Mladi za napredek Maribora 2020
37. srečanje

Pametni zvonec

Raziskovalno področje: **ELEKTROTEHNIKA, ELEKTRONIKA**

Inovacijski predlog

PROSTOR ZA NALEPKO

Avtor: KEVIN BARON LAH, MATIC ŠULC

Mentor: IVANKA LESJAK
Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR
Število točk: 160/ 170

Maribor, 2020

Mladi za napredek Maribora 2020
37. srečanje

Pametni zvonec

Raziskovalno področje: **ELEKTROTEHNIKA, ELEKTRONIKA**

Inovacijski predlog

PROSTOR ZA NALEPKO

Maribor, 2020

KAZALO VSEBINE

KAZALO SLIK

1. ZAHVALA

Zahvaljujemo se mentorju za vso strokovno pomoč in podporo pri izdelavi raziskovalne naloge, ter šoli, ki nam je nudila potrebno opremo in prostore, kjer smo lahko izdelek realizirali. Prav tako gre zahvala tudi vsem ostalim, ki so kakorkoli pripomogli k izdelavi te naloge.

2. POVZETEK

V tej raziskovalni nalogi bomo predstavil delovanje pametnega zvonca. S pomočjo mikrokrmilnika ESP32 – CAM bomo klasičnemu brezžičnemu zvoncu dodali še možnost video nadzora nad obiskovalci. Izdelek bo vseboval tudi ostale varnostne ukrepe, kot so senzor gibanja, ter dodatno shranjevanje slik na oblačne storitve, če bi uporabnik zaznal nedovoljen vstop. Sprejemnik zvonca bo deloval na osnovi mikroračunalnika Raspberry PI, ki bo ob sprožitvi predvajal zvočni signal, ki mu ga bo določil uporabnik.

3. UVOD

Za izdelavo pametnega zvonca smo se odločili, ker doma nimamo možnosti vgraditve stacionarnega zvonca, obstoječe rešitve za brezžične pa so omejene (kratka doba baterije, omejen vmesnik …).

Pri naši rešitvi bo lahko uporabnik sam konfiguriral nastavitve zvonca in prejemal obvestila na želeno mobilno napravo. Naprava bo delovala na baterije in ne bo potrebovala konstantnega napajanja.

4. VSEBINSKI DEL

4.1 ESP32 – CAM

Slika 1: ESP32 – CAM (vir: diymore.cc)

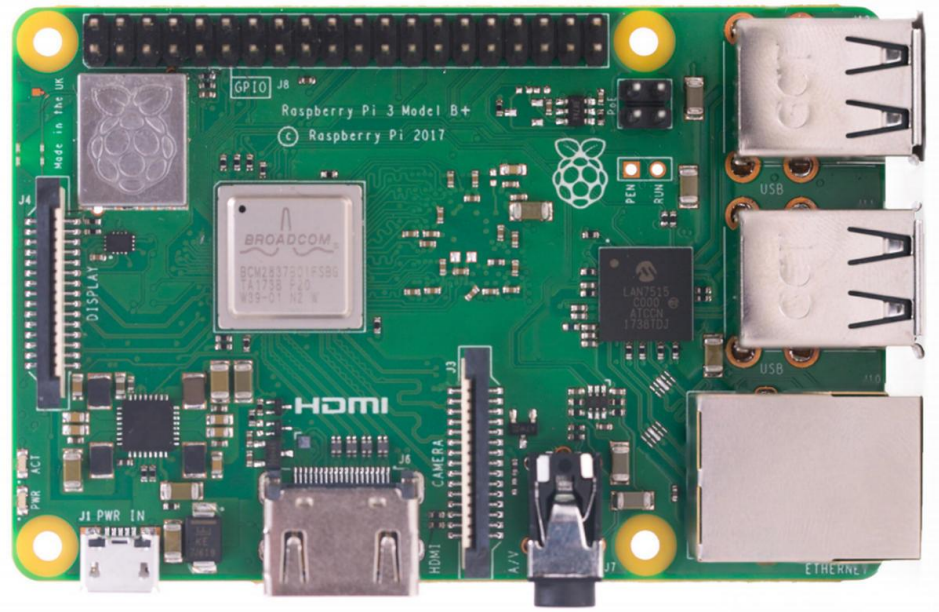
ESP32 – CAM je mikrokrmilnik, ki temelji na svojem predhodniku – ESP32. Po zgradbi mu je enak, le da ima dodan še modul za kamero in SD kartico. ESP32 je serija nizkonapetostnih mikrokrmilnikom z vgrajenim bluetooth-om in Wi-Fi-jem. Srce krmilnika je 32-bitni Tensilica Xtensa LX6 mikroprocesor, ki v »deep sleep« načinu potrebuje le 5μA za delovanje. Ima 9 vhodno izhodnih/pinov, ter deluje na napetostih 3.3 in 5V. Vgrajen ima tudi ADC modul, zato nam vsi vhodno/izhodni pini omogočajo tudi branje analognih vrednosti. Zaradi svojih majhnih dimenzij je primeren za uporabo v nadzornih kamerah, CCTV ipd.

Programiramo ga lahko s programskim orodjem Arduino. Za fizično povezavo lahko uporabimo FTDI pretvornik, ali pa kar razvojno ploščico Arduino, na kateri moramo onemogočiti delovanje ATmega mikrokrmilnika.

Specifikacije:

- Mikroprocesor Tensilica Xtensa LX6 32bit
- 520KB SRAM
- Wi - Fi 802.11 b/g/n/, Bluetooth 4.2
- DIP-16 package
- Obratovalna napetost: 5V, deluje tudi pri 3.3V
- Dimenzije: 27*40.5*4.5mm
- Teža: 10g

4.2 Raspberry PI



Slika 2: Raspbery PI (vir: raspberrypi.org)

Raspberry Pi (RPi) je mikroračunalnik, ki so ga razvili z namenom spodbujanja poučevanja osnov računalništva po svetu. Deluje na operacijskem sistemu Raspbian, ki temelji na Linux-ovem jedru. V našem izdelku ga bomo uporabili kot sprejemnik, ki bo ob prejetem signalu sprožil zvočnik, priključen nanj. Vseboval bo SD kartico, na katerega bomo naložili operacijski sistem in 3.5mm audio priključek, kamor bomo priključili zvočnike. Deluje kot samostojna enota, zato ga bi lahko po želji zamenjali s kakršnimkoli mikrokrmilnikom, ki podpira povezavo z internetom in ima 3.5mm audio priključek.

4.3 PIR senzor

Slika 3: PIR senzor (vir: canadarobotix.com)

Pasivni infrardeči senzor (PIR) zaznava infrardečo svetlobo, ki jo oddajajo objekti, kot so ljudje, živali ipd. Zaznava sevanja v valovnih dolžinah med 7 in 14 µm, zato je idealen za uporabo pri ljudeh, ki oddajajo sevanje v tem spektru. Deluje po načelu fizike, ki pravi, da vsi objekti s temperaturo nad 0K oddajajo toplotno energijo v obliki sevanja. To sevanje s človeškim očesom ni vidno, saj poteka pri infrardečih valovnih dolžinah, zaznajo pa ga lahko elektronske naprave, kot so npr. senzorji. Sam senzor je sestavljen iz piroelektričnega senzorja in leče. Ko zazna premik, na svojem izhodnem pin-u vrne logični 1 (HIGH), počaka 2 sekundi (delay), in vrne logično 0 (LOW), če je oseba zapustila vidno polje.

Specifikacije:

- Obratovalna napetost: 5V
- Zamik: 2s
- Kot zaznavanja: 100 stopinj
- Razdalja zaznavanja: 3-5m

4.4 Tipka

Slika 4: Tipka (vir: amazon.de)

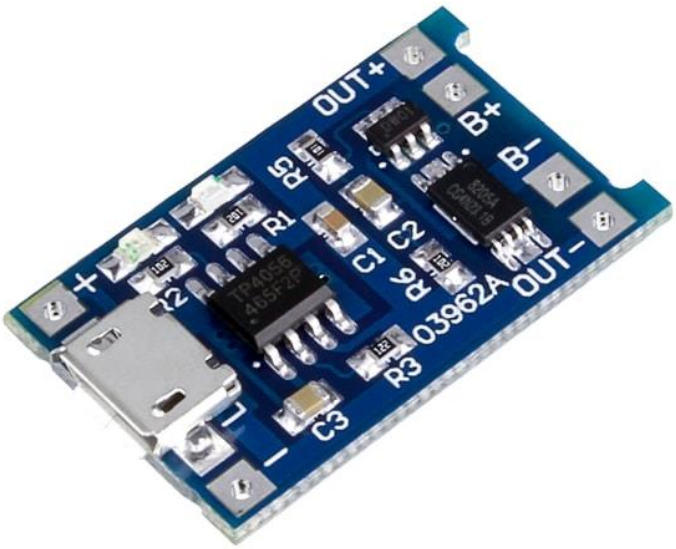
Tipka je mehansko stikalo, ki se uporablja za sklenitev električnega tokokroga ob pritisku. Ima 2 kontakta, na katera priključimo žice v tokokrogu. V našem primeru se bo uporabljala za proženje zvonenja. Posebnost tega stikala je tudi IP 65 odpornost, kar pomeni, da je odporna proti prašnim delcem in vodi pod nizkim pritiskom.

4.5 LED dioda

Slika 5: LED Dioda (vir: conrad.si)

LED dioda je polprevodniški element, ki sveti kadar prevaja električni tok. Različne diode se razlikujejo po svoji barvi, velikosti in karakteristikah. Od klasičnih žarnic se razlikujejo po tem, da imajo mnogo daljšo življenjsko dobo in manjše izgube. Deluje na principu spremembe energetskega stanja elektrona. Ko atom preide na nižje energetsko stanje, odda odvečno energijo v obliki elektromagnetnega sevanja. Ta efekt je znan kot elektroluminiscenca. V našem primeru smo diodo uporabili le kot indikator delovanja in obveščanja o zaznanem premiku oz. kliku.

4.6 TP 4056



Slika 6: TP 4056 (vir: addicore.com)

TP4056 je polnilni modul za eno celične litij - polimerske (Li-Po) in litij - ionske (Li-ion) baterije.

Modul deluje na osnovi čipa TP4056, ki skrbi za polnjenje. Ob njem pa sta na vezju še DW01A in mosfet 8205A, ki varujeta baterijo pred izpraznitvijo in prenapolnjenjem.

Deluje pri vhodni napetosti od 4 do 8V, ki jo lahko priključimo preko USB priključka, ali pa preko stranskih »IN« kontaktov. Privzet tok, ki lahko teče skozi vezje je 1000mA, le tega pa lahko zmanjšujemo z zamenjavo upora, ki je označen z R3.

R_{PROG} (k)	I_{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

Slika 7: Tabela za upor na TP4056

4.7 18650 Li-Ion baterija

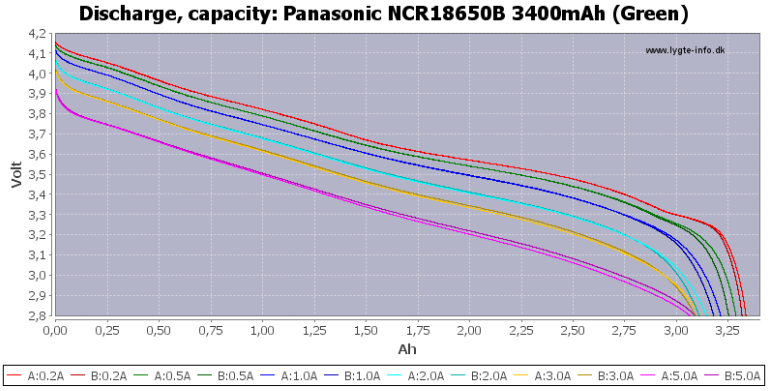
Slika 8: 18650 celica (vir: belidim.si)

»18650« je cilindrična izvedenka litij – ionske baterije, pri kateri se ioni premikajo od negativne elektrode k pozitivni med praznjenjem, ter iz pozitivne proti negativni pri polnjenju. Uporabljajo se v elektroniki, ter so ene najbolj priljubljenih polnilnih baterij.

Celica ima povprečno napetost 3.7V, ko je popolnoma napolnjena doseže prib. 4.2V napetosti. Njihova kapaciteta je odvisna od proizvajalca in posledično tudi od kvalitete. V povprečju imajo le te kapaciteto med 1000mAh in 3500mAh.

Specifikacije:

- Nazivna napetost: 3.7V
- Tok pri polnjenju in praznjenju: 1.3A – 0.52A
- Teža: 46.5g
- Dimenzije: 18.4mm Ø, 65.2mm višina



Slika 9: Graf praznjenja baterije (vir: lygte-info.dk)

4.8 Boost pretvornik

Slika 10: Boost pretvornik (vir: banggood.com)

Za napajanje mikrokrmilnika smo potrebovali konstantno napetost 5V, naša baterija pa je pri maksimalni napolnjenosti dosegla prib. 4.2V. Zato smo uporabili t.i pretvornik navzgor (ang. Boost converter), ki deluje v enosmernem tokokrogu. Deluje tako, da z manjšanjem toka dviguje napetost v vezju. Izbrali smo si verzijo, ki omogoča vhodno napetost od 0.9V do 5V, ter drži konstantno izhodno napetost pri 600mA. Sprva smo želeli uporabiti modul, ki omogoča ročno nastavitev izhodne napetosti, a zaradi padca napetosti pri praznjenju baterije to ni bilo mogoče.

Najprej smo v tokokrogu premerili in sešteli vsote tokov pri 3.3V:

ESP32 – CAM = 220mA

2x LED = 40mA

TP4056 = 10 µA

$$P1 = U * I$$

$$P1 = 3.3V * 0.26001\text{A} = 0.962037\text{W}$$

$$0.962037\text{W} = 5\text{V} * \text{I}$$

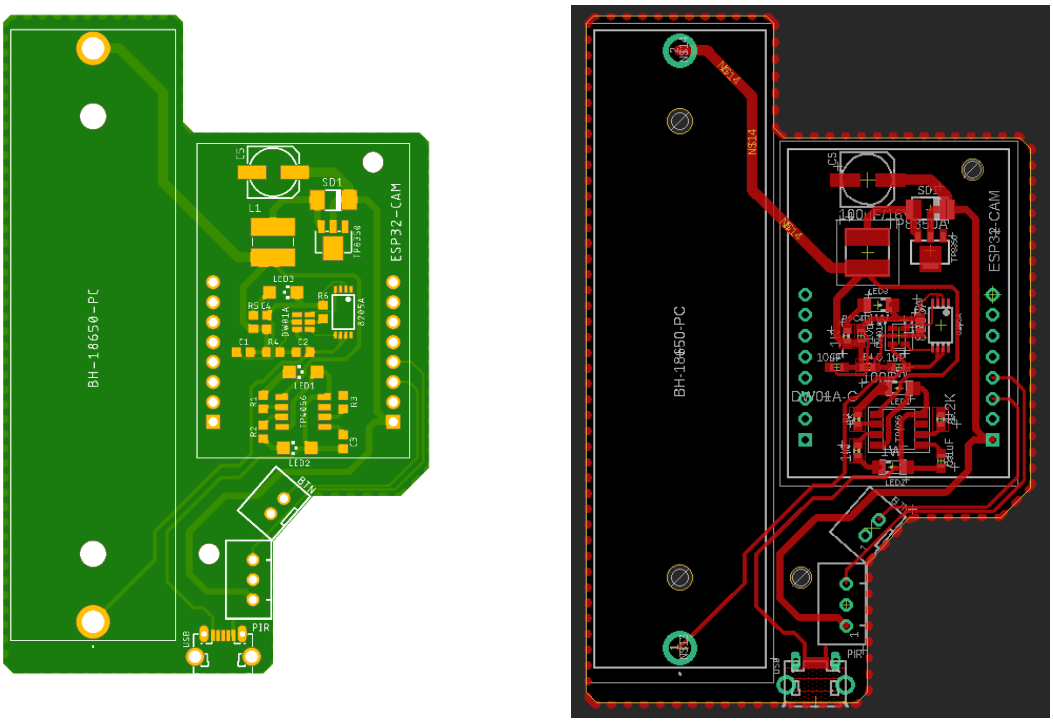
$$I = \frac{0.962037\text{W}}{5V} = 0.1924074\ A$$

S tem izračunom smo ugotovili, da so vse komponente na vezju spodobne delovanja s tokom, ki teče skozi naše vezje.

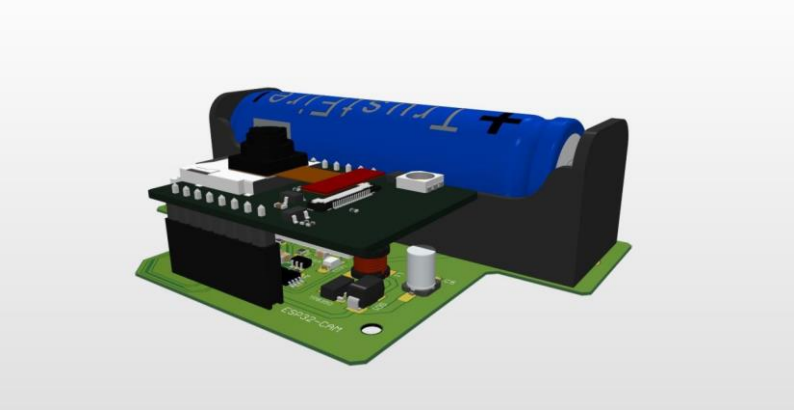
4.9 Model vezja in ohišja izdelka

4.9.1 Model vezja

Model vezja smo izdelali v programu Altium Designer. Vezje je enostransko in vsebuje vse uporabljene komponente v PCB obliki. 3D model vezja smo generirali v enakem programu, ter s pomočjo tega izdelali tudi ohišje izdelka.



Slika 11: Shematika (vir: Avtor naloge)



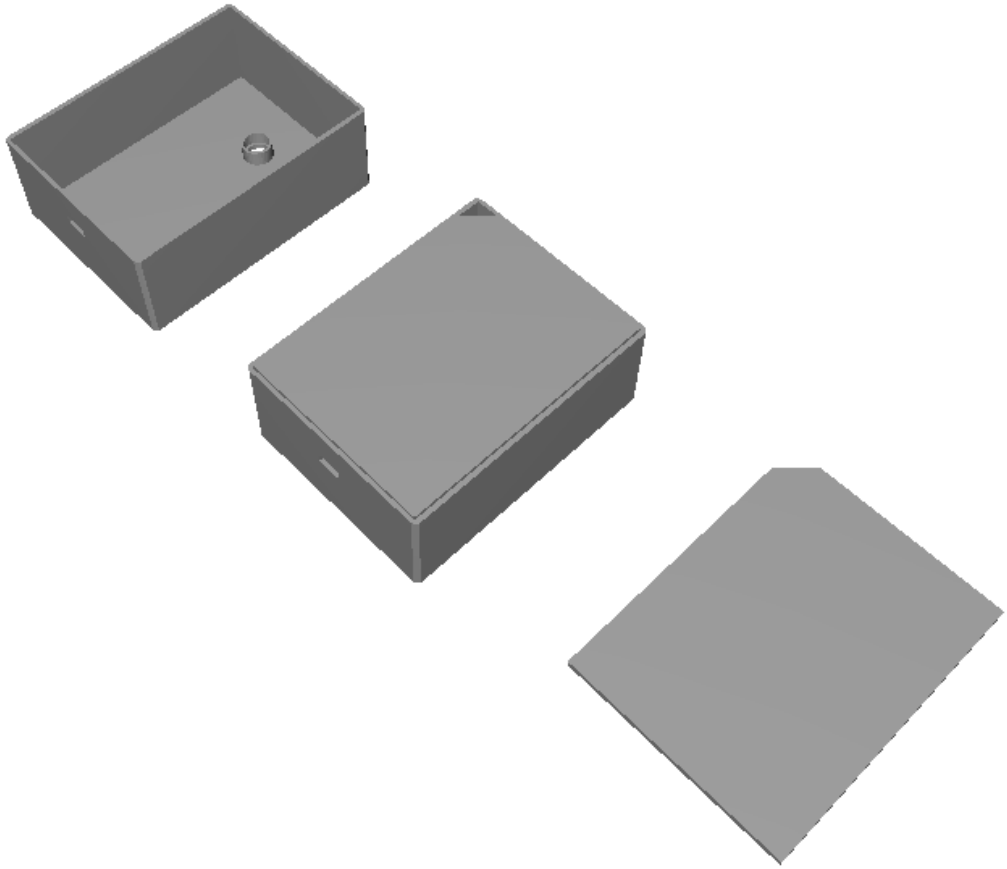
Slika 12: 3D model izdelka (vir: Avtor naloge)

4.9.2 Model ohišja

Model ohišja smo izdelali v programu Blender. Ko smo izvozili shematiko za tiskano vezje, smo pridobili še .stl datoteko, ki smo jo uvozili v program Blender. Na podlagi tega smo lahko natančno izdelali ohišje, ki ga bomo natisnili s 3D tiskalnikom.

Sestavljeno bo iz sprednjega dela, ter zadnjega pokrova, ki ga bo lahko uporabnik zdrsnil v reže. V notranjem sprednjem delu bomo dodali še nosilni mehanizem, ki bo s pomočjo vijakov držal vezje na svojem mestu.

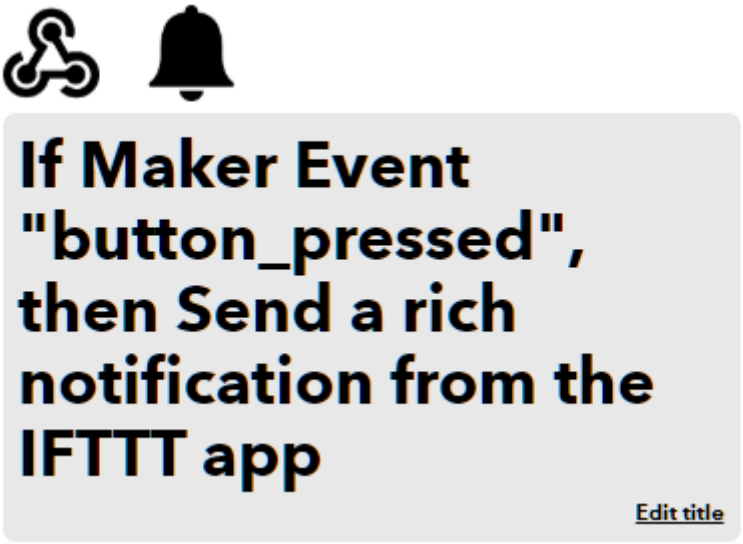
Pri sami izdelavi smo ugotovili, da bo priložen kabel za kamero prekratek, zato smo dokupili tudi podaljšek za 24 pinski FCP kabel, preko katerega je povezan modul za kamero.



Slika 13: Model ohišja (vir: Avtor naloge)

4.10 IFTTT

Za komunikacijo zvonca z mobilno napravo smo uporabili aplikacijo »IFTTT«, oziroma »If This Then That«. (slovensko: Če se zgodi to, stori to). Uporabnik ima možnost kreiranja t.i »applet-a«, kar je v bistvu procedura, ki se izvrši ob klicu na zahtevano dejanje.

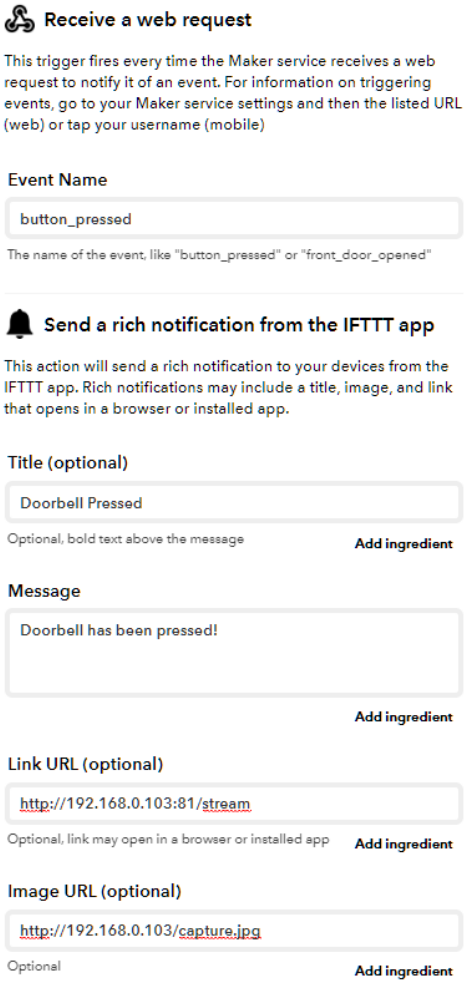


Slika 14: IFTTT applet (vir: Avtor naloge)

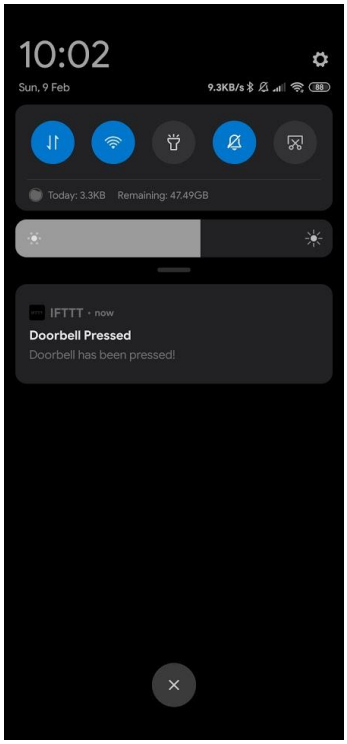
V našem primeru je »This« dejanje, klic na API, ki ga izvrši ESP32 s pomočjo našega privatnega ključa in imena akcije. Ko IFTTT zazna akcijo, sproži obvestilo v mobilni aplikaciji, kateremu lahko določimo razne dodatke. V našem primeru je to sporočilo, trenutna slika iz kamere, ter povezava na video prenos v živo ob kliku. Sekundarni »applet« bo izvedel klic na sprejemnik, ki bo sprožil zvonec.

Uporaba zastonjske aplikacije bo uporabnikom zvonca omogočala prosto razpolaganje s podatki, saj ne bodo omejeni samo na eno mobilno aplikacijo. Sliko oz. video prenos bodo lahko pošiljali na katerokoli napravo, shranjevali v podatkovne baze, ipd. V našem primeru bomo kot dodatni applet dodali še shranjevanje slik v Google Drive, kjer se bodo delile v mape za posamično akcijo (klik zvonca, zaznan premik). To nam bo omogočilo nadzor tudi takrat, ko ne bomo sami imeli dostopa do interneta, da bi preverili slike.

Za testiranje obvestil smo API uporabili direktno v brskalniku. V iskalno vrstico smo vnesli: https://maker.ifttt.com/trigger/ + ime_akcije + /with/key/ + naš_privatni_ključ



Slika 15: IFTTT akcija (vir: Avtor naloge)



Slika 16: Obvestilo na telefonu

5. PROGRAMSKA KODA

Koda.gs

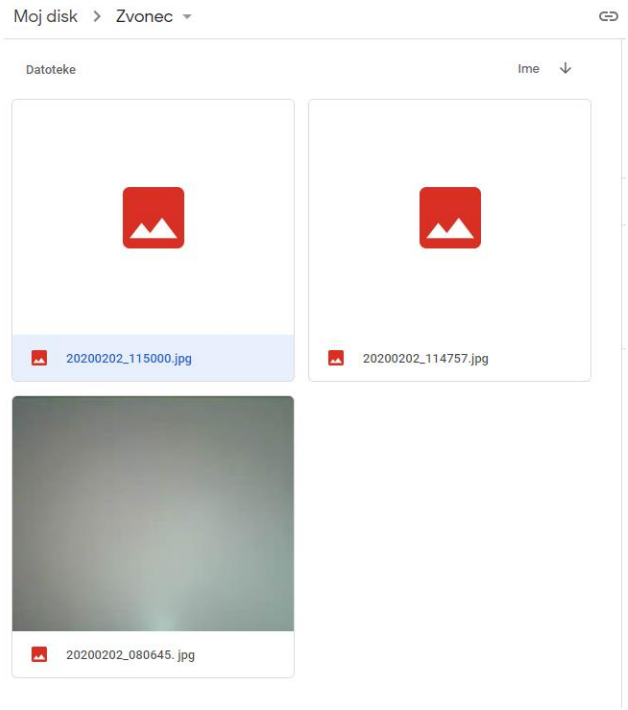
```
function doPost (e) {
  var data = Utilities.base64Decode (e.parameters.data);
  var filename = Utilities.formatDate (new Date (), "GMT+1", "yyyyMMdd_HHmm") + ". jpg";
  var blob = Utilities.newBlob (data, e.parameters.mimetype, filename);


  var folder, folders = DriveApp.getFoldersByName ("Zvonec");
  if (folders.hasNext ()) {
    folder = folders.next ();
  } else {
    folder = DriveApp.createFolder ("Zvonec");
  }
  var file = folder.createFile (blob);

  return ContentService.createTextOutput ("Complete.")
}
```

Slika 17: Skripta za nalaganje na oblak (vir: Avtor naloge)

Za primarno shranjevanje slik na Google Drive smo uporabili Google Script API. Ustvarili smo skripto, ki je kot podatek dobila datoteko .jpg in njeno ime preko metode POST. Ko skripta prejme datoteko, preveri če mapa z imenom »Zvonec« obstaja. Če ne obstaja, jo kreira, v nasprotnem primeru pa spremeni direktorij v mapo. V njej ustvari .jpg datoteko, katere ime je enako: letomesecdan_ureminute.



Slika 18: Slike naložene na oblak (vir: Avtor naloge)

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "Base64.h"               //knjiznica za drive
#define CAMERA_MODEL_AI_THINKER   //izbor nasega modela kamere
#include "camera_pins.h"          //selekcija pinov

void ifttt_send(const char *event); //prototip funkcije s pointerjem na event
void processImage(const char *event);   //procesiranje slike ob kliku/premiku

const char* ssid = "ime_omrezja";
const char* password = "geslo_omrezja";
const char *host = "maker.ifttt.com";
const char *privateKey = "";  //privatni kljuc za ifttt
const char *myDomain = "script.google.com"; //domena za google drive skripto

String myScript  = "";  //pot do google drive skripte
String myFilename = "filename=ESP32-CAM.jpg";
String mimeType = "&mimetype=image/jpeg";
String myImage = "&data=";

const int buttonPin = 2;  //gumb pin
const int sensorPin = 4;  //pir senzor pin

int buttonState;             // trenutno stanje gumba (za debounce)
int lastButtonState = LOW;   // prejsnje stanje gumba

long lastClickTime = 0;  // cas ob zadnjem kliku gumba
long debounceTime = 50;    // zamik za debounce
```

Slika 19: Spremenljivke v programu (vir: Avtor naloge)

V programu smo deklarirali spremenljivke, ki uporabniku omogočajo spremembo vrednosti, brez spreminjanja celotnega programa. Za pravilno delovanje smo morali vključiti tudi knjižnice za WiFi in pretvorbo v base 64 algoritem, ter definirali »pinout« naše kamere, saj obstaja več modelov enakega produkta. Uporabnik naj bi v tem delu programa spreminjal le ime in geslo omrežja, ključ za IFTTT, ter povezavo do skripte za shranjevanje slik.

```
void loop() {
    while (WiFi.status() == WL_DISCONNECTED) { //ce se wifi povezava prekine, resetiraj
    ESP.restart();
    }

   int reading = digitalRead(buttonPin);  //preberi stanje gumba
   if (reading != lastButtonState) {
    lastClickTime = millis();           //zabelezimo kdaj se je gumb nazadnje kliknil
  }
  if ((millis() - lastClickTime) > debounceTime)  //ce je debounce time pretekel
  {
    if (reading != buttonState){
      buttonState = reading;                      //nastavimo buttonState na novo stanje

      if (buttonState == LOW) {                   //ko je gumb pritisnjen
        processImage("button_pressed");           //obdelamo sliko, obvestila...
        Serial.print("DING - DONG");
      }
    }
  }

  lastButtonState = reading;                      //nastavimo zadnji status gumba

  int pirValue = digitalRead(sensorPin);          //preberemo vrednost senzorja (0 ali 1)
  if(pirValue == 1){                              //ko je vrednost 1, je zaznan premik
      processImage("motion_detected");
      Serial.println("Zaznan premik!");
      delay(1000);
    }
}
```

Slika 20: Glavni program (vir: Avtor naloge)

V glavnem programu imamo 2 različna preverjanja. Najprej z uporabo »debounce« algoritma preberemo vrednost gumba. Ta algoritem potrebujemo, ker lahko včasih pride do popačenja signala kar bi po nepotrebnem prožilo kamero. Ko zaznamo pritisk gumba, opravimo klic na funkcijo processImage, ki obdela shranjevanje slik, ter pošlje potrebna obvestila. Enako storimo s PIR senzorjem, a v tem primeru le preberemo digitalno vrednost (0 = ni premika, 1 = zaznan premik).

```
void ifttt_send(const char *event){

  WiFiClient client;

  String url = "/trigger/";
  url += event;
  url += "/with/key/";
  url += privateKey;

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  while(client.connected()){

    if(client.available()) {
      String line = client.readStringUntil('\r');
      Serial.print(line);
    }
    else {
      delay(50);
    };
  }
  client.stop();
}
```

Slika 21: Pošiljanje obvestil preko IFTTT (vir: Avtor naloge)

Za pošiljanje obvestil smo uporabili metodo ifttt_send, ki kot argument prejme ime dogodka, ki se je izvedel. S tem podatkov sestavi url, na katerega se poveže z metodo GET. Nato počaka, da strežnik vrne vrednost, ter jo izpiše v serijskem vmesniku.

```python
import pymysql as sql
import time
zvonenje = "/home/pi/zvonec/ding.mp3"   #definiranje mp3 datoteke

def prozi_zvonenje():                   #subrutina zvonenje
    subprocess.call(["aplay", zvonenje])

while(1):   #neskoncna zanka
    conn = sql.Connect(host='localhost', user='espuser', passwd='esppass123', db='esp_data') #povezava na bazo
    cursor = conn.cursor()
    query = 'SELECT * FROM sensordata ORDER BY timestamp DESC LIMIT 1' #vrnemo samo en (najnovejsi) podatek
    cursor.execute(query)   #izveden sql query
    results = cursor.fetchone() #preberemo prvi rezultat
    if(results[0] == 1)         #ce je zvonec pritisnjen, naj zazvoni
        prozi_zvonenje()
    time.sleep(2)               #pocakamo 2 sekundi, da manj obremenimo procesor
```

Slika 22: Branje iz podatkovne baze (vir: Avtor naloge)

Za sprejemnik smo uporabili Raspberry PI v kombinaciji s podatkovno bazo MySQL. Python skripta ob zagonu proži neskončno zanko, ki preverja status gumba.

```php
<?php
$servername = "localhost"; //spremenjivke za bazo
$username = "espuser";
$password = "esppass123";
$database = "esp_data";
$buttonvalue = $_GET['buttonvalue'];        //vrednost GET, ki jo dobimo iz url naslova. primer: url?spremenljivka=vrednost&...

$conn = new mysqli($servername, $username, $password, $database); //povezava na bazo

$sql = "INSERT INTO sensordata (buttonvalue) VALUES ($buttonvalue)";    //vstavi vrednost v bazo ob kliku gumba
$conn->query($sql); //izvedemo sql

if($buttonvalue == 1){                                                  //ce je vrednost 1
    sleep(3);               //pocakaj 3 sekunde
    $sql = "INSERT INTO sensordata (buttonvalue) VALUES (0)";   //nastavimo vrednost nazaj na 0 (konec zvonenja)
}

$conn->query($sql); //izvedemo sql
$conn->close();     //zapremo povezavo
?>
```

Slika 23: Vstavljanje v podatkovno bazo v PHP (vir: Avtor naloge)

Ob spremembi vrednosti (kliku) gumba uporabimo IFTTT da pošlje t.i »webhook« na našo php skripto, kar vstavi status v podatkovno bazo, skupaj s časovno značko.

buttonvalue	timestamp
0	2020-02-08 20:08:42
1	2020-02-08 20:12:05
1	2020-02-08 20:13:25
1	2020-02-08 20:13:34
1	2020-02-08 20:13:35
1	2020-02-08 20:13:35
0	2020-02-08 20:23:25
1	2020-02-08 20:56:49
0	2020-02-08 20:57:03
0	2020-02-08 21:00:05
0	2020-02-08 21:00:05
0	2020-02-08 21:00:09
0	2020-02-08 21:00:12
1	2020-02-08 21:00:17
1	2020-02-08 21:00:23
1	2020-02-08 21:01:28
0	2020-02-08 21:01:31
1	2020-02-08 21:01:35
0	2020-02-08 21:01:38
1	2020-02-08 21:02:24
0	2020-02-08 21:02:27
1	2020-02-08 21:04:43
0	2020-02-08 21:04:46
1	2020-02-08 21:05:06
0	2020-02-08 21:05:09

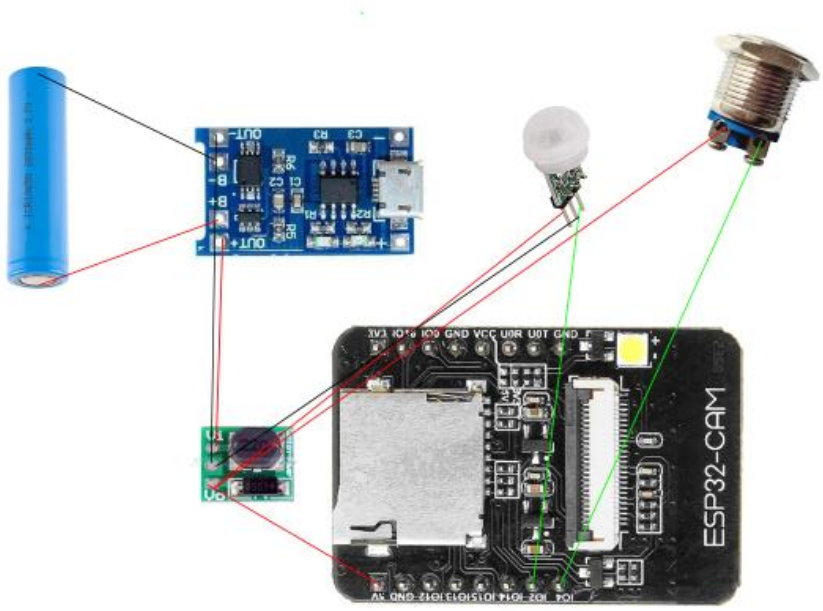
Slika 24: Shema podatkovne baze (vir: Avtor naloge)

6. DELOVANJE IN TESTIRANJE ZVONCA

Zvonec, ki bo v 3D tiskanem ohišju, bo lahko pritrjen po željah uporabnika. Imel bo 2 klica k akcijam, klik gumba in senzor premika. V obeh primerih bo vezje uporabniku poslalo obvestilo na mobilno napravo, ki bo vsebovalo sliko in povezavo do video prenosa v živo. Nato bo poslalo signal, ki bo sporočil sprejemniku, da je nekdo pritisnil tipko in le-ta bo zazvonil.

Celotna naprava bo napajana iz ene 18650 celice, ki bo s pomočjo funkcije »deep sleep« na mikrokrmilniku omogočala daljšo življenjsko dobo baterije. Ob gumbu bosta vgrajeni tudi dve led diodi, ki bosta sporočali status povezave s strežnikom, ter status napajanja. Ko bo naprava izpraznjena, jo bo lahko uporabnik hitro napolnil z uporabo mikro USB priključka, ki bo na spodnji strani naprave.

Shema povezav:



Slika 25: Shema povezav (vir: Avtor naloge)

7. ZAKLJUČEK

Izdelava raziskovalne naloge na področju pametnega doma nam je pomagala k boljšemu razumevanju delovanja mikrokrmilnikov, ter komunikacijskih protokolov med njimi. Spoznali smo tudi alternativne možnosti napajanja, kot je baterija in s tem odkrili pomembnost efektivnosti strojne opreme za daljšo življenjsko dobo izdelkov. Izboljšali smo znanja v programiranju, saj smo združili kar 3 programske jezike, da smo izdelali končni izdelek. Sam izdelek pa bomo lahko v prihodnosti izdelali še za vse avtorje naloge, ter ga testirali doma, v realnih okoliščinah.

8. DRUŽBENA ODGOVORNOST

Naš izdelek bi lahko pripomogel v mnogih gospodinjstvih, ki si želijo »nadgradnje« svojega doma in ne želijo zapraviti večje vsote denarja. Tudi manj tehnično podkovanim uporabnikom bi omogočil, da bi lahko spreminjali vse možne nastavitve brez potrebnega znanja programiranja.

Deloval bo tudi kot varnostni element uporabnika, saj bo ob zaznavi gibanja to sporočil, ter sliko shranil v podatkovno bazo. S tem bi ljudem, ki nimajo možnosti namestitve varnostnega sistema, ter zvonca zaradi omejenega napajanja omogočili oboje v enem izdelku.

9. VIRI

9.1 Knjižni viri

Bervar, G.(2008). C++ NA KOLENIH. 2. posodobljena izdaja. Ljubljana: Študentska založba (Zbirka Scripta)

Fitzgerald, S. in Shiloh, M.(2015). ARDUINO PROJECTS BOOK. 1.izdaja. Torino, Italija (Arduino).

Krkoč, P. (2014). Programirajmo Arduino z lahkoto. 1. izdaja. Ljubljana: Ax Elektronika d.o.o

Valade, J. (2002). PHP & MySql for dummies. 3. izdaja. New Jersey: Wiley Publishing.

9.2 Spletni viri

Autodesk eagle. Pridobljeno iz https://www.autodesk.com/products/eagle/overview, dne 10.1.2020.

TP4056. Pridobljeno iz https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf, dne 12.1.2020.

IFTTT. Pridobljeno iz https://ifttt.com/, dne 10.1.2020.

Google scripts. Pridobljeno iz https://developers.google.com/apps-script, dne 10.1.2020.

Python. Pridobljeno iz https://www.python.org/, dne 10.1.2020.

ESP32 – CAM. Pridobljeno iz https://docs.espressif.com/, dne 22.1.2020

Raspberry PI. Pridobljeno iz https://www.raspberrypi.org/, dne 23.1.2020

Arduino. Pridobljeno iz https://www.arduino.cc/, dne 25.1.2020

PIR senzor. Pridobljeno iz https://en.wikipedia.org/wiki/Passive_infrared_sensor, dne 30.1.2020