Iordache Ionut-Iulian

Project Presentation

Home Assist Station

Introducere

Prezentarea pe scurt a proiectului:

- Panoul ofera informatii relevante cu privire la temperatura, umiditate, calitatea aerului si distanta la care se situeaza un obiect fata de el.De asemenea beneficiaza de un amplificator lm386 si un speaker de 6ohm pentru a reda un mesaj de 'Bun venit!...' si poate avea deasemenea conectat un aparat care functioneaza la 220V care poate fii activat/dezactivat prin intermediul aplicatiei Web si Mobile, Blynk.
- Are ca scop monitorizarea calitatii aerului si a proprietatilor acestuia in incaperea in care se regaseste si transmite toate aceste date catre un server care le pune la indemana utilizatorului prin intermediul unei interfete atat pe web cat si prin intermediul unei aplicatii instalata in prealabil pe smartphone(Blynk).
- Intrucat panoul se vrea a fii situata cat mai aproape de biroul unde imi desfasor activitatea de zii cu zii, ideea a pornit de la dorinta de a cunoaste proprietatile mediului in care ma aflu in mod constant pentru a putea sesiza cat mai propmt schimbarile ce pot aparea asupra calitatii aerului. Am adus acestui panou imbunatatiri precum afisarea valorilor, optiunea de a reda audio folosind mufa jack cat si controlul asupra unei prize prin intermediul aplicatiei pentru a beneficia la maximum de interfata pusa la dispozitie de server-ul Blynk.
- Aceasta statie isi gaseste utilitatea prin faptul ca ofera informatii esentiale despre o resursa vitala omului si anume aerul de care avem atat de multa nevoie pentru a mentine viata. De altfel poate reda sunete de la orice device si controla diferite aparate conectate la priza remote pe care o pune la dispozitie.

Descriere generală

Intreg proiectul consta in transmiterea datelor preluate de la senzorii de gaz, temperatura si umiditate, si cel de distanta catre placuta de Arduino. Concomitent valorile apar pe display-ul care este de asemenea conectat la Arduino (prin I2C). Initial prin SPI se realizeaza de la modulul SDcard preluarea mesajului de bun venit care va fii redat catre utilizator prin intermediul circuitului amplificator(am utilizat amplificatorul operational lm3386) care are atasat un speaker de 6 ohm. Pentru a putea face datele vizibile in Internet si a marii aplicabilitatea acestui sistem am utilizat

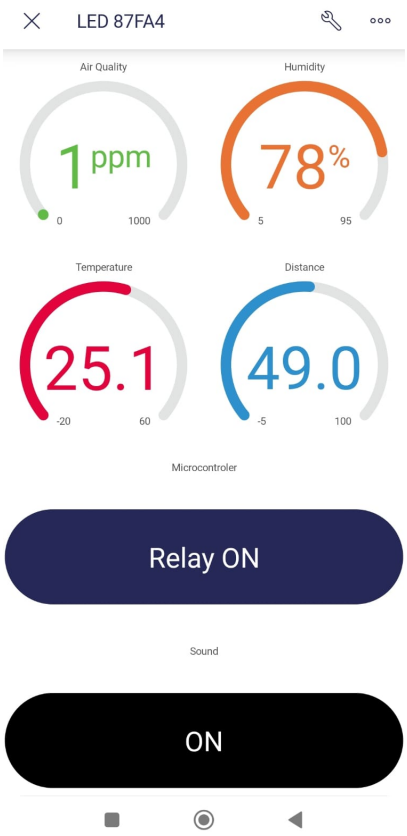
ESP8266 pentru a transmite datele colectate de la senzori catre serverul Blynk. Tot folosind ESP8266 am decis sa activez folosind un releu cu 2 canale, doar atunci cand am nevoie, atat circuitul pentru speaker cat si circuitul care are atasata o priza(220V) pentru diverse aplicatii ce ar necesita o interventie de la distanta(spre exemplu putem conecta aerul conditionat la aceasta priza si sa il activam inainte de a sosii acasa). Pentru a putea observa toate aceste valori si a controla cele 2 canale ale releului, folosind Blynk, am creat o interfata intuitiva care afiseaza informatiile preluate de la senzori si permite in acelasi timp controlul releelor. Interfata Web cat si cea destinata aplicatiei Mobile sunt vizibile mai jos.

Schemă bloc:

Schemă electrica:

Interfata Web:

Interfata Mobile:



Hardware Design

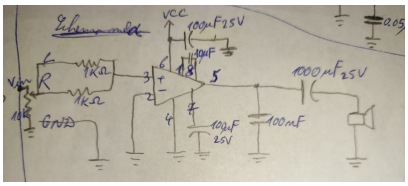
Listă de piese:

- Arduino Uno
- ESP8266 NodeMCU
- placa prototipare
- fire de legatura
- modul senzor de gaz (mq135)
- senzor de temperatura si umiditate (dht11 4pins)

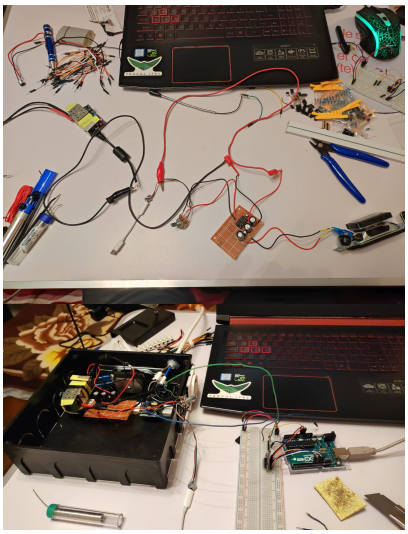
- modul LCD 1602 & I2C
- cablaj de test 70 x 90 mm
- senzor de distanta HC-SR04
- Releu 2 canale
- modul SDCard
- sursa de tensiune 12V
- sursa de tensiune 5v
- priza
- speaker 6 ohm
- rezistente
- condensatori
- amplificator operational lm386
- mufa jack 3.5 mm

Scheme electrice:

- Datasheet ATmega328P
- Arduino UNO pinout
- nodemcu-esp8266-pinout.jpg ← pinout NODEMCU esp8266
- External Link ← amlificator lm386 circuit nemodificat
- In imaginea de mai jos este schema amplificatorului modificata de mine cu potentiometru de 10k si condensatori de valori putin diferite care atenueaza mai bine zgomotul decat valorile condensatorilor prezentati in datasheet:



Rezultatele partiale si finale:



Software Design

Descrierea software utilizat:

- mediu de dezvoltare este Arduino 1.8.13
- pentru a reprezenta schema electrica am folosit EAGLE
- pentru schema bloc am folosit draw.io
- librăriile utilizate: SD.h, TMRpcm.h, SPI.h, DHT.h, Wire.h, LiquidCrystal_I2C.h, SoftwareSerial.h, BlynkEdgent.h

Descrierea codului aplicaţiei (firmware):

- Arduino:
  - Pentru a putea citii valorile senzorilor am declarat pinii 4 si 2 drept trigger(OUTPUT) si echo(INPUT) pentru senzorul ultrasonic. De asemenea am folosit pinul A0 pentru a prelua date de la modulul senzorului de gaz iar pinul A1 pentru a prelua date de la senzorul de temperatura si umiditate.
  - Pentru a putea reda sunetul pe speaker am folosit ca si OUTPUT pinul 9(PB1) impreuna cu functiile din biblioteca TMRpcm.h si SD.h. Comunicarea SPI dintre modulul SDCard si arduino a avut pinul 10(PB2) drept pin de ChipSelect.
  - In functia de setup am apelaat functia begin() din libraria DHT.h pentru a activa citirea valorilor de pe senzorul de temperatura si umiditate. De asemenea am intializat comunicarea cu LCD1602

prin intermediul I2C folosind functia init() si am activat lumina de fundal a display-ului. Pentru a putea reda mesajul de inceput am setat pinul speaker-ului, volumul de redare si fisierul din care se va prelua sunetul, dupa care am afisat un mesaj “Welcome Home SIR” nu inainte de a astepta 22s deoarece aveam diverse erori datorate bibliotecii TMRpcm.h daca nu asteptam ca mesajul audio redat sa se incheie.
    - In functia de loop am receptionat valorile senzorilor pe care le-am afisat pe ecran iar ulterior am format un String cu aceste valori pentru a putea trimtie acest string catre esp8266 prin intermediul comenzii espSerial.println(). Dupa aceasta reinitializam string-ul la valoarea ””.
- ESP 8266:
    - Blynk pune la dispozitia celor care doresc sa interactioneze cu serverul lor mai multe fisiere header alaturi de fisierul principal cu extensia .ino pentru a facilita dezvoltarea de cod mai rapida. Asadar partea dezvoltata de mine se afla in fisierul esp8266Side.ino iar in BlynkEdgent.h am facut doar o mica modificare adaugand un timer BlynkTimer pentru a nu transmite continuu date catre server. Folosind functia sendSensorData() am putut transmite catre server valorile preluate din string si converite la tipurile lor initiale. In ceea ce priveste controlul celor 2 relee am folosit apeluri ale functiei BLYNK_WRITE(virtualPin) puse la dispozitie de interfata Blynk.
    - In functia de setup() am realizat conectarea la server si interfata web si mobile, urmand sa specific intervalul la care sa se faca transmisia de date la server(1s).
    - In functia loop() am citit datele venite de la Arduino caracter cu caracter si am construit valorile initiale citite de senzori. In continuare realizam de fiecare data conectarea catre server cu asteptarea aferenta inainte de a incepe o noua transmisie catre server. Functia getValue() prelua stringul meu, separatorul care am ales sa fie virgula, si indexul pentru valoarea pe care o doream din string, valoare aflata intre 2 virgule consecutive.

```
// Arduino Side Code

#include <SD.h>                         // need to include the SD library
#include <TMRpcm.h>                     // also need to include this library
(used to create instance of TMRpcm)
#include <SPI.h>                        // and this one too for SPI protocol
#include "DHT.h"                        // temperature and humidity sensor
library
#include <Wire.h>                       // wire library
#include <LiquidCrystal_I2C.h>          // liquid crystal with I2C library
#include <SoftwareSerial.h>             // used to transmit data to ESP8266

#define ultrasoundTrigger 4             // pin 4 used as output with
ultrasonic sensor
#define ultrasoundEcho 2                // pin 2 used as input with ultrasonic
sensor
#define smokeA0 A0                      // pin A0 used as input with
air-quality gas sensor
#define DHTPIN A1                       // pin A1 used as input with
temp&hum(DHT11) sensor
#define DHTTYPE DHT11                   // sensor type = DHT11
#define SD_ChipSelectPin 10             // pin 10 used as Chip Select - CS
with SD module
#define SPEAKER_PIN 9                   // pin 9 used as speaker
```

```
long duration;                          // variable for the duration of sound
wave travel
float ultrasoundDistance;               // variable for the distance
measurement
String strArduinoToESP;                 // string to transmit from Arduino to
ESP8266 NODEMCU

DHT dht(DHTPIN, DHTTYPE);               // initialize DHT11 sensor
LiquidCrystal_I2C lcd(0x27, 16, 2);     // set the LCD address to 0x27 for a
16 chars and 2 line display
TMRpcm tmrpcm;                          // used by SD card to play sound
SoftwareSerial espSerial(5, 6);         // RX & TX on pins 5 & 6


void setup() {
  Serial.begin(9600);
  espSerial.begin(9600);//TODO

  // Ultrasonic sensor logic
  // Serial.println("Ultrasonic Sensor HC-SR04 Test");
  pinMode(ultrasoundTrigger, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(ultrasoundEcho, INPUT); // Sets the echoPin as an INPUT

  // Smoke sensor logic
  //Serial.println(F("Smoke sensor test!"));
  pinMode(smokeA0, INPUT);

  // DHT11 sensor logic
  //Serial.println(F("DHT11 sensor test!"));
  dht.begin();

  // LCD&I2C logic
  lcd.init();                           // initialize the lcd
  lcd.backlight();                      // Turn on backlight

  // SDcard logic
  tmrpcm.speakerPin = SPEAKER_PIN;      // speaker ouput pin 9
  if(!SD.begin(SD_ChipSelectPin)) {     // check connection and return proper
message
  // Serial.println("SD fail");
    return;
  }
  tmrpcm.setVolume(6);                  // set output volume on pin selected
  tmrpcm.play("test.wav");              // play sound from file
  delay(22000);

  // Welcome message
  lcd.setCursor(0, 0);                  // set cursor at top left
  lcd.print("  Welcome Home  ");        // print string
  lcd.setCursor(0, 1);                  // set cursor at top left
```

```
  lcd.print("       SIR      ");       // print string
  delay(3000);
}

void loop() {
  delay(1500);                          // process to repeat every 3s (1.5s +
1.5s underneath)
  // Serial.println("=========");
  // Read from smoke sensor
  int smokeVal = analogRead(smokeA0);
  // Serial.print("Pin A0 smoke: ");
  // Serial.print(smokeVal);
  // Serial.println(" ppm");

  // Read from temp&hum(DHT11) sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // Serial.print(F("Humidity: "));
  // Serial.print(h);
  // Serial.print(F(" % | Temperature: "));
  // Serial.print(t);
  // Serial.println(F(" C"));

  // Clears the trigPin condition
  digitalWrite(ultrasoundTrigger, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(ultrasoundTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(ultrasoundTrigger, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(ultrasoundEcho, HIGH);
  delay(1000);
  // Calculating the distance
  ultrasoundDistance = duration * 0.034 / 2; // Speed of sound wave divided
by 2 (go and back)
  // Displays the distance on the Serial Monitor
  // Serial.print("Distance: ");
  // Serial.print(distance);
  // Serial.println(" cm");

  // Print on LCD
  lcd.setCursor(0, 0);                  // set cursor at top left
  lcd.print("Hum: ");                   // print string
  lcd.print(h);                         // humidity value
  lcd.print(" %    ");                  // unit
  // ---
  lcd.setCursor(0, 1);                  // set cursor at left space on 2nd
row
  lcd.print("Temp: ");                  // print string
  lcd.print(t);                         // temperature value
```

```
  lcd.print(" C   ");                  // unit
  // ---
  delay(1500);
  lcd.setCursor(0, 0);                 // set cursor at top left
  lcd.print("Gas val: ");              // print string
  lcd.print(smokeVal);                 // gas value
  lcd.print(" ppm");                   // unit
  // ---
  lcd.setCursor(0, 1);                 // set cursor at left space on 2nd
row
  lcd.print("Dist: ");                 // print string
  lcd.print(ultrasoundDistance);       // distance value
  lcd.print(" cm");                    // unit

  // Send data to ESP as a string
  strArduinoToESP = strArduinoToESP + smokeVal + "," + h + "," + t + "," +
ultrasoundDistance; //TODO am h si t ca float si in ex sunt int-uri
  // Serial.println(strArduinoToESP);
  espSerial.println(strArduinoToESP);  // transmit string to ESP8266

  strArduinoToESP = "";                // reinitialize string value

}
```

```
// ESP8266 Side Code

#define BLYNK_TEMPLATE_ID "TMPL3SY-JM6J"
#define BLYNK_DEVICE_NAME "Blynk LED"
#define BLYNK_AUTH_TOKEN "RkiSi9tnBfPHmic1gaBwiC9upodMfHy1"

#define BLYNK_FIRMWARE_VERSION        "0.1.0"

#define BLYNK_PRINT Serial
//#define BLYNK_DEBUG

#define APP_DEBUG

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_SPARKFUN_BLYNK_BOARD
//#define USE_NODE_MCU_BOARD
//#define USE_WITTY_CLOUD_BOARD
//#define USE_WEMOS_D1_MINI

#include "BlynkEdgent.h"
#include <SoftwareSerial.h>

String strArduinoToESP;               // string received from Arduino to
ESP8266 NODEMCU
int smokeVal;
float humVal, tempVal, ultrasoundVal;
```

```
char rdata;                             // received character

const int motorspeed_pin = D1;
const int DIRA = D2;
const int DIRB = D7;

BLYNK_WRITE(V0)
{
  int pinValue = param.asInt();
  digitalWrite(D5, pinValue);
}

BLYNK_WRITE(V5)
{
  int pinValue = param.asInt();
  digitalWrite(D6, pinValue);
}

BLYNK_WRITE(V7)
{
  int pinValue = param.asInt();
  if (pinValue == 1) {
    digitalWrite(motorspeed_pin, HIGH);
    digitalWrite(DIRA, HIGH);
    digitalWrite(DIRB, LOW);
  } else if (pinValue == 0) {
    digitalWrite(motorspeed_pin, LOW);
    digitalWrite(DIRA, LOW);
    digitalWrite(DIRB, LOW);
  }
}

void sendSensorsData() {
  Blynk.virtualWrite(V1, smokeVal);
  delay(200);
  Blynk.virtualWrite(V2, humVal);
  delay(200);
  Blynk.virtualWrite(V3, tempVal);
  delay(200);
  Blynk.virtualWrite(V4, ultrasoundVal);
  delay(200);
}

void setup()
{
  Serial.begin(9600);
  delay(100);
  pinMode(D5, OUTPUT);
  pinMode(D6, OUTPUT);
  pinMode(motorspeed_pin, OUTPUT);
  pinMode(DIRA, OUTPUT);
```

```
  pinMode(DIRB, OUTPUT);

  BlynkEdgent.begin();
  delay(2000);
  edgentTimer.setInterval(1000L, sendSensorsData);
}

void loop() {
  if (Serial.available() > 0 ) {
    rdata = Serial.read();
    strArduinoToESP = strArduinoToESP + rdata;
    // Serial.print(rdata);
    if ( rdata == '\n') {
      // Serial.println(strArduinoToESP);

      String l = getValue(strArduinoToESP, ',', 0);
      String m = getValue(strArduinoToESP, ',', 1);
      String n = getValue(strArduinoToESP, ',', 2);
      String p = getValue(strArduinoToESP, ',', 3);

      smokeVal = l.toInt();
      humVal = m.toFloat();
      tempVal = n.toFloat();
      ultrasoundVal = p.toFloat();

      // Serial.print("val: ");
      // Serial.print(smokeVal);
      // Serial.print(" | val: ");
      // Serial.print(humVal);
      // Serial.print(" | val: ");
      // Serial.print(tempVal);
      // Serial.print(" | val: ");
      // Serial.print(ultrasoundVal);
      // Serial.print(" | val: ");
      // Serial.println(" |");

      BlynkEdgent.run();
      delay(1000);
      edgentTimer.run();              // Initiates SimpleTimer
      delay(1000);

      strArduinoToESP = "";
    }
  }
}

String getValue(String data, char separator, int index)
{
    int found = 0;
    int strIndex[] = { 0, -1 };
```

```
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Rezultate Obţinute

Concluzii

Cea mai smechera chestie pe care am facut-o, deocamdata! Concluzii si ce am obtinut poate fii vazut accesand link-ul urmator. Project Presentation

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

21.04.2022 : Finalizare alegere tema proiect si creare pagina de wiki cu informatii aferente acestuia.
07.05.2022 : Modificare panou pentru a corespunde cerintelor proiectului curent.

08.05.2022 : Definitivare carcasa necesara proiect cat si testare amplificator realizat + bug fixing.

14.05.2022 : Consolidare idee comunicatie dintre cele 2 microcontrolere pana la interactiunea cu serverul Blynk.

15.05.2022 : Alegerea intre diverse scheme de interconectare a celor 2 microcontrolere dupa mai multe teste.

21.05.2022 : Cel mai urat moment atunci cand ceea ce mergea pe bredboard are interferente si un comportament nedefinit odata adaugat la proiectul final.

22.05.2022 : Completarea sectiunilor OCW.

22.05.2022 : Realizare schema electrica si modificare schema bloc.

22.05.2022 : Adaugarea de imagini sugestive cu privire la parcursul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite.

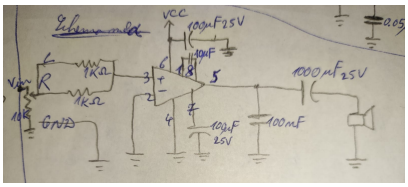
Resurse software:

- mediu de dezvoltare este Arduino 1.8.13

- pentru a reprezenta schema electrica am folosit EAGLE
- pentru schema bloc am folosit draw.io

Resurse hardware:

- Datasheet ATmega328P
- Arduino UNO pinout
- nodemcu-esp8266-pinout.jpg ← pinout NODEMCU esp8266
- External Link ← amlificator lm386 circuit nemodificat
- In imaginea de mai jos este schema amplificatorului modificata de mine cu potentiometru de 10k si condensatori de valori putin diferite care atenueaza mai bine zgomotul decat valorile condensatorilor prezentati in datasheet:



- Send Data from Arduino To ESp8266
- Informatii USART, I2C, SPI
- Serial Communication
- HC-SR04 Working Principle
- Gas Sensor Working Principle

Export to PDF