blynk Android aplikacija*

Aplikacijo povežemo s ploščico tako, da na ploščico iz Arduino okolja zapečemo že pripravljen primer aplikacije, ki mu je potrebno nastaviti nekaj parametrov, in sicer:

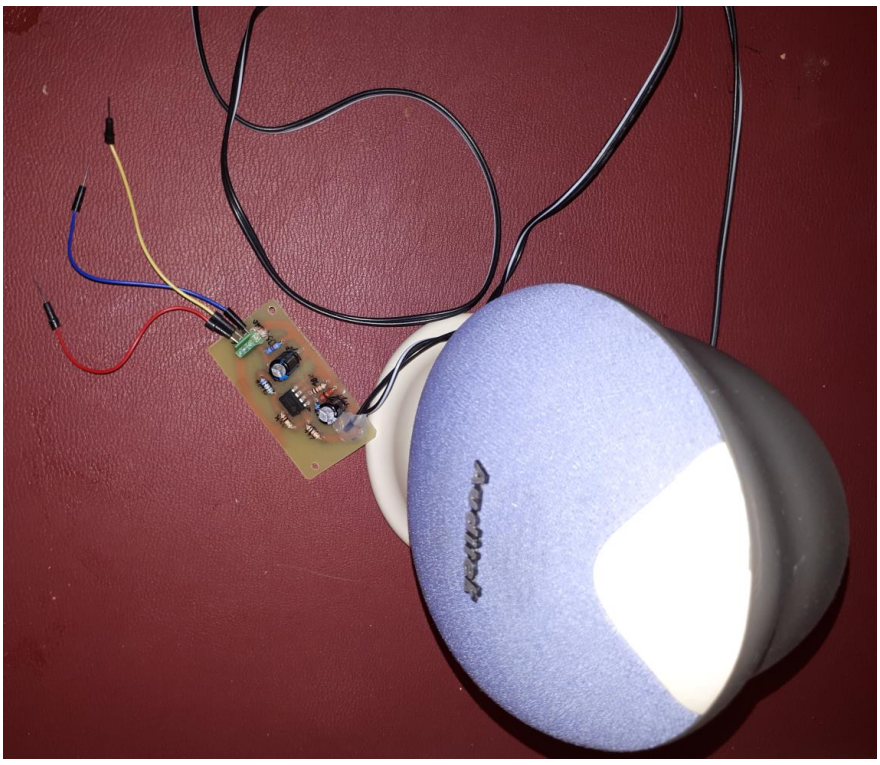
- ime WiFI omrežja,
- geslo za WiFi omrežje,
- kodo, ki nam jo zgenerira Blynk aplikacija.

Ko ploščo resetiramo, se le-ta poveže z WiFi omrežjem in nato prek interneta z Blynk strežnikom, ki se mu predstavi s kodo aplikacije. Če je vse v redu, v aplikaciji vidimo, da se je naša naprava povezala in komunikacija že deluje v obe smeri. Ploščica pošilja podatke aplikaciji, aplikacija pa na našo zahtevo pošilja ukaze ploščici. V mojem primeru sem med drugim imel števec, ki je prikazoval vrednost analognega vhoda na ploščici.

# 3 Merilnik hrupa - eksperimentalni del

## 3.1 Senzor hrupa

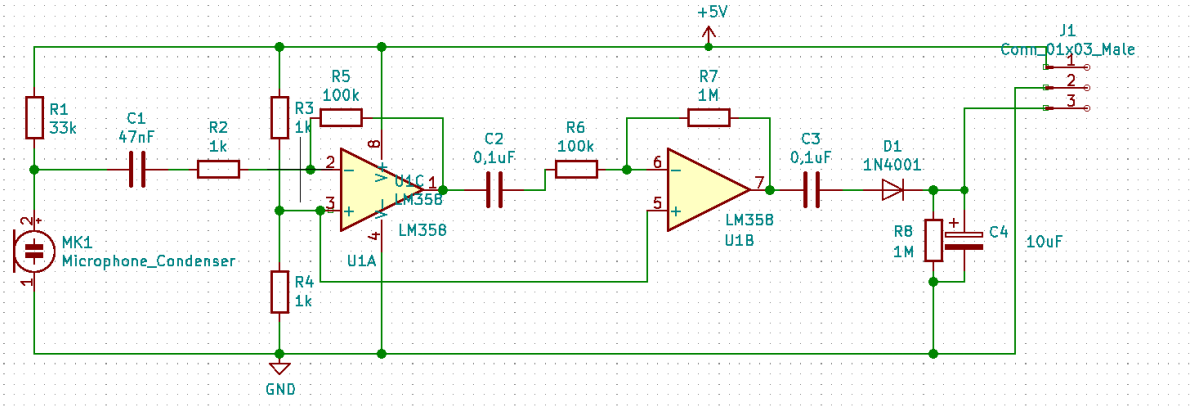
Kot sem že v uvodu omenil, sem že lansko leto izdelal senzor jakosti hrupa. S pomočjo operacijskega ojačevalnika sem izdelal vezje, ki meri nivo hrupa in na izhodu generira sorazmerni napetostni nivo.



*Slika 4 - Senzor hrupa*

Za zaznavo hrupa sem lansko leto uporabil kar zvočnik, ki sem ga uporabil kot mikrofon, saj nisem imel pravega mikrofona. Zvočnik se obnaša podobno kot dinamični mikrofon in ko zvok zatrese membrano in tuljavo na njej, ta generira na sponkah majhno napetost, ki jo ojača operacijski ojačevalnik [3]. Nato signal usmerim z diodo in z njim polnim kondenzator C4. Napetost na kondenzatorju je tako sorazmerna jakosti hrupa. Vezje (Slika 4) potrebuje napajanje 5 V (rdeča), maso GND (modra) in daje napetost na izhodu (rumena).

Ker je zvočnik velik in ni najbolj občutljiv, sem letos spremenil vezje in uporabil pravi elektret mikrofon. Ta potrebuje napajanje, zato sem moral vezje malenkost spremeniti. To je prikazano na stikalnem načrtu (Slika 5).



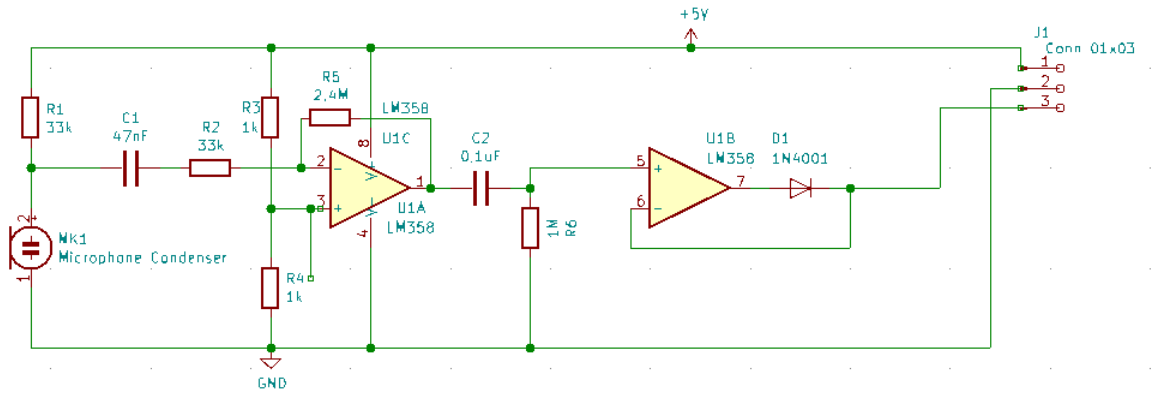

*Slika 5 - Senzor hrupa, stikalni načrt*

### 3.1.1 Optimizirano vezje

Ker sem imel z obstoječim vezjem ob menjavi zvočnika s kondenzatorskim mikrofonom kar precej težav, sem se odločil nekoliko spremeniti načrt vezja. Pri menjavi mikrofona se je pojavila težava, da sem na izhodu vezja dobil zelo majhen signal tudi ob zelo velikem hrupu. S poskušanjem sem ugotavljal, kako velikost upora R1 vpliva na občutljivost mikrofona, in prišel do ugotovitve, da nizke vrednosti upora povzročijo, da mikrofon ne zajema zvoka iz okolice, ampak samo iz neposredne bližine. S povečanjem upornosti R1 je mikrofon začel zajemati tudi oddaljene zvoke, čeprav je bil signal na mikrofonu zato manjši. Zato sem se moral posvetiti tudi razmerju uporov R2 in R5, ki določata ojačanje prve ojačevalne stopnje U1A čipa LM358 [3]. Izkazalo se je, da kljub temu, da sem imel razmerje 100, na izhodu nisem dobil zadosti velikega signala. Ugotovil sem, da ima velik vpliv tudi vhodna upornost prve stopnje, ki jo določa upor R2. Ko sem v prvotnem vezju uporabljal namesto mikrofona zvočnik, ki ima zelo nizko notranjo upornost, je bil kiloohmski upor v redu, za kondenzatorski mikrofon pa to ni bilo več dobro, ker ima kondenzatorski mikrofon veliko večjo notranjo upornost. Zato sem povečal upor R2 na 33 kiloohmov. Ker se je zaradi tega preveč zmanjšalo ojačanje, sem moral povečati tudi upor R5. Vzel sem največji upor, kar sem jih imel, 2,4 megaohma, in rezultati so bili veliko boljši, kljub temu, da se je celotno ojačanje zmanjšalo iz 100 na 72.

Ker sem na izhodu vezja pri največjem hrupu dobil komaj 1,5 V signala, sem želel izboljšati še to, saj analogni vhod na modulu ESP8266 za najvišjo vrednost potrebuje okrog 3,3 V. Z

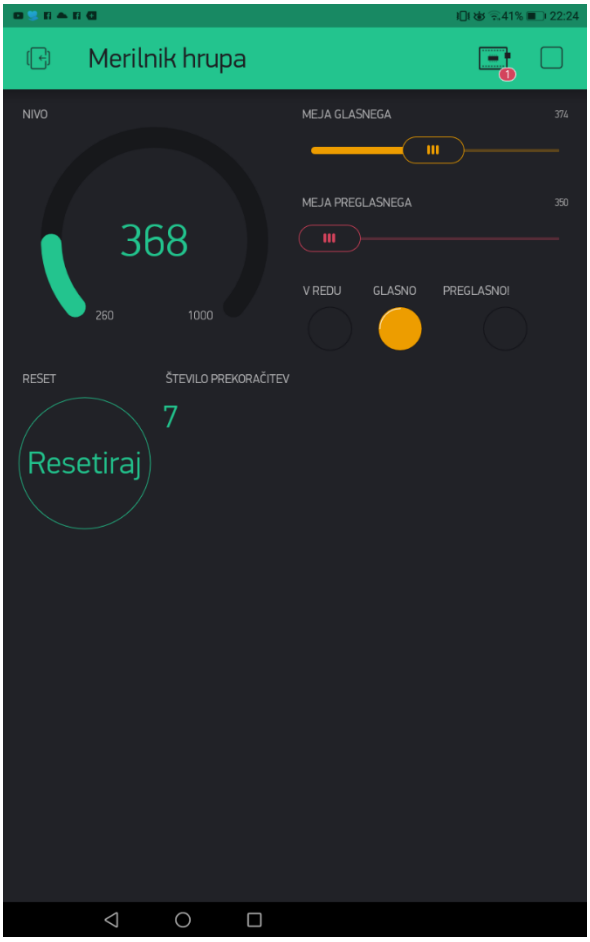
analizo vezja sem ugotovil, da mi usmerniška dioda na koncu pobere 0,7 V. Na internetu nem našel vezje za »idealno« diodo. Ko v povratno vezavo operacijskega ojačevalnika vstavimo diodo, se izhod operacijskega ojačevalnika postavi v stanje, kot bi bila kolenska napetost diode praktično 0 V (Slika 6). Takšna dioda sicer ne prenese velikih tokov, vendar je to dovolj za moj senzor, ko potrebujem le izhodno napetost. Ker ima čip LM358 dva operacijska ojačevalnika, sem drugega namesto druge ojačevalne stopnje uporabil raje za posnemanje delovanja idealne diode [4]. Na ta način sem na izhodu dobil signal okrog 2,3 V, kar je bistveno več kot prej, pri tem pa se je vezje še poenostavilo. V vezju je pomemben tudi upor R6, ki predstavlja breme in določa izhodno napetost. Če tega upora ni, se kondenzator C2 nabije in tok preneha teči, posledično na izhodu ne dobimo nobene odzivnosti. Upora R3 in R4 predstavljata napetostni delilnik tako, da ustvarita navidezno maso za operacijski ojačevalnik. To pomeni, da vhodno napetost 5 V razdelita na 2 krat 2,5 V, kar je podobno, kot če bi operacijski ojačevalnik simetrično napajali s +/- 2,5 V. Kondenzator C1 je vezni kondenzator, ki zagotovi, da iz mikrofona v ojačevalnik teče le izmenični signal. Enako funkcijo ima tudi izhodni vezni kondenzator C2.




*Slika 6 - Senzor hrupa, optimiziran stikalni načrt*

## 3.2 Programski del merilnika NodeMCU

Senzor hrupa priključimo na analogni vhod krmilnika in v zanki beremo napetostni nivo. Nato z enostavno primerjavo z referenčnima vrednostma za glasno in preglasno prižigamo ustrezne izhode. Če je vrednost signala nižja od nastavljene glasnosti, vklopimo izhod za zeleno LED in izklopimo izhoda za rumeno in rdečo LED. Če nivo signala preseže vrednost za glasno in je nižji od preglasnega, vklopimo izhod za rumeno LED ter izklopimo zeleno in rdečo. Ko nivo signala preseže mejo preglasnega, vklopimo izhod za rdečo LED ter izklopimo zeleno in rumeno. Procesor izvaja meritve v zanki večkrat na sekundo in stanja izhodov pošilja prek WiFi na Blynk strežnik. Aplikacija na telefonu je povezana s strežnikom in prikazuje izmerjene vrednosti, pri čemer prižiga ustrezne lučke (Slika 7).



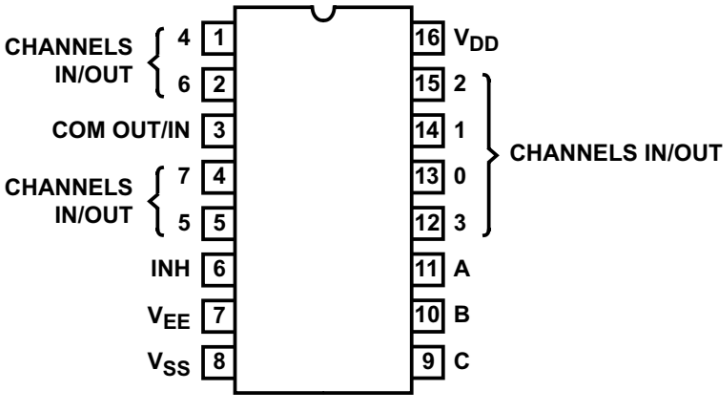

*Slika 7 - Aplikacija na telefonu za merjenje hrupa*

# 4 Merilnik hrupa kot senzor za zaznavanja lokacije živali

Ko želimo ugotoviti lokacije živali, lahko to naredimo tako, da postavimo več kamer in spremljamo, kje se živali nahajajo. V splošnem so kamere relativno drage in če želimo sistem avtomatizirati, moramo narediti tudi prepoznavanje slike, kar v splošnem zahteva kompleksno programsko opremo, ki ni poceni.

V našem primeru bomo zaznavo pozicije živali naredili s pomočjo senzorjev zvoka tako, da jih bomo razporedili večje število na znane lokacije in potem spremljali nivo hrupa na posameznem senzorju. Lahko trdimo, da se žival nahaja v bližini tistega senzorja, na katerem izmerimo največji nivo hrupa. Ker poznamo položaj vsakega senzorja, lahko ob vsakem času glede na nivo hrupa na ta način določimo položaj živali. Če žival miruje in ne oddaja zvoka, je senzorji ne zaznajo, zato za njeno lokacijo privzamemo zadnjo poznano lokacijo. Tak sistem je v osnovi veliko preprostejši, cenejši in ne zahteva kompleksne programske opreme. Ker senzor hrupa na izhodu daje napetostni signal, ki je sorazmeren nivoju hrupa, ga moramo priklopiti na analogni vhod krmilnika NodeMCU. Iz slike krmilnika vidimo (Slika 2), da ima

omenjen krmilnik le en analogni vhod, kar bi pomenilo, da nanj lahko priključimo le en senzor. Tak način priključevanja bi bil zelo potraten in neučinkovit. V praksi krmilnik odčita vrednost senzorja nekajkrat na sekundo ali celo manj pogosto, preostali čas pa lahko dela kaj drugega, zato lahko preko dodatnega vezja na krmilnik priključimo več senzorjev in jih sekvenčno beremo. V delčku časa izberemo en senzor, preberemo njegovo vrednost, v naslednjem trenutku preberemo vrednost naslednjega senzorja in to delamo toliko časa, da preberemo vrednosti na vseh senzorjih, nato vse skupaj ponovimo. Ker branje enega senzorja traja le nekaj ms, lahko v eni sekundi preberemo veliko senzorjev. Naprava, ki nam omogoča takšno časovno izbiro senzorja, se imenuje multiplekser. Takšno vezje že obstaja v obliki integriranega vezja CD4051 [5], ki je prikazano na spodnji sliki (Slika 8).




*Slika 8 - Multiplekser CD4051*

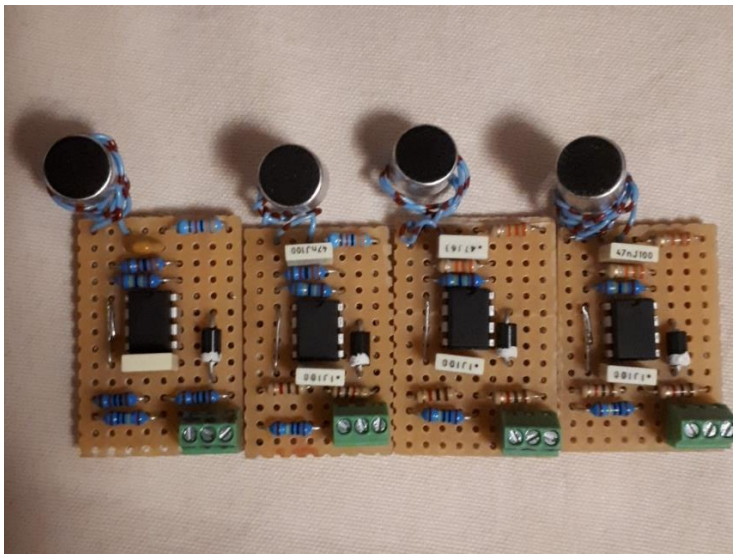
Ta multiplekser omogoča preslikavo 8 vhodov/izhodov na en skupni izhod/vhod. Z ustrezno binarno kombinacijo, ki jo nastavimo na pinih A, B in C, izberemo ustrezen kanal, ki ga preslikamo na skupni vhod na pinu 3. Če vrednosti ABC nastavimo na logično vrednost 000, je izbran kanal 0 – to pomeni, da se pin 13 električno preklopi na pin 3. Če izberemo kombinacijo 001, preklopimo na pin 3 kanal 1 – torej pin 14. Pine A, B in C krmilimo z digitalnimi GPIO pini krmilnika NodeMCU, ki jih imamo 10 (D0-D9), kar zadostuje.

To pomeni, da s pomočjo takega vezja lahko na krmilnik priključimo 8 senzorjev hrupa. Če bi želeli priključiti še več senzorjev, bi morali v kaskado povezati več multiplekserjev. Z 9 čipi bi lahko razširili število vhodov na 64, za izbiro senzorja pa bi potrebovali 6 GPIO pinov (2^6 = 64), ki so prav tako na voljo. Takšno število senzorjev lahko na terenu pokrije že precej velik prostor. Na primer, če jih razporedimo v kvadratno mrežo po 8 krat 8 senzorjev in jih postavimo na 5 metrov, z njimi pokrijemo površino 1600 m2 ali 16 arov, kar je v največ primerih dovolj.

## 4.1 Praktična izvedba

V mojem primeru sem se odločil narediti vezje s 5 senzorji, ki v praksi pokaže princip delovanja, hkrati pa tudi omogoča priklop vseh senzorjev in vstavitev naprave v primerno ohišje. Odločil sem se, da senzorje izdelam po shemi, prikazani na stikalnem načrtu (Slika 6)

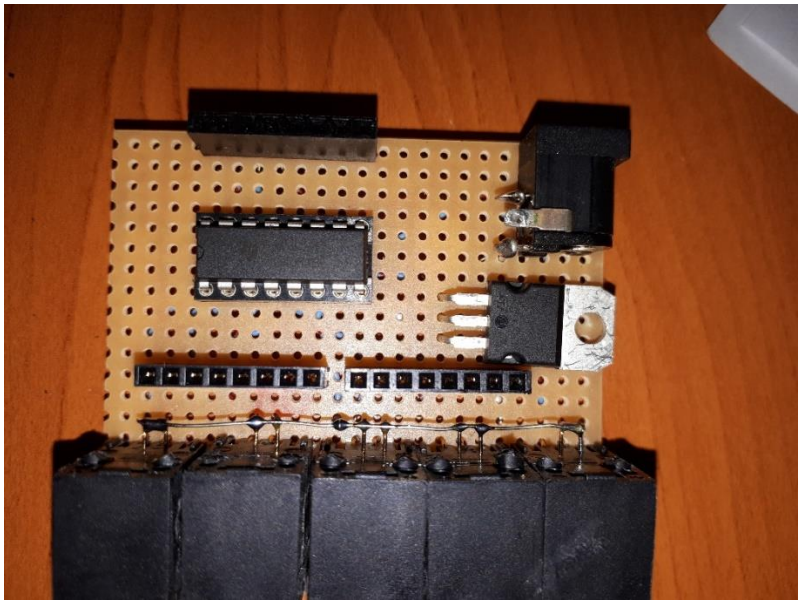
na proto ploščici (Slika 9), ker sem se s tem izognil izdelavi tiskane ploščice. Tak način dela žal ne omogoča izdelave večjega števila senzorjev, ker je vsaka ploščica unikat, zato sem število senzorjev omejil na 5. Slika 9 prikazuje štiri take senzorje.



*Slika 9 - Izdelani senzorji hrupa*

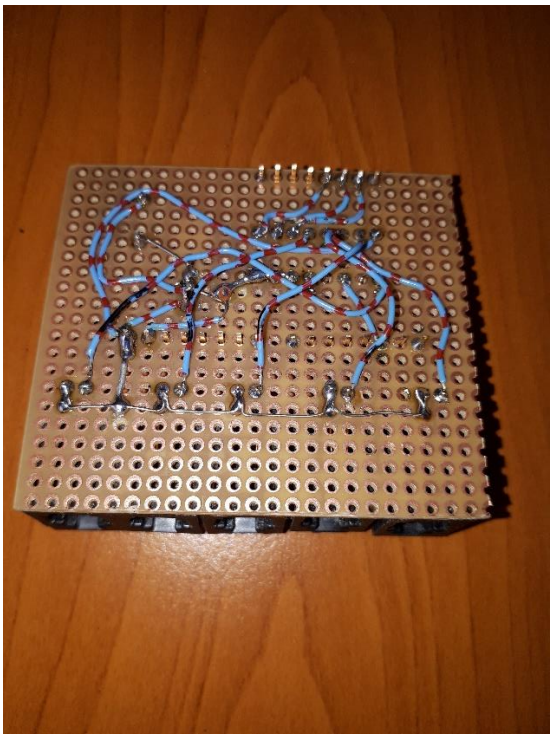
### 4.1.1 Izdelava krmilnega dela

Krmilni del ne zahteva veliko elementov, saj potrebujemo le krmilnik NodeMCU, multiplekser CD4051 ter napajanje 5 V. Ker sem za napajanje uporabil 12 V napajalnik, ki sem ga imel doma, sem dodal še regulator napetosti za 5 V [6]. Vse to sem prav tako izdelal na proto ploščici, kjer je bilo le potrebno povezati ustrezne vhode in izhode (Slika 10 in Slika 11). Za priključitev senzorjev pa sem se odločil uporabiti RJ9 konektorje, ki se uporabljajo za priključevanje telefonskih slušalk in imajo 4 priključke (od teh sem uporabil le 3).



*Slika 10 - Vezje krmilnega dela zgoraj*

Ker se je regulator napetosti nekoliko pregreval, sem mu moral dodati še hladilni element, ki sem ga izdelal iz aluminijaste pločevine (Slika 12).



*Slika 11 - Plošča krmilnega dela spodaj*

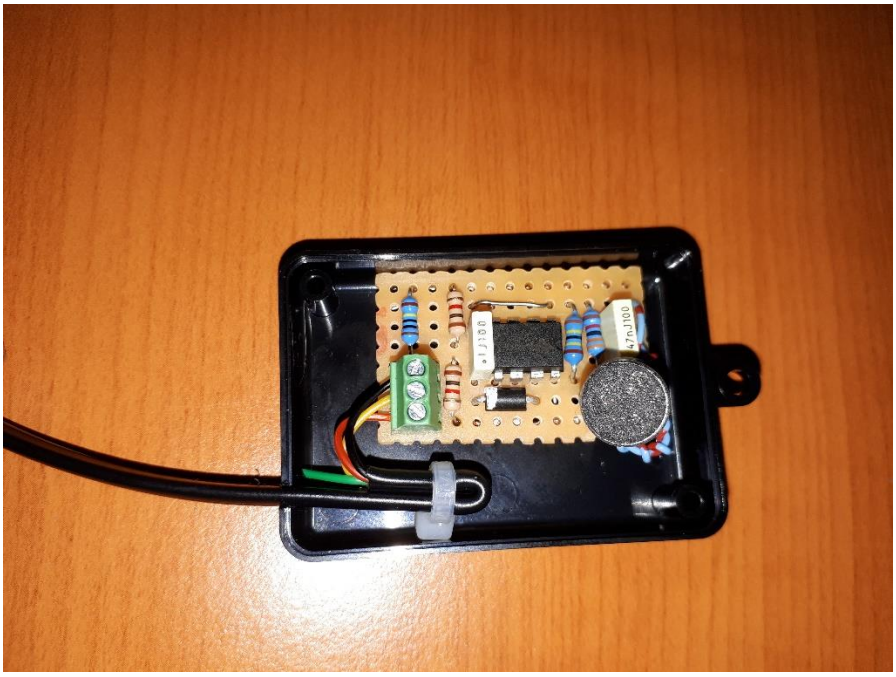
*Slika 12 - Plošča krmilnega dela z natičnim krmilnikom NodeMCU*

Vse skupaj sem zapakiral v primerno ohišje, ki sem ga kupil v trgovini Conrad. V ohišju je bilo potrebno izrezati odprtine za konektorje senzorjev in napajalni priključek. Vse to je prikazano na sliki krmilnega dela v ohišju (Slika 13). Več o tem bom opisal v naslednjem poglavju.

*Slika 13 - Krmilni del v ohišju*

### 4.1.2 Izvedba ohišja senzorja

Podobno kot samo krmilno vezje je bilo potrebno tudi vse senzorje zapakirati v ustrezno ohišje (Slika 14), saj bi se sicer lahko hitro zgodilo, da bi se od senzorja odlomila kakšna priključna žička. Za ohišje senzorja sem izbral manjšo škatlico, v katero sem moral narediti luknjico za mikrofon in priključni kabel. Pokrov ohišja senzorja je privit z dvema vijakoma tako, da je celotno ohišje tudi mehansko odpornejše. Končna podoba s priključnim kablom je prikazana na sliki senzorja s priključnimi kabli (Slika 15).



*Slika 14 - Ohišje senzorja*

*Slika 15 - Senzor v ohišju s priključnim kablom in konektorjem*

## 4.2 Programski del

Krmilnik NodeMCU lahko programiramo v razvojnem okolju (IDE) Arduino [7] v programskem jeziku C/C++. Za osnovo sem vzel primer Blynk iz Arduinove knjižnice in ga razširil [8]. Princip delovanja programa je tak, da program v glavni zanki funkcije loop() izvaja zanko for, ki za posamezni senzor nastavi izhodne pine D[1-3] na binarno vrednost številke senzorja ter ga s tem izbere. Nato prebere vrednost analognega vhoda A0, ki v tistem trenutku ustreza signalu na izbranem senzorju. Rezultate potem v glavni zanki povprečim tako, da preberem za vsak senzor določeno število vzorcev. S tem sem se izognil, da bi ena meritev lahko preveč vplivala na rezultat. Ko dobim izračunano povprečje za vsak senzor, se na podlagi kriterijske funkcije določi, ali se bo »LED« indikator v Blynk aplikaciji prižgal ali ne. Ker »LED« element omogoča tudi izbiro svetlosti indikatorja, sem se odločil, da bom za vsak senzor prikazoval 8 odtenkov svetlosti. V aplikaciji lahko indikatorju določimo 255 odtenkov svetlosti. Vrednost 0 predstavlja ugasnjen »LED«, 255 pa najsvetlejši odtenek na »LED« indikatorju, jaz pa sem se omejil na 8 odtenkov in stanje 0 tako, da je med njimi opazna razlika. Vrednosti, ki jih uporabljam so: 0, 32, 64, 96, 128, 160, 192, 223, 255. Za prižiganje indikatorjev se v aplikaciji Blynk uporablja tako imenovane »virtualne pine«, v mojem primeru V20-V25. Poleg tega sem za prvi senzor v aplikaciji Blynk dodal še kazalčni indikator, ki prikazuje točno vrednost prebranega senzorja. Ta števec prav tako uporablja virtualni pin V8. Da lahko vidim, če aplikacija deluje, v zanki krmilim tudi modro LED diodo na ploščici, ki je vezana na izhod D4. Če se ta nahaja v stanju 0, dioda sveti, če pa je v stanju 1, je ugasnjena. Ko se zanka izvaja, modra dioda utripa in s tem prikazuje normalni način delovanja. V spodnjem okvirju je prikazan bistveni del programske kode.

```
void loop () {

  ...
for (sensor=0; sensor<5; sensor++){
    bool data1, data2, data3;
    data1=sensor&1;
    data2=sensor&2;
    data3=sensor&4;

    digitalWrite(D1, data1);
    digitalWrite(D2, data2);
    digitalWrite(D3, data3);

    delay (5);
    soundValue = analogRead(A0);

    if (n<= N_MEASURES){
       soundAvg[sensor] = (soundValue + (n-1)*soundAvg[sensor])/n ;
    }
    else {
       if (sensor == 0) {
          Blynk.virtualWrite(V8, ((int) soundAvg[sensor]));
       }
       Blynk.virtualWrite(V20+sensor, ledConvert(soundAvg[sensor]));
    }
    if (n > N_MEASURES){
       static bool ledState = true;
       n = 0;
       ledState = !ledState;
       digitalWrite(D4, ledState);
    }

}
```

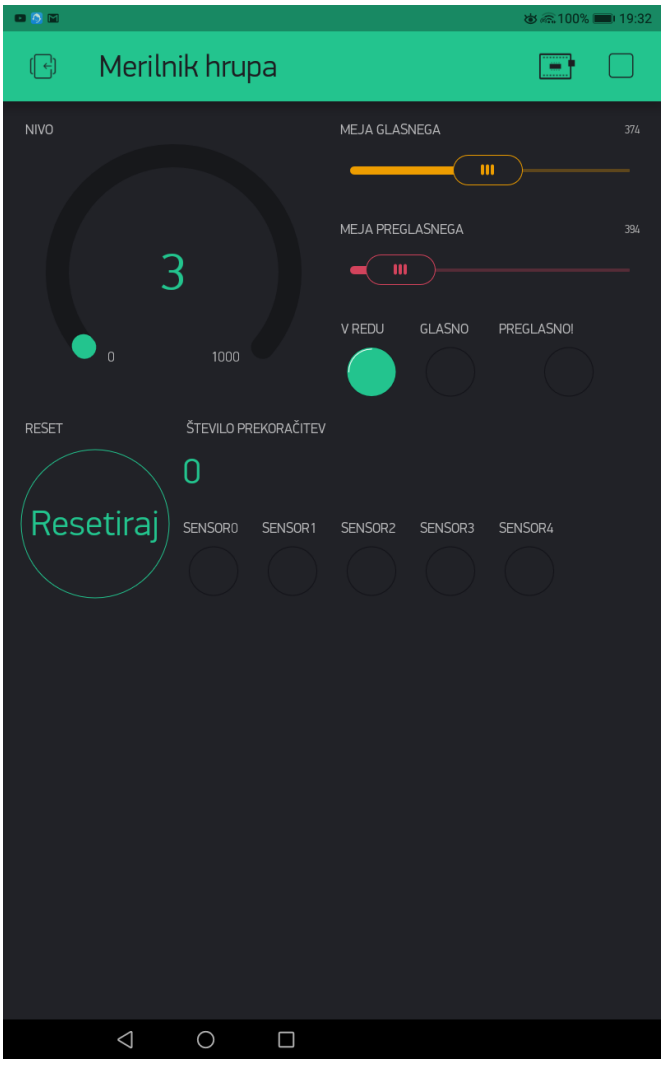
Za preslikavo jakosti zvoka v intenziteto »LED« indikatorja sem naredil funkcijo za pretvorbo ledConvert, ki na osnovi prebrane vrednosti senzorja določi svetlost »LED« indikatorja. Vrednosti sem določil s poskušanjem.

```
int ledConvert(int a)
{
  int rv=0;

  if (a > 350) rv = 255;
  else if (a > 310) rv = 223;
  else if (a > 270) rv = 192;
  else if (a > 230) rv = 160;
  else if (a > 190) rv = 128;
  else if (a > 150) rv = 96;
  else if (a > 110) rv = 64;
  else if (a > 70) rv = 32;
  else rv = 0;

  return rv;
}
```

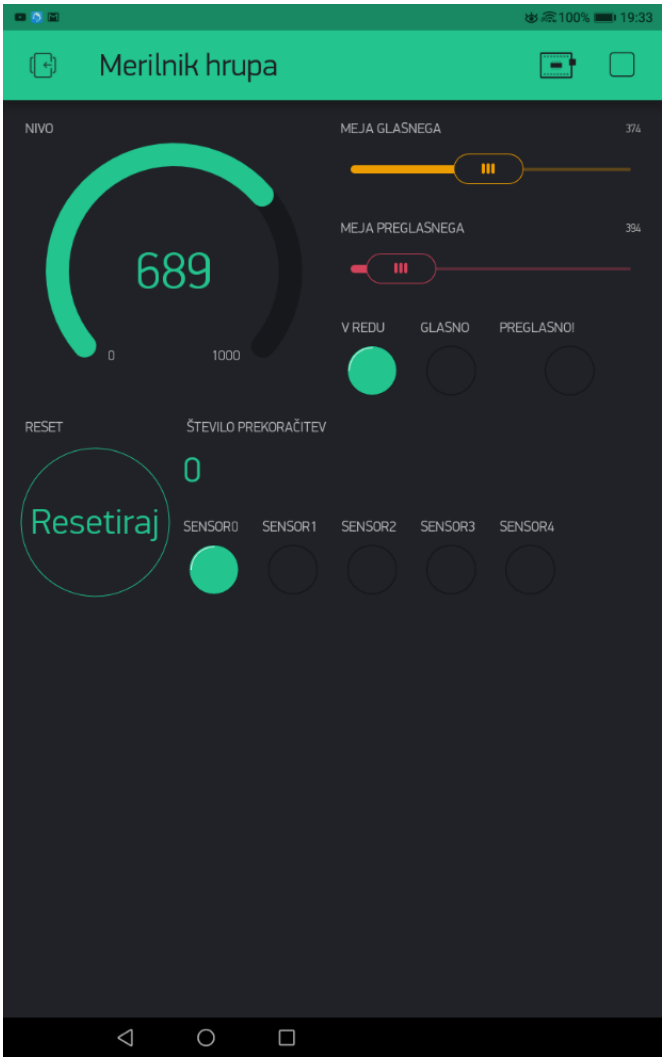
## 4.3 Grafični del aplikacije Blynk

Za nadzor senzorjev sem v Blynk-u naredil aplikacijo, ki za vsak senzor uporablja »LED« indikator, za prvi senzor pa sem dodal še merilni kazalec, ki prikazuje tudi točno vrednost izhoda senzorja. Analogni vhod je priključen na analogno digitalni pretvornik (A/D), ki analogni signal pretvori v vrednosti med 0 in 1024. Vrednost 0 ustreza 0 V na vhodu, vrednost 1024 pa 3,3 V, kar je tudi največja dopustna vrednost. V mojem primeru da senzor na izhodu največjo napetost okrog 2,3 V, kar številčno pomeni vrednosti okrog 700. »LED« indikatorji in merilni kazalec delujejo tako, da berejo tako imenovane »virtualne porte«, katerih vrednosti nastavlja programska koda na krmilniku NodeMCU tako, da kliče funkcijo virtualWrite, ki za parameter potrebuje številko virtualnega porta ter vrednost, ki jo nastavi.

```
Blynk.virtualWrite(Vn,vrednost);
```

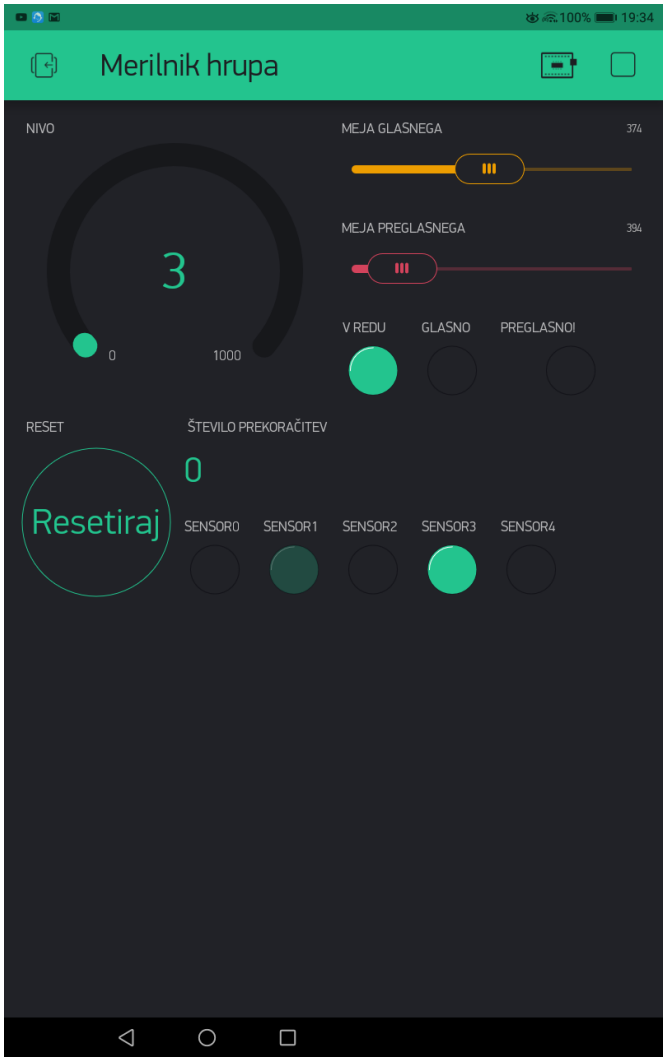



*Slika 16 - Aplikacija Blynk za 5 senzorjev*




*Slika 17 - Aplikacija Blynk, hrup na 1. senzorju*

Ko je aplikacija povezana s krmilnikom, se odziva na jakost zvoka na posameznem senzorju tako, da se spreminja odtenek barve »LED« indikatorja za posamezni senzor. Slika 17 prikazuje stanje, ko je na prvemu senzorju (sensor0) zaznan močan hrup, zato je prvi indikator obarvan v najsvetlejši zeleni barvi. Slika 18 prikazuje stanje, ko je na 4. senzorju (sensor3) glasen zvok, na 2. senzorju (sensor1) pa je zaznan šibkejši zvok, ostali senzorji pa hrupa ne zaznavajo, zato so obarvani črno. Slika 16 prikazuje stanje, ko noben senzor ne zaznava hrupa.




*Slika 18 - Aplikacija Blynk, hrup na 2. in 4. senzorju*

## 4.4 Testiranje naprave in rezultati

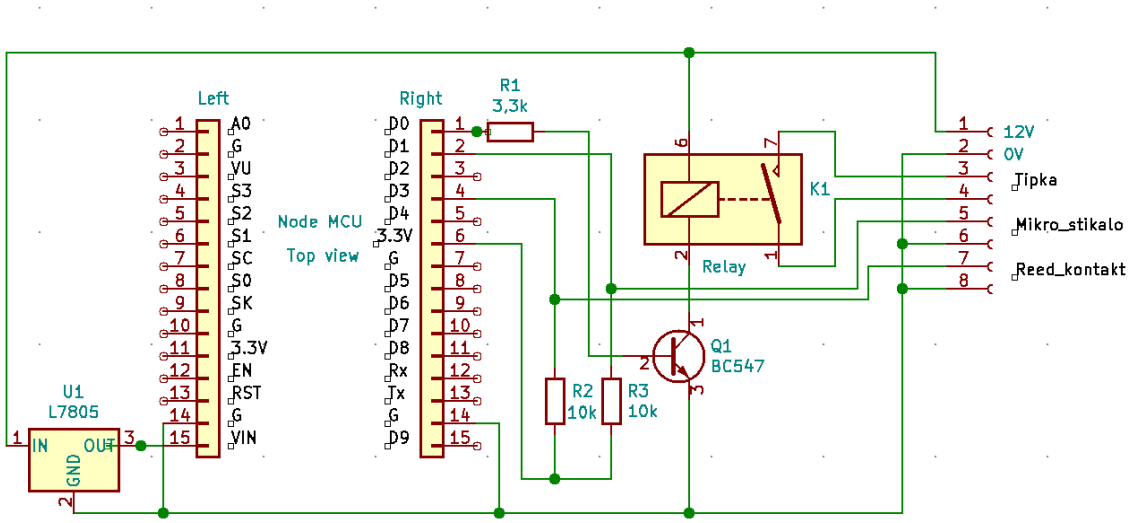
Ker naprave nisem mogel preizkušati v resničnih okoliščinah s pravimi živalmi, sem za ta namen moral narediti izvor zvoka. Za ta namen sem uporabil piezo piskač, ki sem ga preko stikala in upora priklopil na 9 V baterijo. Upor sem uporabil za znižanje glasnosti, saj je piskač zelo glasen. Slika 19 prikazuje testni izvor zvoka, ki nadomešča žival. Za preskušanje sem vključil piskač in ga približal izbranemu senzorju. Pri tem sem opazoval, kako se je na aplikaciji prižgal »LED« indikator z ustrezno intenzivnostjo. Če sem izvor zvoka približal senzorju, je indikator svetil svetleje, z oddaljevanjem pa šibkeje. V primeru, ko se je izvor zvoka nahajal v bližini dveh ali več senzorjev, sta se na aplikaciji prižgala dva »LED« indikatorja. Eden je svetil močneje, drugi pa šibkeje. Tako stanje je prikazano na sliki aplikacije Blynk (Slika 18), ko je izvor zvoka bližje senzorju4 in nekoliko bolj oddaljen od senzorja2.


*Slika 19 - Testni izvor zvoka*

# 5 Uporaba NodeMCU za nadzor in krmiljenje dvižnih garažnih vrat

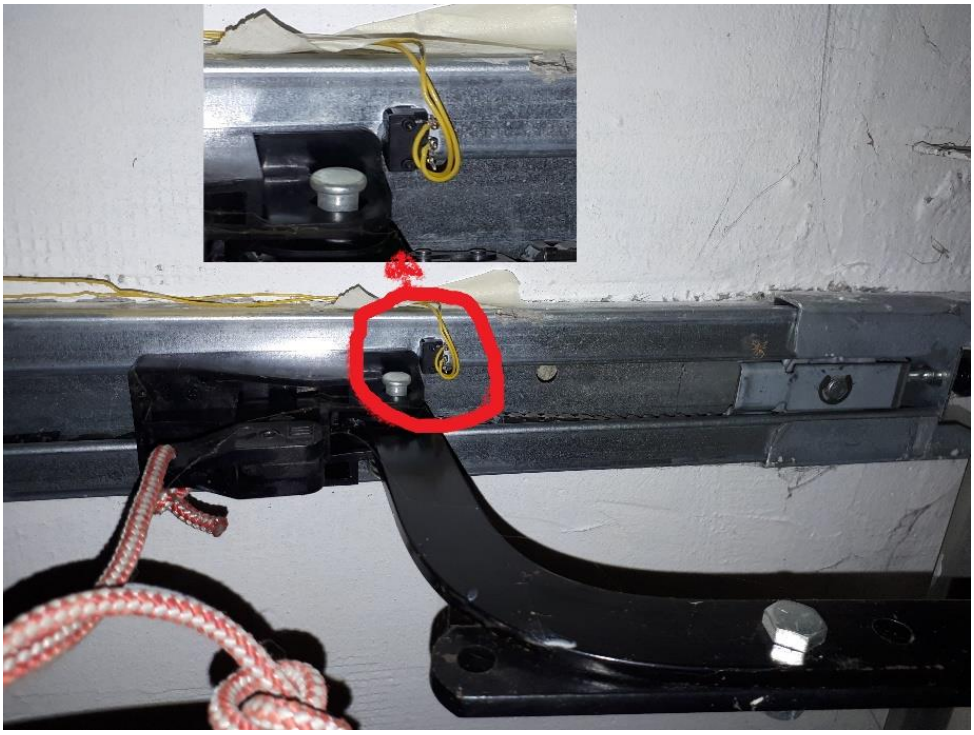
Doma imamo v garaži električna dvižna vrata, ki imajo tudi daljinski upravljalnik, vendar je le en sam, pa še njegov doseg je slab. Znotraj se vrata odpirajo na tipko tako, da se z enim pritiskom odprejo, ko pritisnemo drugič, pa se zaprejo. Ker se vrata zapirajo počasi, lahko med tem časom zapustimo garažo. Vendar s tem obstaja velika možnost, da se zaklenemo ven, še posebej, če pozabimo vzeti ključ ali če ni nikogar doma. Zaradi tega sem si vedno želel vrata odpirati s telefonom, ki je vedno pri roki.

Tako mi je na misel prišla ideja, da bi lahko s krmilnikom NodeMCU in aplikacijo Blynk odpiral in zapiral vrata, hkrati pa bi lahko na daljavo preveril, ali so vrata zaprta ali odprta. Zato sem naredil vezje (Slika 20), kjer sem uporabil izhod D0 za krmiljenje releja, ki ga lahko aktiviramo z aplikacijo Blynk.



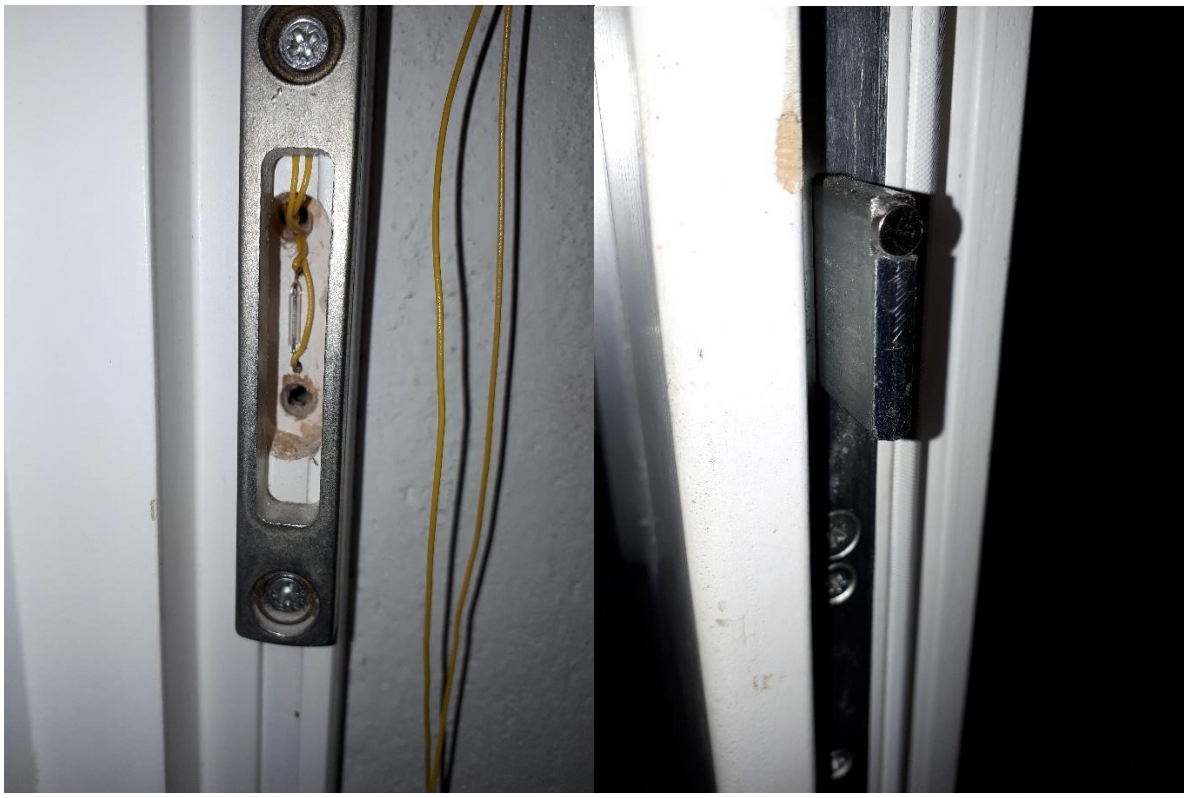

*Slika 20 - Stikalni načrt krmilnika garažnih vrat*

Za krmiljenje releja sem dodal še tranzistor, ker izhod krmilnika ne prenese tako velikega toka in bi se lahko poškodoval. Kontakte releja sem priključil vzporedno k tipki, ki je originalno uporabljena za odpiranje in zapiranje vrat. Posledično ni bilo potrebno nobenih posegov v originalno vezje mehanizma za krmiljenje vrat. Poleg tega sem na vhod D1 priključil mikro stikalo (Slika 21), ki sem ga prilepil na mehanizem in z njim signaliziral stanje, ali so vrata zaprta ali odprta. Če so vrata zaprta, je stikalo razklenjeno in je na vhodu D1 visok nivo, če pa so vrata odprta, je stikalo sklenjeno in povezano na maso, zato je na vhodu D1 nizek nivo.

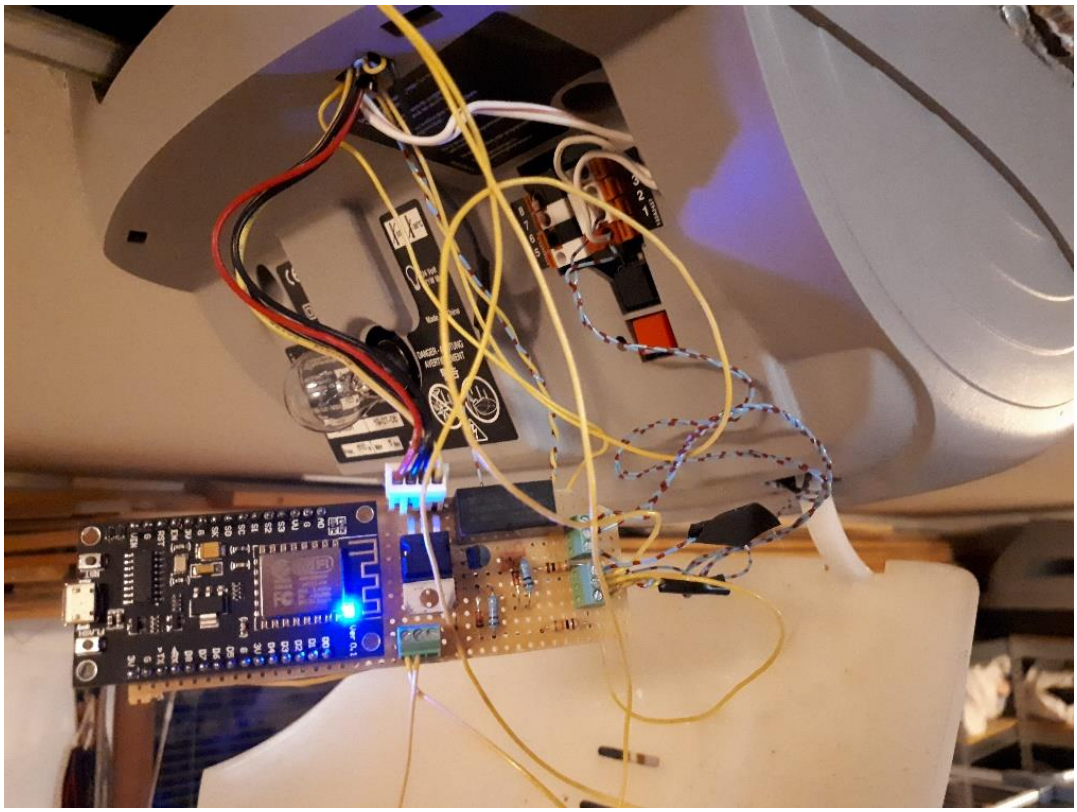


*Slika 21 - Stikalo dvižnih vrat*

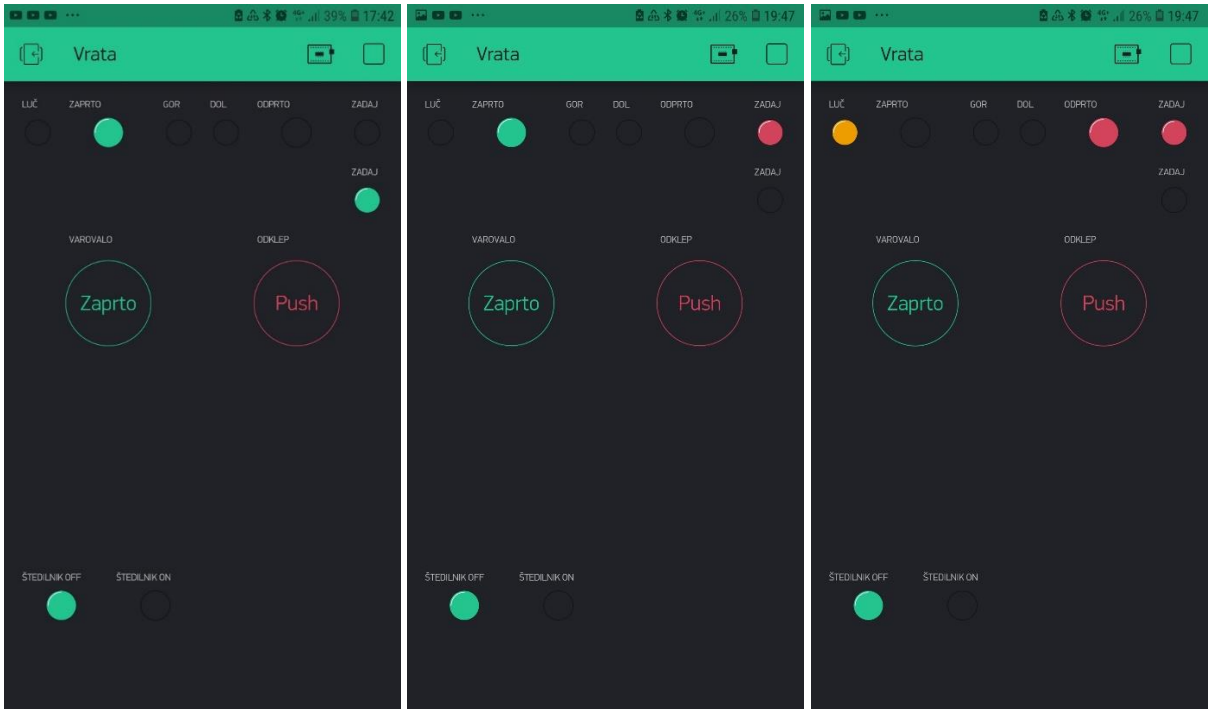
V aplikaciji Blynk (Slika 24) na ta način krmilim dva »LED« indikatorja, ki mi povesta, ali so vrata odprta ali zaprta. Za zaprta vrata sem uporabil zelen indikator, za odprta vrata pa rdeč »LED« indikator. Indikatorje krmilim s pomočjo virtualnih vrat, podobno kot pri merilniku zvoka, ki je opisan v poglavju 4.3. Sistem sem razširil še z dodatnim stikalom tako, da lahko nadzorujem še zaklenjenost zadnjih vrat v garaži. Za ta namen sem uporabil magnetno (reed) stikalo in magnet, ki sem ga namestil na zapah ključavnice. Reed kontakt sem namestil v odprtino na podbojih vrat (Slika 22). Ko so vrata zaklenjena, se zapah z magnetom približa reed kontaktu in ga sklene, ko pa so vrata odklenjena, je zapah ključavnice in s tem posledično magnet odmaknjen, zato reed kontakt ni sklenjen. S stanjem stikala, tako kot pri dvižnih vratih, krmilim še dva »LED« indikatorja na aplikaciji Blynk tako, da zelen indikator pomeni zaklenjena vrata, rdeč pa odklenjena zadnja vrata. Obe stikali sta z žicami povezani na krmilnik NodeMCU, ki sem ga namestil v ohišje dvižnega mehanizma garažnih vrat (Slika 23). Stikalo za zadnja vrata uporablja vhod D4, ki je privzeto povezan z modro LED na krmilniku. Tako pri zaklenjenih vratih na krmilniku sveti v temi dobro vidna modra lučka. Vsak, ki zvečer pogleda v garažo, že od daleč vidi, da so tudi zadnja vrata zaklenjena in mu jih ni potrebno posebej preverjati.



*Slika 22 - Magnetno stikalo za običajno ključavnico*



*Slika 23 - Krmilnik, nameščen v ohišje dvižnega mehanizma garažnih vrat*




*Slika 24 - Aplikacija Blynk za odpiranje vrat*

# 6 Uporaba NodeMCU za daljinsko krmiljenje centralne kurjave

Za konec bom opisal še primer uporabe, ki ga sicer še nisem izvedel, vendar ga gotovo bom, saj se mi zdi zelo uporaben. Imam dedka, ki živi v Kranjski Gori, med tednom pa večino časa preživi v Črnučah. V Kranjski Gori ima centralno peč na olje, ki jo izključi, ko tam ne živi. Ko se pozimi vrne, je hiša bolj hladna, zato mora ob prihodu najprej prižgati peč, hiša pa se prijetno segreje šele čez nekaj ur. Še bolj kritičen primer pa je, ko se temperature naglo spustijo pod ničlo. Če tak mraz vztraja več dni, obstaja nevarnost, da zmrzne voda. Takrat mora fizično oditi v Kranjsko Goro samo zato, da prižge peč na minimalno temperaturo, da prepreči zmrzal, nakar se vrne v Črnuče. Zato sem prišel na idejo, da bi lahko peč vključeval s pomočjo krmilnika NodeMCU in aplikacije Blynk. V principu gre za zelo podoben primer kot pri garažnih vratih. Vklop peči je mogoče enostavno narediti z relejem, ki vklaplja in izklaplja 220 V napajanje gorilnika. V aplikaciji Blynk je potrebno samo zamenjati tipko z ON/OFF gumbom, tako da ob vklopu ustrezni izhod krmilnika ostane stalno vključen. S tem izhodom krmilnika potem preko tranzistorja krmilimo rele. Seveda mora biti rele dimenzioniran za ustrezen tok, v tem primeru vsaj za 10 A in 240 V izmenične napetosti. Ker je to nevarna napetost, je potrebno krmilno vezje dobro narediti in tudi dobro ločiti od nizkonapetostnega krmilnega dela. Čeprav gre v principu za enako vezje kot pri garažnih vratih, pa pri slednjih krmiljen tok in napetost na releju nista pomembna, ker krmilna tipka deluje na nizki napetosti in ne potrebuje skoraj nobenega toka. Zaradi nevarnosti dela z visokimi napetostmi se tega vezja še nisem lotil, saj se bom pred izdelavo vezja predhodno posvetoval s strokovnjakom.

Nadgradnja opisanega primera pa je nadzor nad samo pečjo. S stikalom peč lahko vklopimo, ne vemo pa, če ta res deluje. Ker uporablja peč oljni gorilnik, ki je po naravi precej glasen, ga lahko nadziramo kar s senzorjem zvoka. Če je vse v redu, moramo po vključitvi peči na aplikaciji Blynk ob delovanju gorilnika zaznati določen hrup . Po nekem času, ko se voda v peči segreje, pa se mora gorilnik izključiti, kar prav tako zaznamo s senzorjem hrupa. Sistem lahko enostavno izboljšamo še tako, da dodamo še en senzor zvoka, s katerim zaznavamo delovanje obtočne črpalke. Ko je črpalka vključena, proizvaja šum, ki ga senzor lahko zazna.

Dodatna izboljšava je možna tudi tako, da merimo temperaturo vode v ceveh. Za ta primer pa bi moral narediti senzor temperature, kar lahko naredim z uporabo termistorja ali pa bi uporabil namenski čip LM35. Ker gre v tem primeru za analogne senzorje, bi moral zopet uporabiti multiplekser, ki pa sem ga že izdelal za senzorje zvoka. Za nadzor peči bi tako priključil dva senzorja zvoka in en senzor temperature. Razlika bi bila le ta, da bi v Blynk aplikaciji za temperaturni senzor moral zamenjati »LED« indikator s številčnim merilnikom in ga umeriti v stopinjah Celzija.

# 7 Razprava v obliki ankete

## 7.1 Anketni vprašalnik

V nadaljevanju je prikazan vprašalnik, ki sem ga razdelil učencem in učiteljem na svoji šoli z namenom, da zberem mnenje o tematiki svoje raziskovalne naloge med ljudmi, ki o tej nalogi nič ne vedo. Mnenje mi je pomembno, ker me je zanimalo, kako ljudje razmišljajo in gledajo na možnost uporabe določenih tehnologij v vsakdanjem življenju. Pred reševanjem anketnega lističa sem učencem predstavil, kakšen problem rešujem v okviru raziskovalne naloge.

*Iskreno vas prosim, da izpolnite anketo in se že vnaprej zahvaljujem za vaše konstruktivne odgovore.*

*Anketa je anonimna.*

1. Ali se vam zdi, da je javne površine potrebno nadzorovati? (obkroži)
   DA
   NE

2. Kakšen nadzor javnih površin se vam zdi najbolj primeren? (napiši)

_____

3. Na kakšen način bi spremljali premikanje živali? (obkroži)
   A) Z videonadzorom
   B) Z zvočnim nadzorom
   C) Drugo

4. Ali se vam zdi pametno nadzirati gospodinjske aparate prek telefona? (obkroži)
   DA
   NE

5. Ali se vam zdi nadzor zvoka v prostoru prek telefona dobra zamisel? (obkroži)
   DA
   NE

6. Prosim, napišite še druge možne načine uporaba nadzora zvoka. (napiši)

_____

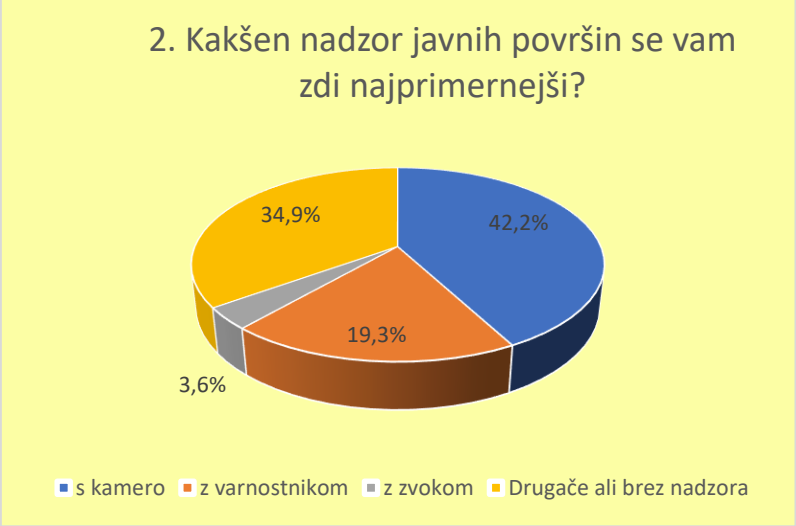
***Hvala lepa.***

## 7.2 Analiza rezultatov ankete

Vprašalnik sem razdelil med učence predmetne stopnje ter med učitelje. Nazaj sem prejel 83 izpolnjenih vprašalnikov, ki sem jih nato statistično analiziral. V nadaljevanju so podani rezultati v grafični obliki.


*Slika 25 - Analiza odgovorov na 1. vprašanje*

Prvo vprašanje je čisto splošne narave, saj me je zanimalo, kaj anketiranci menijo o nadzoru javnih površin. Velika večina, več kot dve tretjini, se je strinjala, da je javne površine potrebno nadzirati.




*Slika 26 - Analiza odgovorov na 2. vprašanje*

Drugo vprašanje je bilo odprtega tipa in zanimivo je, da je velika večina, več kot 40 odstotkov, menila, da je najboljši nadzor javnih površin video nadzor. Polovica manj anketirancev, slabih 20 odstotkov, pa meni, da je najboljši nadzor fizični z varnostniki, policijo ali redarji. Majhna skupina, komaj 3,6 odstotkov, bi javne površine nadzirala z zvokom. Preostala skupina 35 odstotkov pa na vprašanje ni odgovorila ali pa je podala

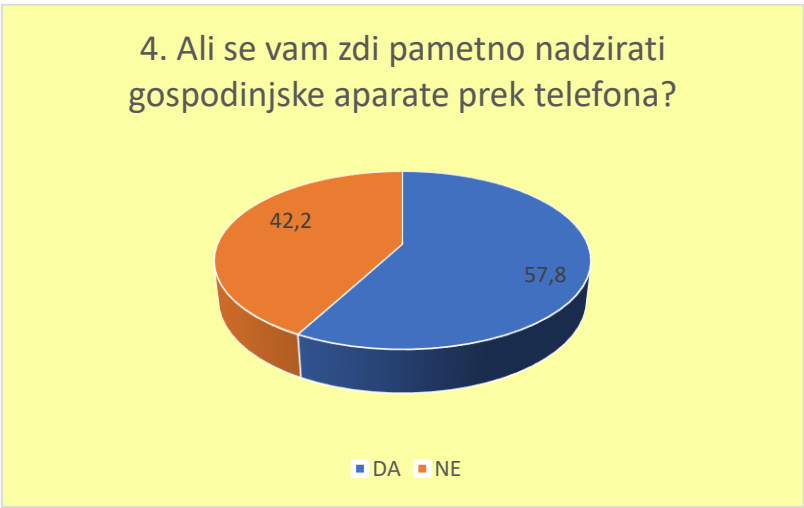
unikaten odgovor. Zanimiv mi je bil odgovor »merilci temperature«, ki ustreza trenutno aktualnemu dogajanju – širjenju korona virusa.

Tretje vprašanje se je konkretno nanašalo na tematiko raziskovalne naloge, le da tega anketiranci niso vedeli.


*Slika 27 - Analiza odgovorov na 3. vprašanje*

Večina anketirancev, skoraj 40 odstotkov, je za odgovor izbrala video nadzor, kar lahko razumemo kot standardno rešitev. Zanimivo pa je, da je popolnoma enako število, 38,6 odstotkov, izbralo zvočni nadzor, kar je po mojem mnenju alternativna rešitev, ki je tema moje naloge. Slaba četrtina pa je izbrala tretjo možnost.

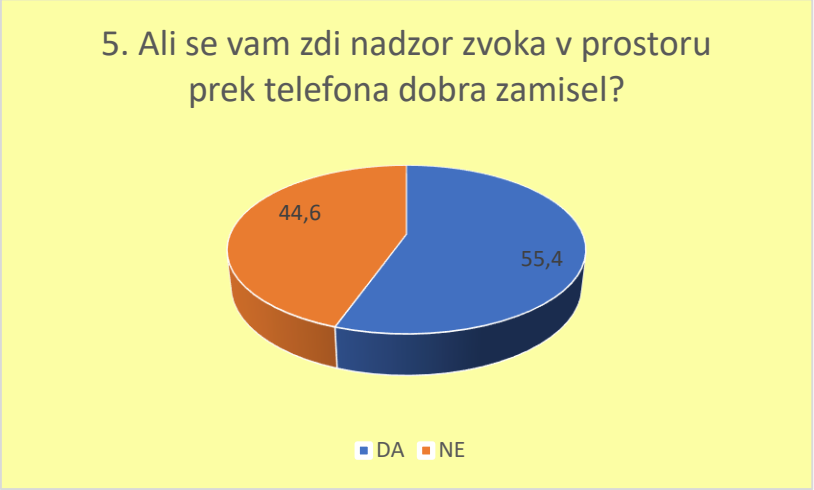
Pri četrtem vprašanju me je zanimalo, kako so anketiranci naklonjeni uporabi telefonov za nadzor domačih naprav, kar je tudi tema moje raziskovalne naloge.




*Slika 28 - Analiza odgovorov na 4. vprašanje*

Odgovor je pokazal, da je več kot polovica anketirancev, skoraj 58 odstotkov, naklonjena nadzoru naprav prek telefona.

Peto vprašanje se prav tako nanaša na tematiko raziskovalne naloge.




*Slika 29 - Analiza odgovorov na 5. vprašanje*

Tudi tu je več kot polovica anketirancev odgovorila pritrdilno, le da v malenkost manjšem odstotku.

Zadnje vprašanje je bilo odprtega tipa, s katerim sem želel pridobiti ideje anketirancev.


*Slika 30 - Analiza odgovorov na 6. vprašanje*

Na vprašanje je sicer odgovorila približno polovica anketirancev, vendar so bili njihovi odgovori zelo različni. Iz vseh pa sem vseeno lahko prepoznal tri skupine odgovorov. Od teh jih je bilo največ, 12 odstotkov, za odganjanje mrčesa, živali ali ptičev. Druga skupina, 6 odstotkov, je prepoznala možnost, da prek zvoka zaznamo delovanje aparatov, tretja skupina, 3,6 odstotkov, pa je uporabnost prepoznala za proženje alarma.

Med ostalimi različnimi odgovori pa so mi bili zelo zanimivi naslednji:

- za sledenje, sledimo lahko izvoru zvoka
- za  izbor aktivne kamere, prikazujemo tisto sliko, kjer je zaznana največja glasnost
- upravljanje naprav
- ugotavljanje nivoja hrupa
- prepoznavanje določenih frekvenc
- štetje prometa

# 8 Zaključek

V pričujoči raziskovalni nalogi sem prikazal več možnosti uporabe modula ESP8266, predvsem v kombinaciji s senzorjem zvoka, ki sem ga tudi samostojno razvil in izdelal. S kombinacijo množice senzorjev, multiplekserja, krmilnika NodeMCU in aplikacije Blynk sem izdelal prototip naprave za spremljanje lokacije živali na nekem prostoru. Princip delovanja sem tudi v praksi potrdil tako, da sem zvok živali simuliral s piezo piskačem. Zavedam se, da to še ni končni izdelek in da bi ga bilo potrebno za dejansko uporabo še izpopolniti in razširiti. Povečati bi bilo potrebno število senzorjev in jih za zunanjo uporabo zaščititi pred vodo in vlago. Zvočne senzorje bi bilo verjetno potrebno narediti še bolj občutljive oziroma z nastavljivo občutljivostjo, ki bi jo lahko celo programsko krmilili. Za ta namen bi moral razširiti tudi krmilni del, da bi podpiral večje število vhodov. Za resnično uporabo bi bilo potrebno marsikaj postoriti tudi na uporabniškem vmesniku. Trenutna izvedba zgolj prikazuje senzor, ki zaznava največjo glasnost. Če se žival preneha premikati, trenutna aplikacija zatemni vse indikatorje, ker ne zaznava nobenega zvoka, zato si moramo zanjo pozicijo živali zapomniti. Ker Blynk podpira tudi REST protokol, bi lahko izdelali spletno aplikacijo, ki bi grafično prikazovala nadzorovan prostor. Pozicijo živali bi na osnovi prebranih podatkov senzorjev lahko bolj natančno izračunali glede na jakost sosednjih senzorjev. Žival bi grafično prikazali s točko ali njeno podobo, ki bi se v skladu z gibanjem živali pomikala po zaslonu. Če se žival ustavi, njena podoba ostane na zadnjem izmerjenem mestu. Na ta način lahko z lahkoto prikažemo tudi gibanje več živali v prostoru. Iz povedanega in same izvedbe naprave lahko potrdimo hipotezo, da se da površine nadzirati tudi drugače.

V nalogi sem tudi prikazal, da lahko enake senzorje zvoka in krmilnik enostavno uporabimo tudi za drugačne namene. Prvi primer je neposredna uporaba rešitve za merjenje hrupa v razredu. S petimi senzorji lahko v celoti pokrijemo razred. Prilagoditi je potrebno le kodo v krmilniku tako, da meri poprečje vseh senzorjev in omogoča nastavljanje mejnih vrednosti hrupa. Na osnovi te vrednosti lahko krmilimo indikatorje v aplikaciji Blynk ali pa fizične lučke, ki ponazarjajo glasnost. Prav tako imamo možnost centralnega zbiranja podatkov, ki jih lahko kasneje statistično obdelujemo, ovrednotimo in na njihovi osnovi tudi ukrepamo.

Senzor zvoka lahko uporabimo tudi za zaznavanje delovanja določenih naprav, kot na primer nadzor peči centralne kurjave. Z njim pa lahko naredimo tudi alarm. Če nekdo vstopi v prostor, povzroči nek hrup, ki ga senzor zazna in sproži alarm. Krmilnik lahko prek releja vključi sireno, hkrati pa nam na telefon pošlje elektronsko sporočilo, saj aplikacija Blynk to omogoča.

S tem je hipoteza o drugih načinih uporabe potrjena.

Na podlagi ankete sem spoznal še nekaj možnosti uporabe, ki mi prej niso prišle na misel. Zanimiva možnost se mi zdi tudi uporaba senzorja za odganjanje škodljivih živali, na primer ptičev, ki kradejo vrtne pridelke. Senzor lahko zazna čivkanje ptičev, krmilnik pa vključi glasen zvočni vir, ki jih prežene. Podobno bi lahko zaznali leteči mrčes, ki s krili povzroči brenčanje. Ko bi krmilnik zaznal njihov hrup, bi vključil močan ventilator, ki bi jih odpihnil. Na

podlagi ankete lahko potrdim hipotezo, da so predstavljene rešitve sprejemljive za sovrstnike.

Za konec lahko rečem, da kombinacija krmilnika ESP8266 s senzorjem zvoka in različnih stikal ter relejev omogoča najrazličnejše možnosti uporabe v vsakdanjem življenju. Pri iskanju le-teh pa smo največkrat omejeni le z lastno domišljijo.

# 9 Viri in literatura


[1] RANDOM NERD TUTORIALS. Pridobljeno 3. 3. 2020 s https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/

[2] Blynk Getting started. Pridobljeno 3. 3. 2020 s https://blynk.io/en/getting-started

[3] LM358 Datasheet. Pridobljeno 3. 3. 2020 s http://www.ti.com/lit/ds/slos068w/slos068w.pdf

[4] Precision rectifier. Pridobljeno 3. 3. 2020 s https://en.wikipedia.org/wiki/Precision_rectifier

[5] CD5041 Datasheet. Pridobljeno 3. 3. 2020 s https://global.oup.com/us/companion.websites/fdscontent/uscompanion/us/pdf/microcircuits/students/logic/cd4051-ti.pdf

[6] LM7805 Datasheet. Pridobljeno 3. 3. 2020 s https://www.st.com/resource/en/datasheet/l78.pdf

[7] Arduino reference. Pridobljeno 3. 3. 2020 s https://www.arduino.cc/reference/en/

[8] Node MCU Blynk example. Pridobljeno 3. 3. 2020 s https://www.instructables.com/id/Simple-Led-Control-With-Blynk-and-NodeMCU-Esp8266-/