Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Calculatoare și Tehnologica Informației

Autobuz Smart

Anul universitar 2021 – 2022, AC, CTI RO, AN 3

Sisteme Incorporate

Echipa:

Baluta Andrei-Viorel

Bancila Emanuel

Ciatlosi Ioan-Adrian

Ciobota Isac-Daniel

Enunt

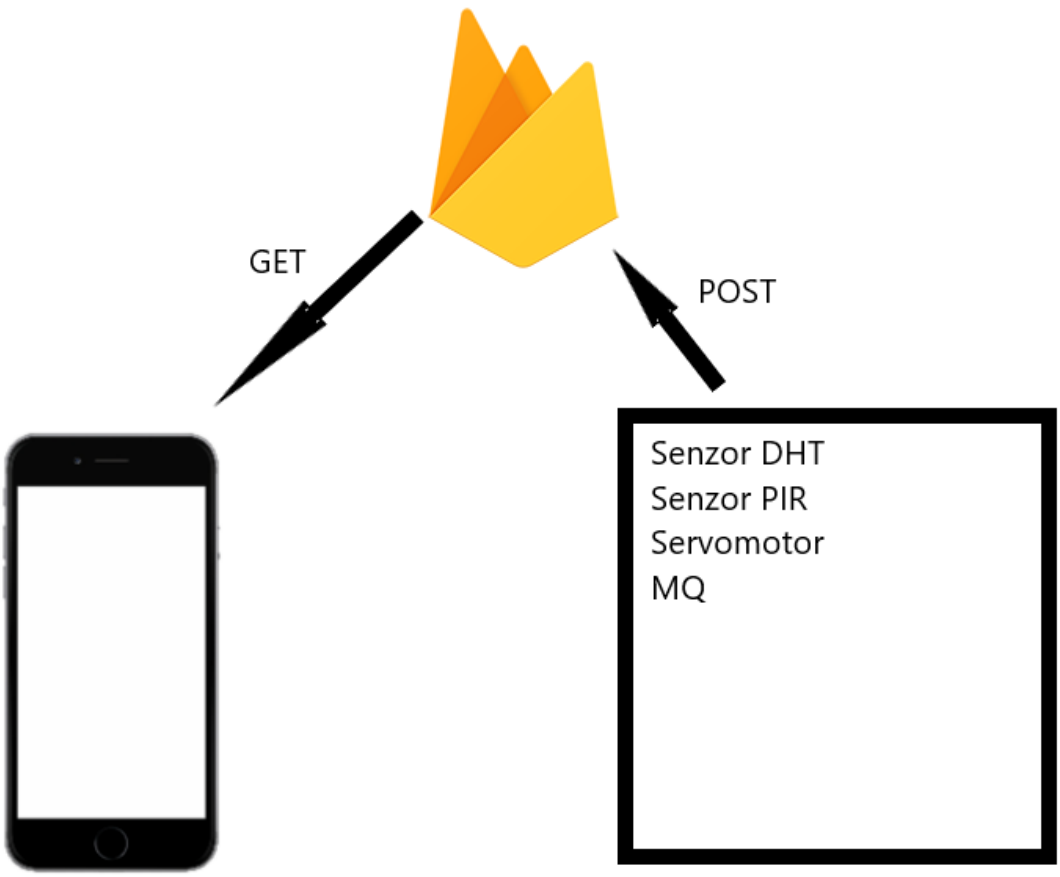
Realizarea unui sistem incorporat Autobuz Inteligent cu urmatoarele componente

- Placa Arduino Uno R3
- Modul GPS/GPRS SIM808
- Senzor de gaz MQ135
- Senzor de temperatura si umiditate DHT11
- Senzori de detectie miscare cu infrarosu PIR
- Servomotor
- Aplicatie mobila

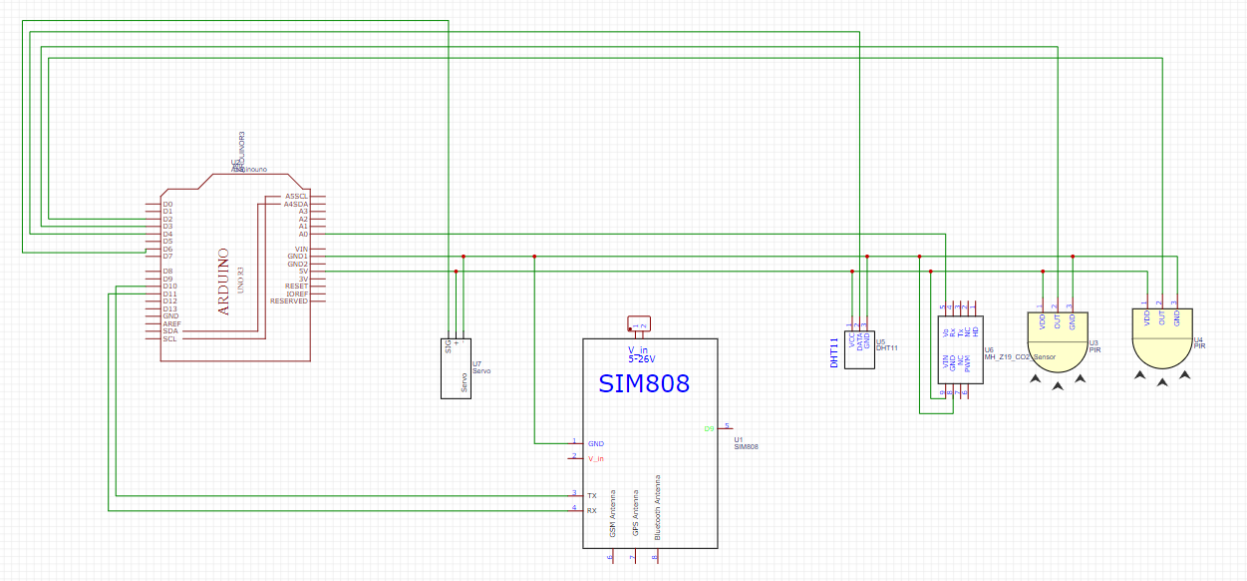
Descriere si scop

Sistemul nostru este creat pentru a oferi informatii despre numarul de persoane din autobuz si calitatea aerului prin intermediul unei aplicatii. De asemenea in functie de temperatura si calitatea aerului din autobuz, sistemul va deschide si inchide geamul cu ajutorul servomotorului

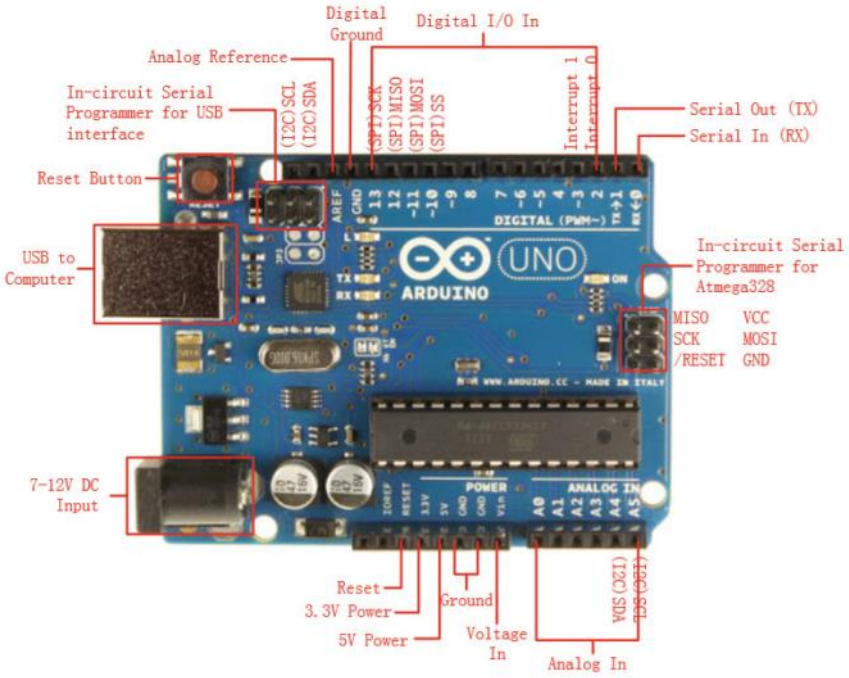
Workflow



Arhitectura



Descrierea placii de dezvoltare Arduino Uno R3



Arduino UNO are următoarele:

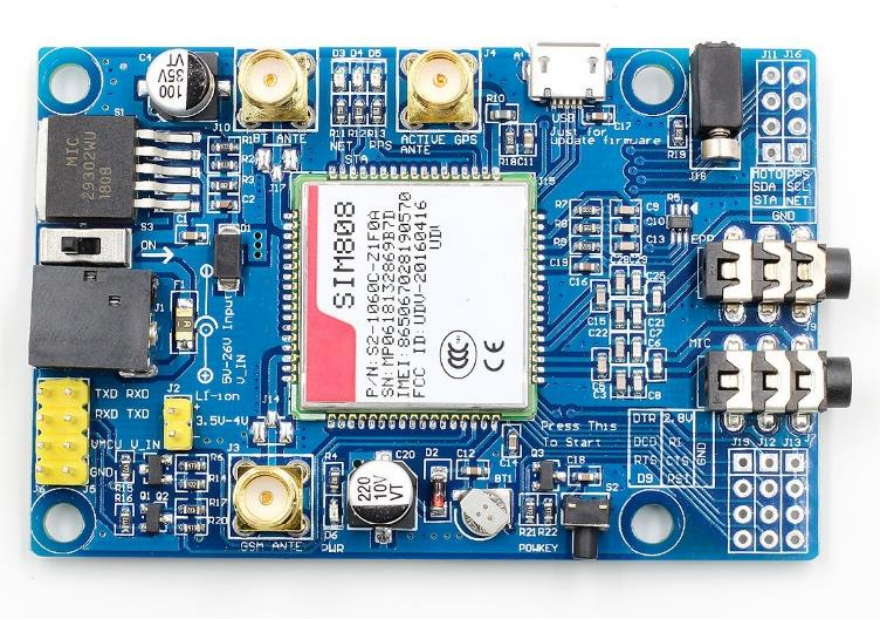
- 14 pini digitali de intrare/ieșire (0 – 13)
- 6 pini analogici (A0 – A5)
- un rezonator ceramic de 16 MHz
- un conector USB
- sursă de alimentare
- un header ICSP
- un buton de reset

Specificațiile plăcii de dezvoltare Arduino UNO sunt următoarele:

- Microcontroler: ATmega328
- Tensiune de lucru: 5V
- Tensiune de intrare (recomandat): 7-12V
- Tensiune de intrare (limita): 6-20V
- Pini digitali: 14 (6 PWM output)
- Pini analogici: 6
- Curent per pin I/O: 40 mA
- Curent 3.3V: 50 mA
- Memorie Flash: 32 KB (ATmega328) 0.5 KB pentru bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

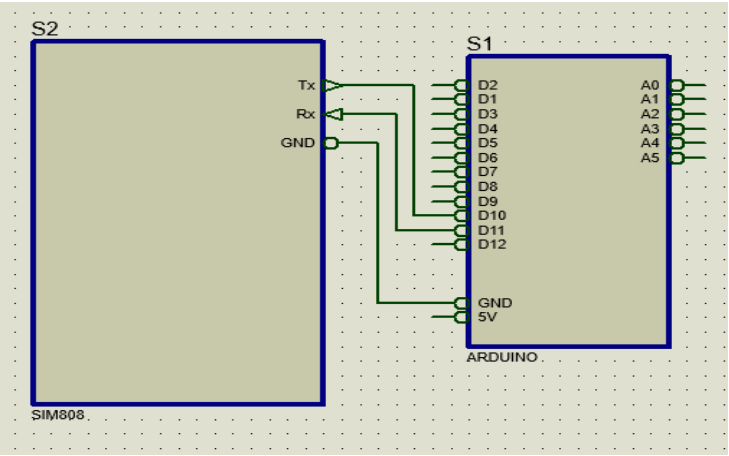
1. Trimiterea datelor la baza de date cu ajutorul SIM808

Descrierea modulului SIM808



SIM808 este un modul GPS si GSM. Modulul are un consum foarte mic de curent, factor important pentru sistemul nostru care trebuie sa fie mobil. El poate fi alimentat de la o baterie Li-Ion reincarcabila care ii asigura un timp indelungat de viata. Modulul este controlat prin comenzi AT primite prin UART.

Conectarea la Arduino



Modulul SIM808 se conecteaza simplu la arduino. Pinul Rx se conecteaza la un pin digital al Arduino ales ca Tx in cod (D11), iar pinul Tx al modulului se conecteaza la pinul Rx ales pe arduino (D10). GND se conecteaza la GND de pe Arduino.

Comenzile pentru GPS si GSM

Pentru GPS comenzile folosite sunt AT+CGPSPWR=1 pentru a porni GPS-ul si AT+CGPSINF=0 pentru a obtine informatiile de la GPS in format simplu.

Pentru GSM comenzile folosite sunt:

- AT+SAPBR=3,1,"CONTYPE","GPRS", pentru a stabili tipul de conexiune
- AT+SAPBR=3,1,"APN","live.vodafone.com" pentru a stabili punctul de acces al cartelei Vodafone
- AT+SAPBR=3,1,"USER","live" pentru a stabili usernameul acceptat de Vodafone pentru conexiunea la internet
- AT+SAPBR=1,1 pentru a porni internetul atunci cand modulul are semnal GSM
- AT+HTTPINIT pentru a initializa procedeul HTTP
- AT+HTTPPARA=”CID”,1 pentru a seta parametrul HTTP “CID” pe 1
- AT+HTTPPARA="URL","http://ec2-18-212-247-203.compute-1.amazonaws.com:5050” pentru a alege url-ul la care se trimite request
- AT+HTTPACTION=0 pentru a face un GET request la url-ul stabilit

Din cauza ca aplicatia noastra foloseste baza de date Firebase, iar aceasa accepta doar request-uri de tip HTTPS, iar modulul SIM808 trimite doar request-uri de tip HTTP si din cauza ca modului SIM808 trimite doar reqeust-uri de tip POST sau GET, am folosit un REST API intermediar pentru a comunica cu baza de date. Modului SIM808 trimite un GET request la server cu parametrii “name” si “value”. Serverul trimite un PATCH request cu acesti parametrii la Firebase.

Serverul

Serverul nostru este o aplicatie Node.js care ruleaza in Amazon Web Services.

```
var express = require('express');
const routes = require('./routes');
var app = express();
const PORT = 5050
app.listen(PORT, function () {
console.log(`Demo project at: ${PORT}!`); });
const { books } = require('./request')
app.use('/', routes);
```

Acesta ruleaza pe portul 5050 al instantei EC2 din AWS si are o singura ruta, cu un singur tip de request.

```
const request = require('request');

exports.get = async (req, res) => {
    const name=req.query.name;
    const val=parseFloat(req.query.value);
request.patch(
    //First parameter API to make post request
    'https://arduino-test-e96d3-default-rtdb.firebaseio.com/arduinoApp.json',
    //The second parameter, DATA which has to be sent to API
        { json: {
                [name]: val
            }
          },

    //The third parameter is a Callback function
    function (error, response, body) {
        if (!error && response.statusCode == 200) {
            console.log(body);
            console.log(response.statusCode);
        }
    }
);
        res.status(200).json(req.body);
};
```

Cand serverul primeste un request de tip GET, acesta trimite la Firebase un request de tip PATCH cu un json care are ca si cheie variabila “name” si ca si valoare, variabila “value”.

Transmiterea datelor la server cu Arduino si SIM808 si preluarea datelor GPS

Arduino comunica cu modulul SIM808 prin biblioteca SoftwareSerial. Pinul 10 este ales ca Rx al Arduino, iar pinul 11 ca Tx. Componentele comunica cu un baud rate de 9600.

```
#include <SoftwareSerial.h>
#include <stdlib.h>
SoftwareSerial sim808(10, 11); // RX, TX
```

In setup:

```
sim808.begin(9600);
```

Pentru primirea datelor de la GPS am implementat o functie care sa trimita comenzile AT+CGPSPWR=1 si AT+CGPSINF=0 la SIM808, sa citeasca raspunsul modulului, sa prelucreze acest raspuns si sa seteze variabilele de latitudine si longitudine.

```
void getGpsData()
{
 sendsim808("AT");
 sendsim808("AT+CGPSPWR=1");
 while(sim808.available())
 sim808.read();
 sim808.println("AT+CGPSINF=0");
 delay(1000);
 char s[300];
 int i=0;
 while(i<200&&sim808.available())
 {s[i]=(char)sim808.read();
 i++;
 delay(10);
 }
 delay(100);
 s[100]='\0';
 char lat[100],lon[100];
 strncpy(lat,s+27,11);
 lat[11]='\0';
 strncpy(lon,s+39,11);
 lon[11]='\0';
 latitude=atof(lat)/(double)100;
 longitude=atof(lon)/(double)100;

 delay(100);
}
```

Functia trimite comenzile si citeste raspunsul SIM808, prelucrand datele pentru a obtine latitudinea si longitudinea.

Pentru trimiterea datelor la server, am implementat 2 functii, una pentru transimetere de date float si una pentru date int:

```
void postNum(char* nume,int val)
{
    char url2[] = "AT+HTTPPARA=\"URL\",\"http://ec2-18-212-247-203.compute-1.amazonaws.com:5050/?name=";
    char url[150];
    strcpy(url,url2);
    Serial.println(nume);
    strcat(url,nume);
    char v[]="&value=";
    Serial.println(v);
    strcat(url,v);
    Serial.println(url);
    char buf[100];
    itoa(val,buf,10);
    buf[9]='\0';
    Serial.println(buf);
    strcat(url,buf);
    strcat(url,"\"");


 sendsim808("AT");
 sendsim808("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
 sendsim808("AT+SAPBR=3,1,\"APN\",\"live.vodafone.com\"");
 sendsim808("AT+SAPBR=3,1,\"USER\",\"live\"");
 sendsim808("AT+SAPBR=1,1\n",3000);
 sendsim808("AT+HTTPINIT");
 sendsim808("AT+HTTPPARA=\"CID\",1");
 sendsim808(url);
 sendsim808("AT+HTTPACTION=0");
}
```

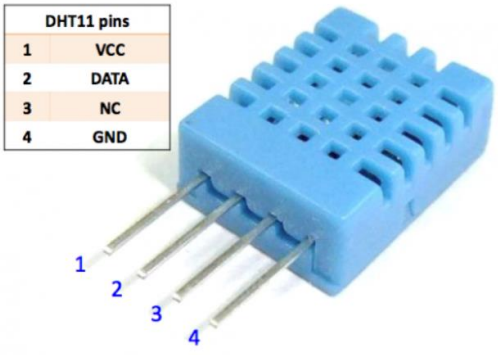
Functia primeste un sir de caractere si un int si creeaza URL-ul cu parametrii corespunzatori, iar apoi trimite un GET request la acesta.

```
void postLL(char* nume,double val)
{

    char url2[] = "AT+HTTPPARA=\"URL\",\"http://ec2-18-212-247-203.compute-1.amazonaws.com:5050/?name=";
    char url[150];
    strcpy(url,url2);
    Serial.println(url);
    strcat(url,nume);
    Serial.println(url);
    char v[]="&value=";
    char buf[100];
    strcat(url,v);
    Serial.println(url);
    int l=val;
    itoa(l,buf,10);
    strcat(url,buf);
    Serial.println(url);
    l=val*100;
    l=l%100;
    char p[]=".";
    strcat(url,p);
    Serial.println(url);
    char buf2[100];
    itoa(l,buf2,10);
    strcat(url,buf2);
    strcat(url,"\"");
    Serial.println(url);

 sendsim808("AT");
 sendsim808("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
 sendsim808("AT+SAPBR=3,1,\"APN\",\"live.vodafone.com\"");
 sendsim808("AT+SAPBR=3,1,\"USER\",\"live\"");
 sendsim808("AT+SAPBR=1,1\n",3000);
 sendsim808("AT+HTTPINIT");
 sendsim808("AT+HTTPPARA=\"CID\",1");
 sendsim808(url);
 sendsim808("AT+HTTPACTION=0");
}
```

Functia primeste un sir de caractere si un float si creeaza URL-ul cu parametrii corespunzatori, iar apoi trimite un GET request la acesta.

2. DHT11

Descriere



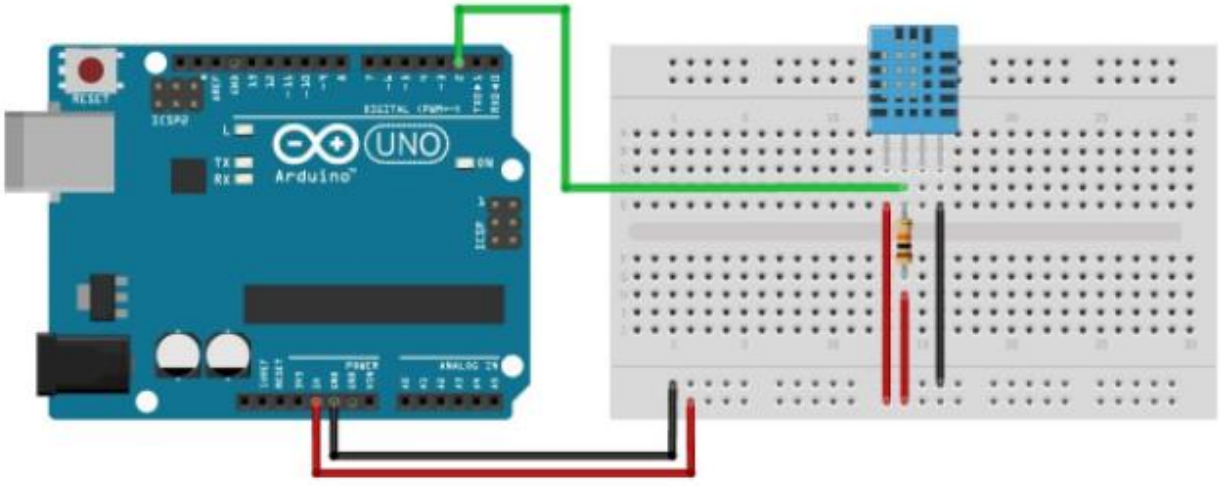
DH11 este un senzor digital de temperatura si umiditate, low-cost. Acesta incorporeaza un senzor de umiditate capacitiv si un termistor, pentru a masura aerul din jur si da un semnal digital pe pinul de date (nu necesita pini de intrare analogici).

Specificații tehnice

Alimentare si I / O: 3 - 5V
Curent maxim: 2.5mA
Pentru gama de umiditate 20-80% are o precizie de 5%
Pentru gama de temperatura 0-50 ° are o precizie de ± 2 ° C
Rata de esantionare de 1 Hz (o data pe secunda)
Dimensiuni: 16mm x 12.6mm x 5.7mm

Conectarea la Arduino

Pin 1 – Vcc, Pin 2 – Digital Pin, Pin 3 – nefolosit, Pin 4 - GND



DHT 11 masoara *umiditatea relativa*. Aceasta reprezinta raportul dintre apa din aer si apa din aer la momentul de saturatie (momentul in care se formeaza roua). Cel din urma, variaza in functie de temperature aerului. Aerul rece poate pastra mai putina apa pana sa ajunga la

saturatie, iar aerul cald poate pastra mai multa apa. Umiditatea relativa se exprima sub forma de procent: la 100% RH se produce condensul, la 0% RH aerul este complet uscat.

$$RH = \left(\frac{\rho_w}{\rho_s}\right) x\ 100\%$$

$$RH: Relative\ Humidity$$
$$\rho_w: Density\ of\ water\ vapor$$
$$\rho_s: Density\ of\ water\ vapor\ at\ saturation$$

Cum masoara umiditatea si temperatura?

DHT11 detecteaza apa masurand rezistenta electric dintre 2 electrozi. Componenta de detectare a umidității este un substrat care reține umiditatea, cu electrozi aplicați pe suprafață sa. Când vaporii de apă sunt absorbiți de substrat, ionii sunt eliberați de substrat, ceea ce crește conductivitatea dintre electrozi. Modificarea rezistenței dintre cei doi electrozi este proporțională cu umiditatea relativă. Umiditatea relativă mai mare scade rezistența dintre electrozi, în timp ce umiditatea relativă mai scăzută crește rezistența dintre electrozi Temperatura de masoara cu ajutorul unui thermistor integrat in circuit.

Cod

Includem libraria DHT.h

```
#include "DHT.h"
```

Asignam pinul

```
#define DHTPIN 2
```

Definim tipul de senzor folosit

```
#define DHTTYPE DHT11
```

In setup() initializam comunicarea dintre Arduino si senzor

```
void setup() {
  dht.begin();
}
```

Citim umiditatea si temperature (Celsius e default)

```
float h = dht.readHumidity();
```

```
float t = dht.readTemperature();
```

Verificam daca a citit correct

```
if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
}
```

Calculam heat index (Fahrenheit e default, setam isFahrenheit = false). Heat index calculeaza temperature perceputa de om (tine cont si de umiditate)

```
float hic = dht.computeHeatIndex(t, h, false);
```

Afisam rezultatele

```
Serial.print(F(" Humidity: "));
Serial.print(h);
Serial.print(F("%  Temperature: "));
Serial.print(t);
Serial.print(F("°C "));
Serial.print(f);
Serial.print(F("°F  Heat index: "));
Serial.print(hic);
Serial.print(F("°C "));
```

3. MG995

Descriere

Servomotoarele sunt folosite în aplicații foarte variate în care este necesară precizia. Servomotoarele sunt alcătuite dintr-un motor de curent continuu, un potențiometru acționat de axul motorului ce măsoară unghiul la care acesta se rotește, un circuit ce compară semnalul provenit de la potențiometru cu comanda primită de la utilizator și un mecanism cu roți dințate ce reduce turația motorului, dar crește cuplul acestuia.

Pozitia axului motorului depinde de duty cycle al semnalului. Daca semnalul este pe high pentru 0.5ms intr-un singur ciclu, axului se muta la 0 grade. Pentru a roti axului la 90 de grade, semnalul trebuie sa fie pe high pentru 1.5ms, iar pentru a-l roti la 180 de grade, semnalul trebuie sa fie pe high pentru 2.5ms.

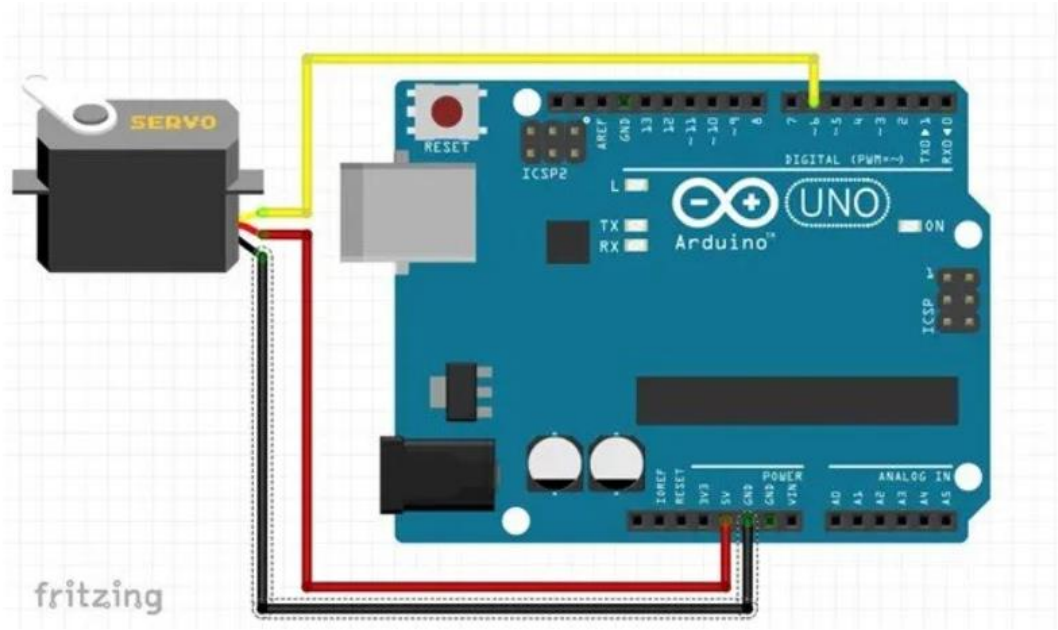
Specificații tehnice

Tensiune de operare: 4.8V - 7.2V;

Curent: 4.8V - 8.8mA, 6V - 9.1mA (în gol);
Curent în stall: 4.8V - 350mA, 6V - 450mA;
Viteză: 0.2 s/60º (4.8V), 0.16 s/60º (6V);
Cuplu: 8.5 kgf·cm (4.8V), 10 kgf·cm (6V).
Dimensiuni: 40.7 x 19.7 x 42.9 mm

Conectarea la Arduino

Brown – GND, Red – Vcc, Orange – Digital Pin



Cod

Includem libraria “Servo.h”

```
#include <Servo.h>
```

Asignam pinul

```
#define Servo_PWM 6
```

Declaram o instanta a lui Servo, cu numele de MG995_Servo

```
PWM signal Servo MG995_Servo;
```

In setup(), atasam instanta noastra la pinul asignat

```
void setup() {
      MG995_Servo.attach(Servo_PWM);
}
```

Folosim write() pentru a roti motorul, detach() pentru a-l oprit si attach() pentru a-l atasa din nou pinului asignat

```
void loop() {
      MG995_Servo.write(0); //Turn clockwise at high speed
      delay(3000);
      MG995_Servo.detach();//Stop.
      delay(2000);
      MG995_Servo.attach(Servo_PWM);//Always use attach function after detach
      to re-connect your servo with the board
      MG995_Servo.write(180);
      delay(3000);
      MG995_Servo.detach();//Stop delay(2000);
      MG995_Servo.attach(Servo_PWM);
}
```

4. Modul PIR

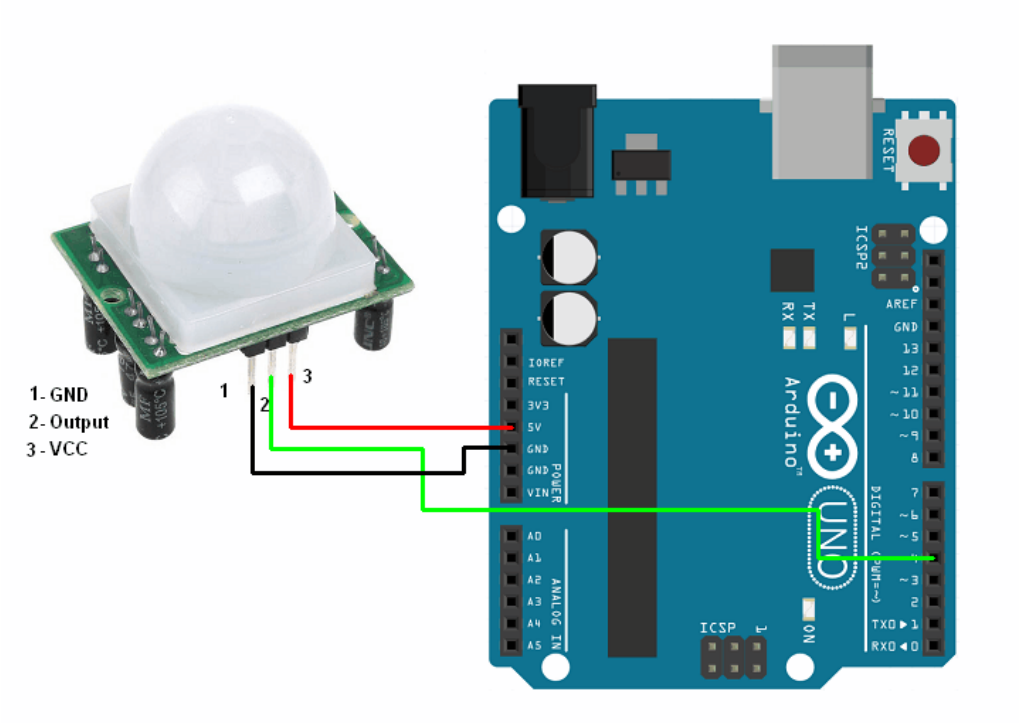
Descriere

Modulul PIR este un senzor pasiv infrarosu (PIR). Acesta poate detecta miscarea bazata pe schimbari in lumina infrarosie din mediul in care este amplasat.

Specificatii tehnice

- Tensiune de operare: 5V - 20V;
- Curent: 65mA;
- Output digital TTL: 3.3V / 0V;
- Distanta pentru detectare: 7 metri (unghi de 120grade)
- Timp de delay: intre 3 si 300 secunde, ajustabil
- Temperatura optima pentru functionare: -15℃ ~ +70℃
- Greutate:6.6g

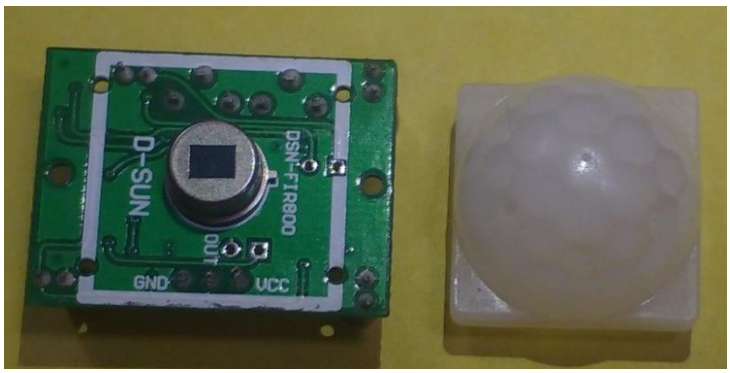
Conectarea la Arduino



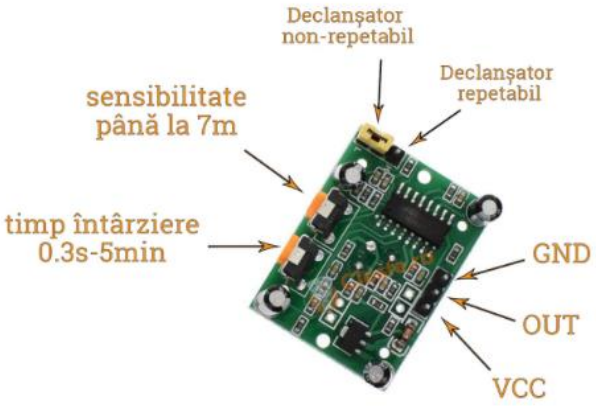
Pin 1 – GND, Pin 2 – Output, Pin 3 – VCC

Cum functioneaza?

Corpurile vii emana energie termica intr-o forma de radiatie infrarosie, cum ar fi animalele sau oamenii. Astfel daca o vietuitoare va intra in raza senzorului PIR, acesta ii va detecta prezenta. PIR nu emite nicio energie pentru detectare, ci detecteaza energia trimisa prin senzorul sau. Acesta este acoperit cu lentile Fresnel, care au rolul de a concentra razele infrarosii primite pe senzorul piroelectric.



Modulul PIR are in alcatuirea sa 2 potentiometre, unul pentru a regla timpul de intarziere, intre 3sec si 5 minute, iar celalalt pentru reglarea sensibilitatii, pana la 7 metri.



Pe langa cei 2 pini pentru GND si VCC, are un pin de iesire ce da un nivel logic ridicat atunci cand un corp este detectat. Declansatoarele au rolul de a seta preferinta, daca senzorul va da nivel logic ridicat pana la terminarea timpului setat, sau pana cand corpul detectat iese din raza senzorului.

Cod

```
Asignam pinii 2 si 3 pentru fiecare senzor
const byte sensorPin1 = 3;
const byte sensorPin2 = 2;
Initializam counterul de persoane inregistrate, state-ul prezent si precedent al fiecarui senzor, si countere pentru numarul de dati in care a fost activ fiecare senzor.
int counter=0;
int state1 = LOW;
int state2 = LOW;
int laststt1=LOW;
int laststt2=LOW;
int state1count=0;
int state2count=0;
Variabila cu rol de a incrementa numarul de oameni
int ok=0;

In Setup, asignam pinii ca si input pentru Arduino, si initializam Serial Monitor
void setup(){
  pinMode(sensorPin1, INPUT);
  pinMode(sensorPin2, INPUT);
  Serial.begin(9600);
}
In Loop, citim starea curenta a fiecarui senzor
void loop(){

  state1=digitalRead(sensorPin1);
  state2=digitalRead(sensorPin2);
```

Daca starea curenta e diferita de cea precedenta, si cea curenta este HIGH, atunci incrementam counterul senzorului 1, si facem la fel si pentru senzorul 2

```
if(state1!=laststate1 && state1==HIGH){
        state1count++;
}
 if(state2!=laststate2 && state2==HIGH){
        state2count++;
 }
```

Daca o stare este mai mare cu 1 fata de cealalta, inseamna ca a trecut un om, deci ok=1. Daca este o diferenta mai mare, sau un om activeaza doar primul senzor, vom egala counterele pentru a nu se crea o diferenta mai mare de 1.

```
 if(state1count==(state2count+1)){
        ok=1;
 } else state1count=state2count;
 if(state2count==(state1count+1)){
        ok=1;
 } else state1count=state2count;
```

Testam ambele cazuri, sa vedem din ce parte a trecut omul, pentru a incrementa sau decrementa counterul de persoane. Obligatoriu ok trebuie sa fie 1, iar apoi verificam daca senzorul 2 a trecut de la high la low in timp ce senzorul 1 este high, ceea ce indica faptul ca o persoana a trecut (ca exemplu presupunem ca senzorul1 este plasat in stanga senzorului 2) de la stanga la dreapta. Celalalt caz verifica opusul, persoana trecand de la dreapta la stanga indicand ca a iesit, deci decrementam. La final ok=0

```
 if(ok==1 && laststate2==LOW &&state2==HIGH && state1==HIGH){
   counter++;
   ok=0;
   delay(300);
   Serial.println(counter);
 }
 else if(ok==1 && laststate1==LOW && state2==HIGH && state1==HIGH){
```

```
    counter--;
    Serial.println(counter);
    ok=0;
    delay(300);
  }
```

La final trebuie sa atribuim starii precedente valoarea starii curente, deoarece aceasta va deveni cea precedenta in urmatorul loop.

```
  laststl=state1;
  laststl=state2;
}
```

5. MQ-135

Descriere

Senzorul MQ-135 este un senzor sensibil la amoniac, benzen, hidrogen, C02 la o concentratie de la 10 la 10000ppm

Specificatii tehnice

Output: analog 0 - 5 V

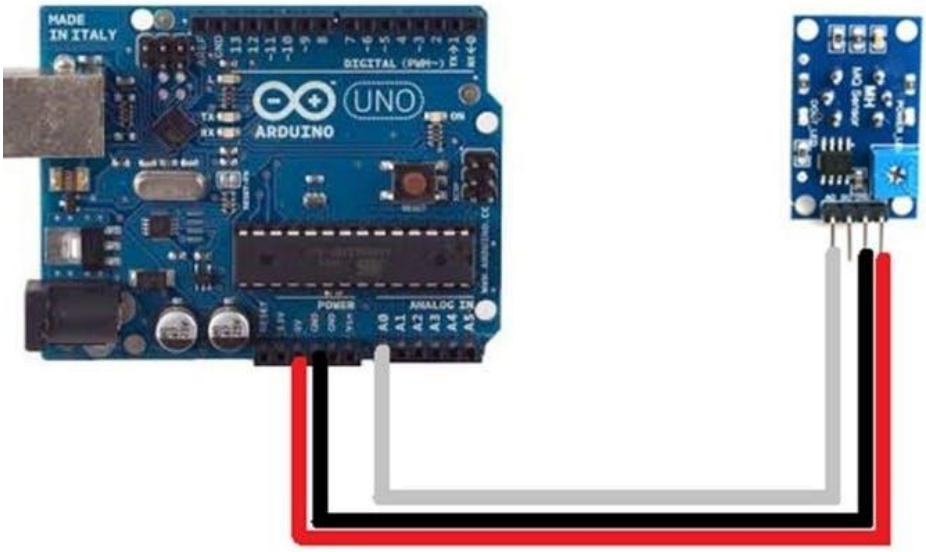
Tensiune de alimentare: 5V

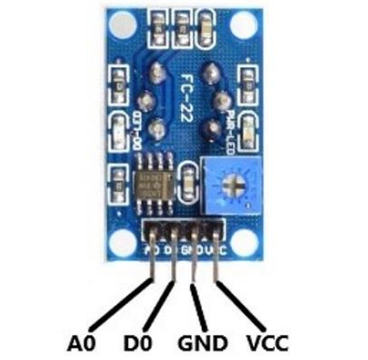
Două semnale de output (analog si TTL)

Tensiunea de ieșire creste cu câte 0.1V pentru fiecare 20ppm

Temp. Utilizare -10℃ - 45℃

Conectarea la Arduino





Cum functioneaza?

Senzorul are o conductivitate scazuta in aerul curat, iar cand gazele mentionate mai sus (amoniac, benzen, hidrogen,CO2) sunt prezente in aer, conductivitatea senzorului este mai mare. Pentru masurarea in PPM(parts-per-million), trebuie folosit pinul analog.

Pentru a fi folosit, acesta trebuie preincalzit 24 de ore. Folosind pinul digital, se introduce senzorul in gazul cautat si senzorul va da nivel logic ridicat. Folosind pinul analog, valuarea returnata va fi direct proportionala cu concentratia de gaz din aerul detectat.

MQ135 foloseste in senzorul sau SnO2 (Tin Dioxide) care are o rezistenta mai mare in aer curat ca si material detector de gaz. Atunci cand exista gaze poluante, rezistenta scade invers proportional. Ca sa putem masura PPM trebuie sa ne uitam la urmatorul grafic.

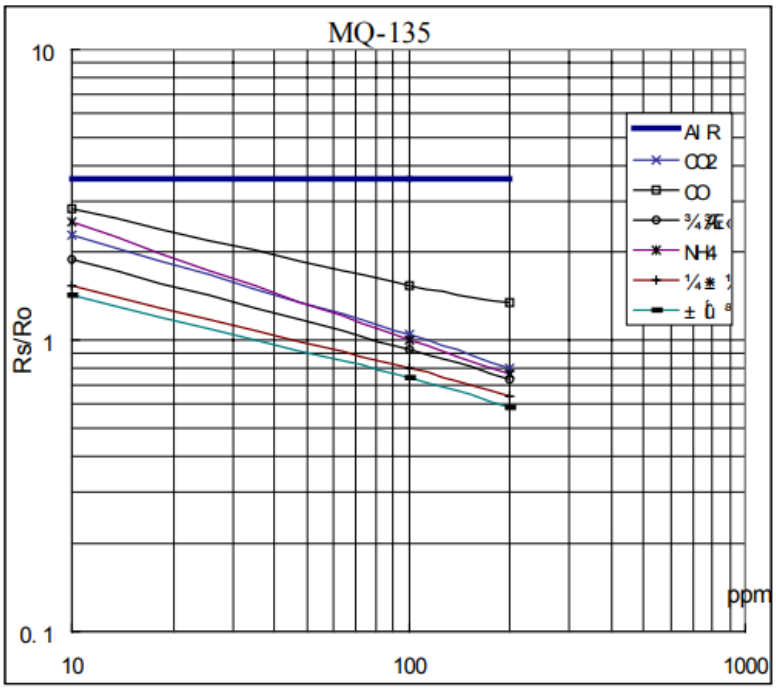


Fig.3 is shows the typical sensitivity characteristics of the MQ-135 for several gases.
in their: Temp: 20
Humidity: 65%
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of NH_3 in the clean air.
Rs:sensor resistance at various concentrations of gases.

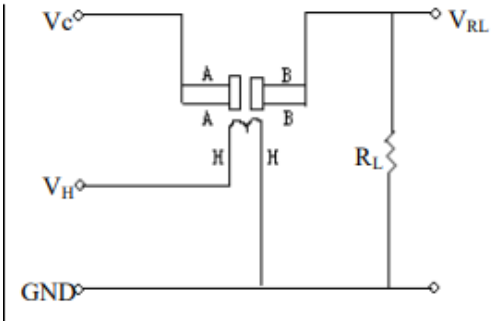
Avem sensibilitatea mq135 pentru diferite gaze in 20 grade Celsius, 65% umiditate,21% Concentratie O2.

R0 este rezistenta senzorului la 100ppm de NH3 in aer curat.

RS este rezistenta senzorului in concentratii diferite de gaze.

Prima oara ar trebui sa calibram senzorul prin aflarea valorilor R0 in aer liber, si apoi sa gasim RS folosind formula :

Resistance of sensor(Rs): Rs=(Vc/VRL-1)×RL



Odata ce am calculat RS si R0 putem gasi ratia RS/RO iar apoi putem afla PPM pentru gazul respectiv.

Cod

Includem libraria senzorului, definim RLOAD de pe senzor, RZERO dupa calibrare si pinul analog.

```
#include "MQ135.h"
#define RLOAD 102.0
#define RZERO 76.63
int sensorIn = A0;
```

Functia pentru aflarea ppm unde folosim constructorul din librarie si apelam functia

```
float ppm;
void MQ(){
  MQ135 gasSensor = MQ135(A0);
   ppm = gasSensor.getPPM();
  Serial.print ("ppm: ");
  Serial.println (ppm);
}
```

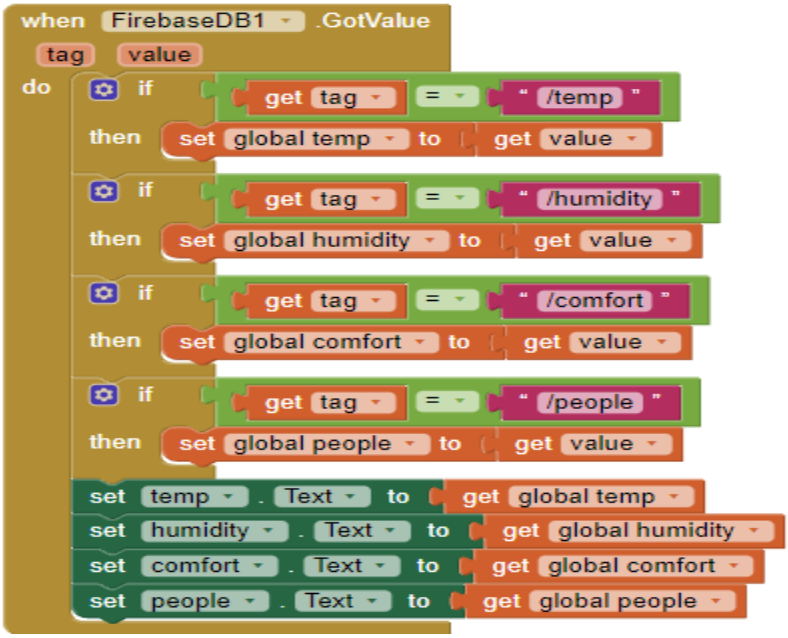
6. Aplicatia mobila

Pentru ca utilizatorul sa poata vedea datele primite de la sistemul incorporat cu Arduino, am creeat o aplicatie mobila Android. Cu ajutorul serverului, se transmit datele de la sistem catre baza de date din Firebase, iar mai apoi aplicatia mobila preia aceste date si le afiseaza / prelucreaza.

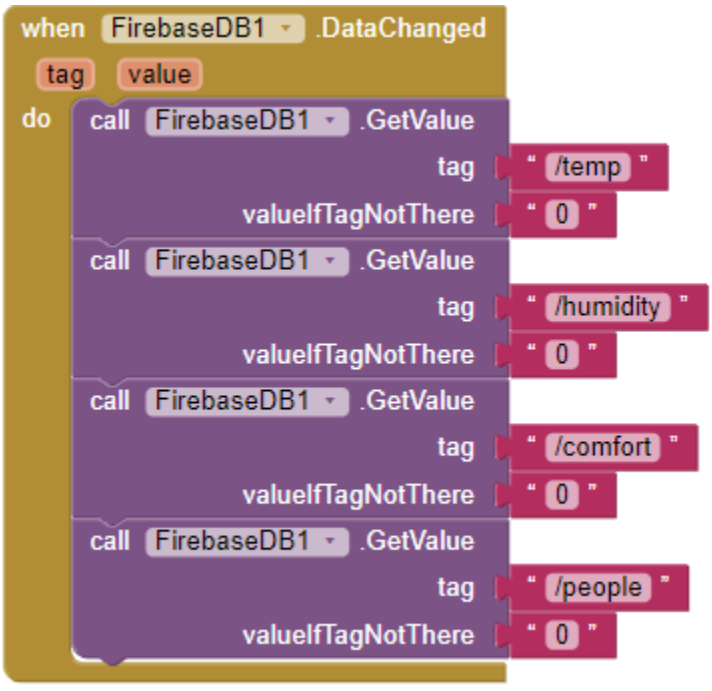
Aplicatia a fost dezvoltata cu ajutorul MIT App Inventor (https://appinventor.mit.edu), pe baza de block-uri pentru functionalitate.

Exista 2 ecrane, pe primul avem afisate temperatura, umiditatea, gradul de comfort si numarul de oameni. De asemenea, avem un buton care duce spre cel de al doilea ecran, pe care scrie “Vezi pozitia autobuzului pe harta”.

Mai intai se initializeaza 4 variabile globale reprezentand datele ce trebuiesc afisate. Mai apoi exista un block care detecteaza cand exista valori in Firebase, le ia pe baza tag-ului si seteaza variabilele globale si text-ul de pe ecran pe valorile respective.

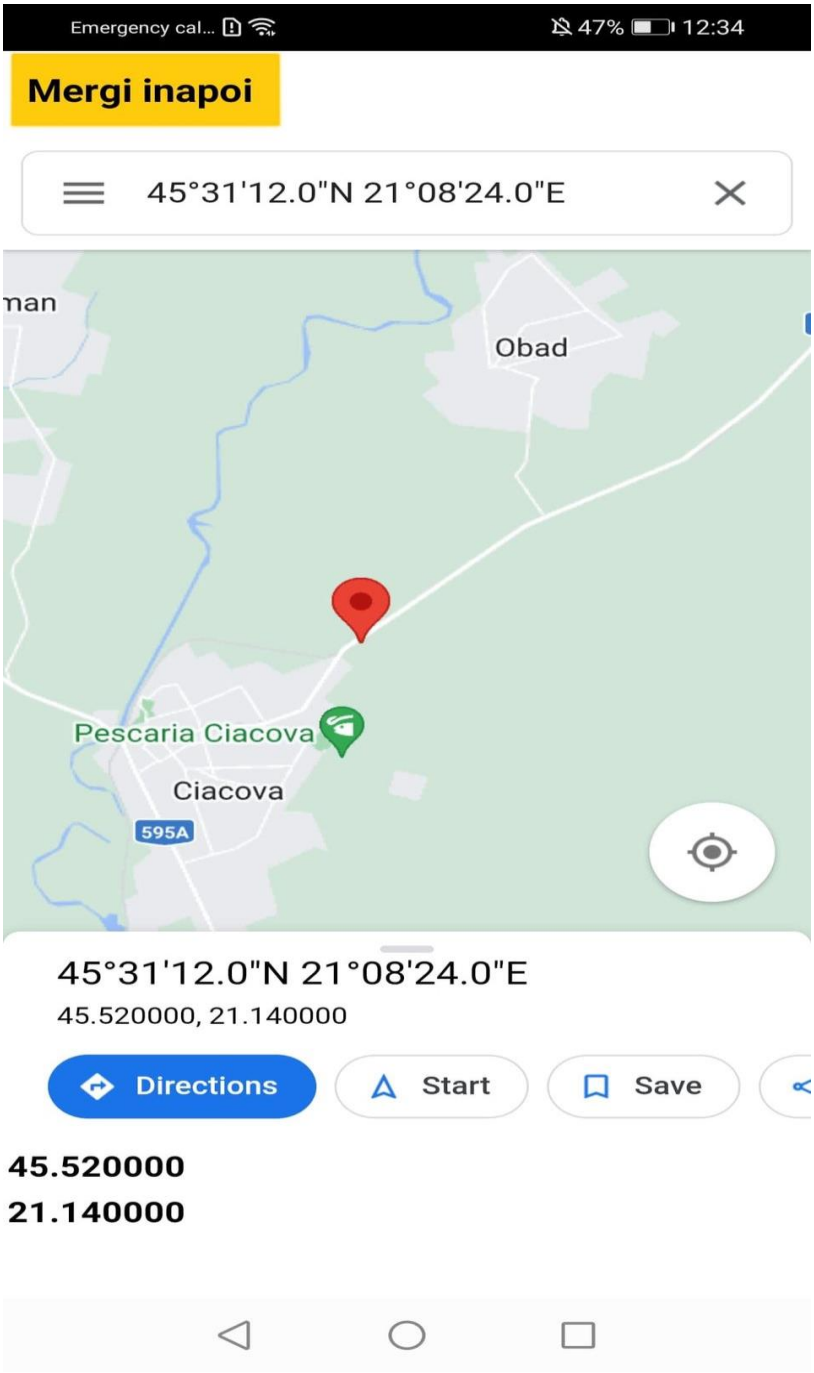


De asemenea exista un block care detecteaza daca a aparut o schimbare a vreunei valori in Firebase, si seteaza din nou valorile corespunzatoare in acelasi mod.

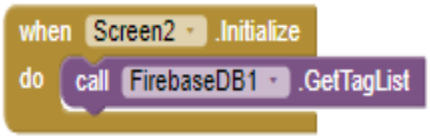


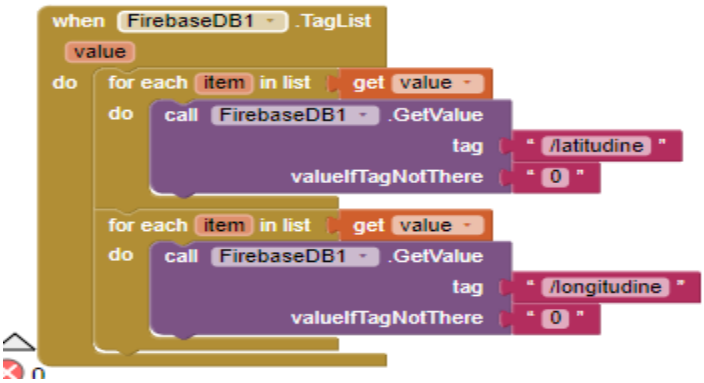
Ultimul block schimba ecranul 1 in ecranul 2 cand este apasat butonul.

Pe al doilea ecran avem o harta, de la Google Maps, pe care se da cautare automat catre latitudinea si longitudinea din Firebase, si se arata un marker catre acea locatie specifica, la care se afla autobuzul. De asemenea este si un buton care duce spre primul ecran, si 2 label-uri unde este afisata latitudinea si longitudinea direct.

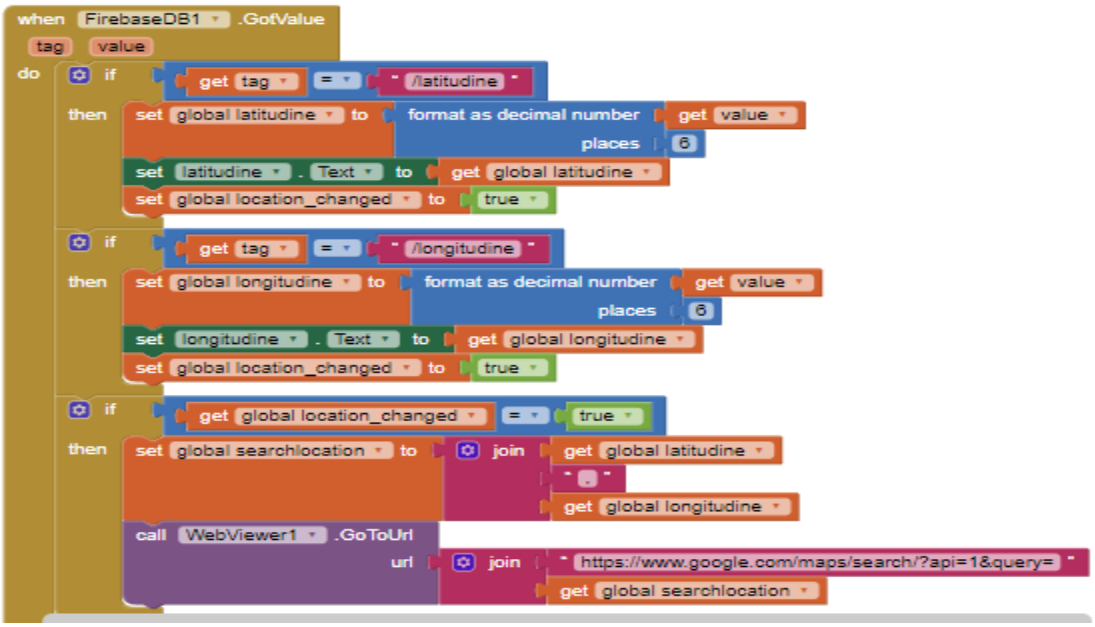


La inceput se initializeaza variabilele globale, si se iau valorile initiale pentru latitudine si longitudine din Firebase.

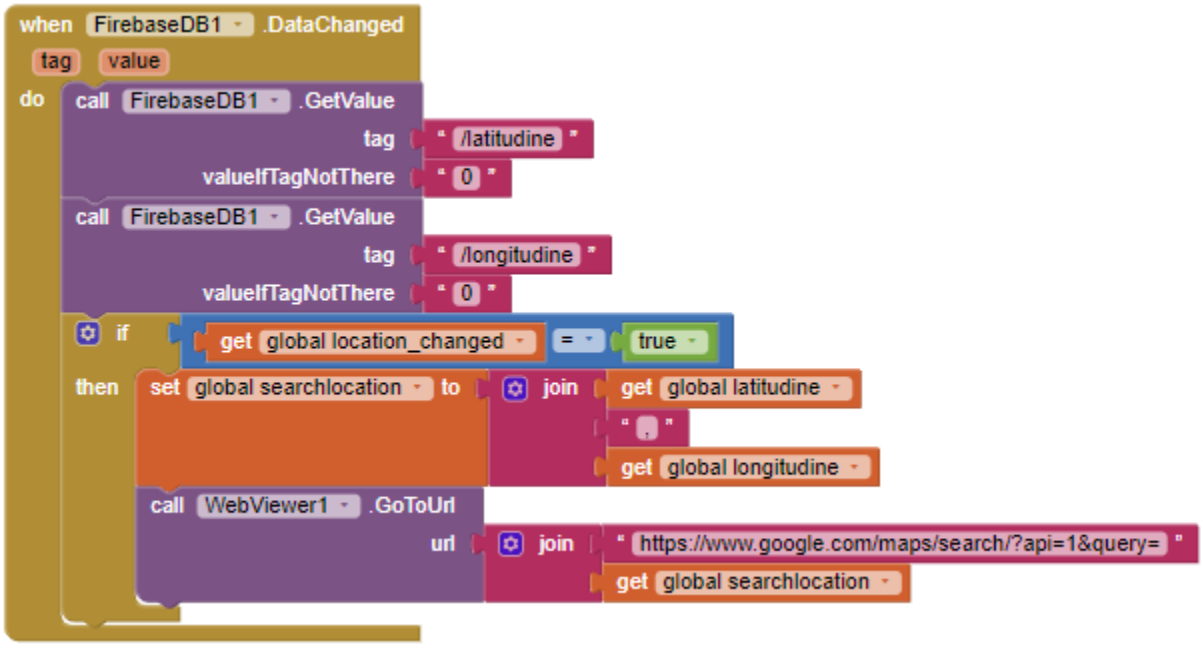




Avem un block care daca exista valorile in Firebase, pune text-ul labelurilor pe valorile corespunzatoare, formatandu-l intr-o valoare cu 6 zecimale, si se pune si o variabile location_changed pe true, care ajuta cand vom afisa locatia pe harta. Daca acest location_changed este true, atunci in variabila searchlocation punem locatia de tip " latitudine, longitudine " , si prin WebViewer navigam catre un url de pe google maps, care va cauta pozitia dupa aceste coordinate, si va fixa un marker pe harta.



In mod similar avem un block care atunci cand se schimba coordonatele din Firebase, le seteaza corespunzator, si schimba locatia pe harta.



Ultimul block schimba ecranul 2 in ecranul 1 cand este apasat butonul.

7. Bibliografie

- https://cleste.ro/modul-senzor-detector-gaz-mq-135.html
- https://ro.onetransistor.eu/2017/12/indice-confort-termic-dht11-arduino.html?fbclid=IwAR1zKfSGOmDik6TlXJggdnESaz2fv0cEI3gWeb2pK19Kyjt0ftQ81-lA66M
- https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor
- https://community.appinventor.mit.edu/t/intro-google-maps-with-app-inventor/51518
- https://microcontrollerslab.com/mg995-servo-motor-pinout-interfacing-with-arduino-features-examples/
- https://www.arduino.cc/en/uploads/Main/Quectel_M10_AT_commands.pdf
- https://firebase.google.com/docs/database
- https://expressjs.com/en/starter/hello-world.html