Dokumentacja

Sensory w aplikacjach wbudowanych	Temat II: Stacja Pogodowa
Założenia i plan pracy	**Mateusz Salamon** Damian Czajka Marcin Sikora

1. Założenia podstawowe

- Zbieranie danych z sensorów.

- Temperatury
- Ciśnienia
- Wilgotności
- Światła
    - Ewentualnych innych dostępnych na płytce.

- Kalibracja czujników (porównanie wartości z obu płytek).
- Wyświetlanie wartości na LCD.
- Odświeżanie wyników (uzależnione od czasu odświeżania sensorów).

Dodatkowo:

- Komunikacja przez Bluetooth z urządzeniem mobilnym.
- Logowanie danych do pliku.

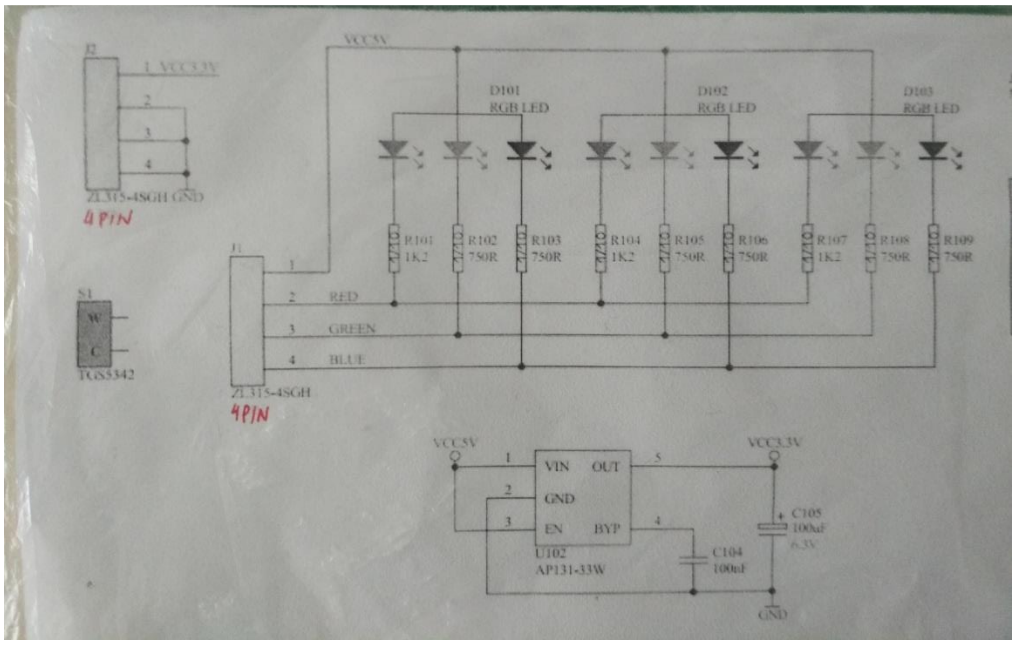
Harmonogram

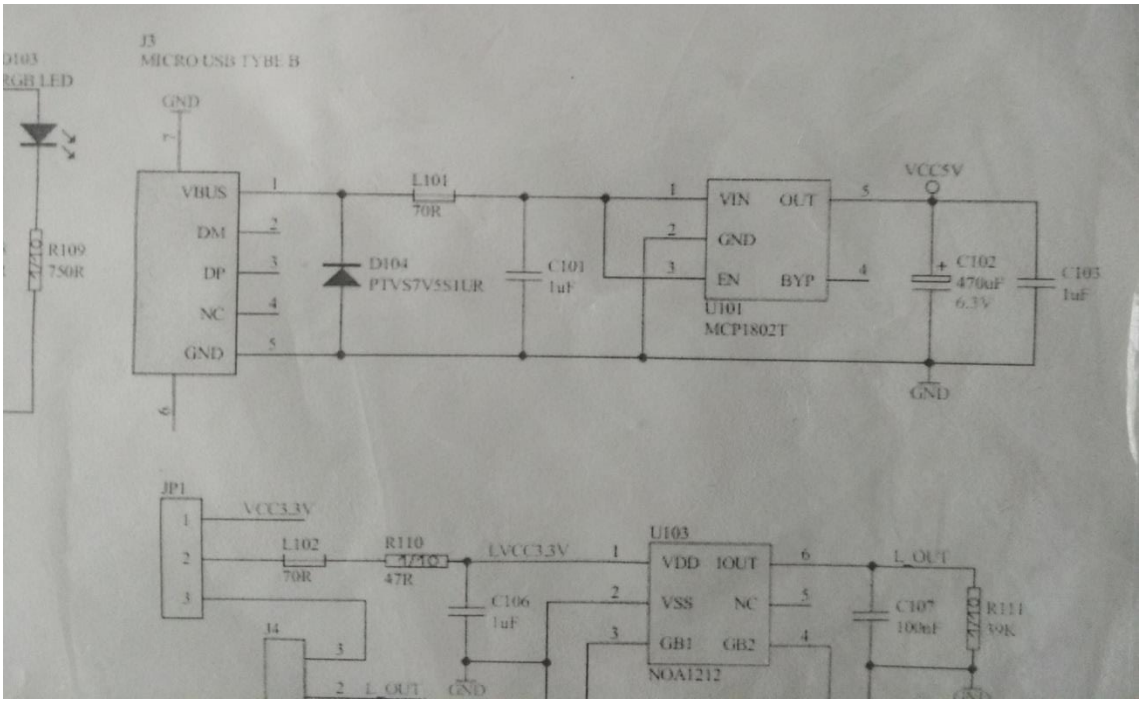
- 3 tygodnie (4.04) → Zbieranie danych z czujników.
  → Wyświetlanie na LCD.
- 2 tygodnie (18.04) → Kalibracja czujników.
  → Obróbka danych.
- 2 tygodnie (2.05) → Logowanie do pliku.
- 2 tygodnie (16.05)→ Poprawa błędów.
- 3 tygodnie(6.06) → Komunikacja przez Bluetooth z urządzeniem mobilnym.

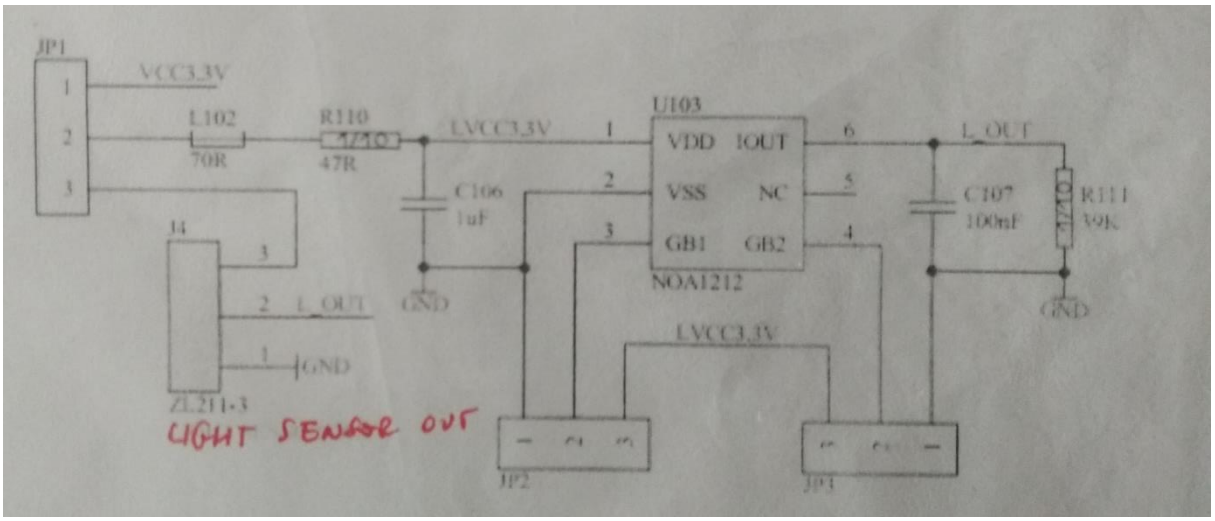
Spis treści

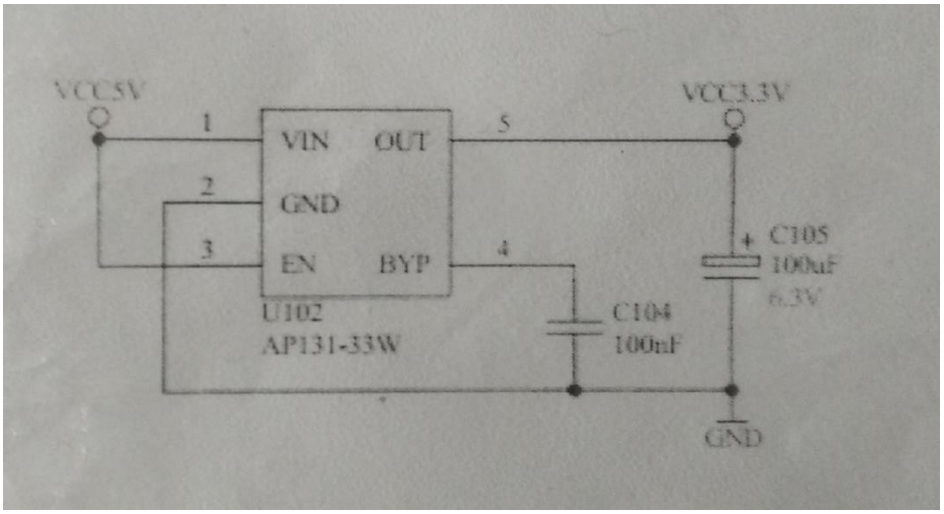
Wersja wysłana w ciągu semestru:

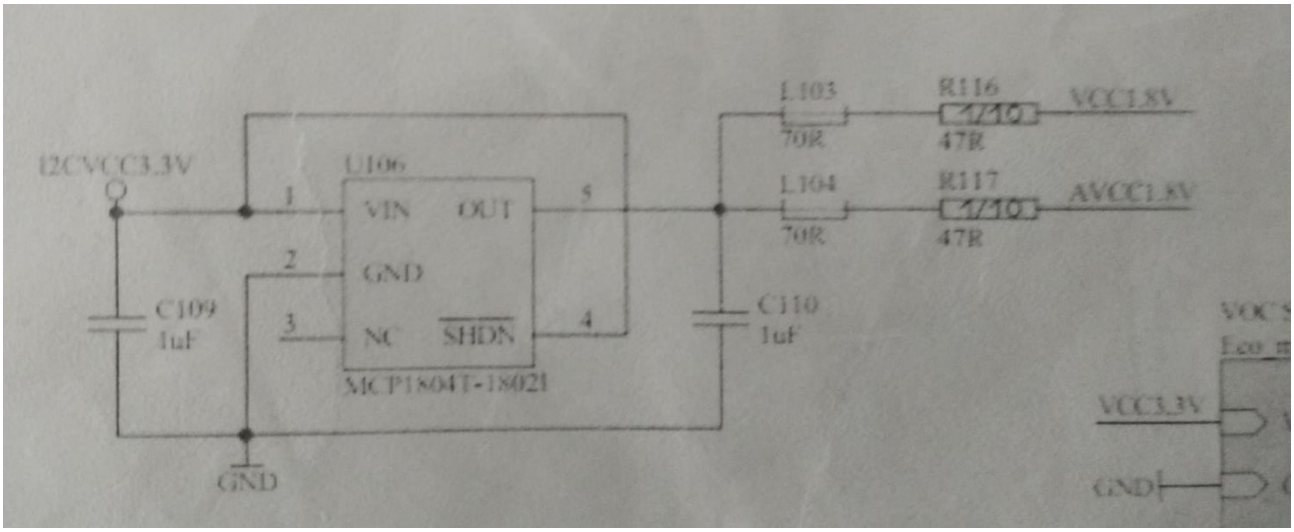
Schematy:

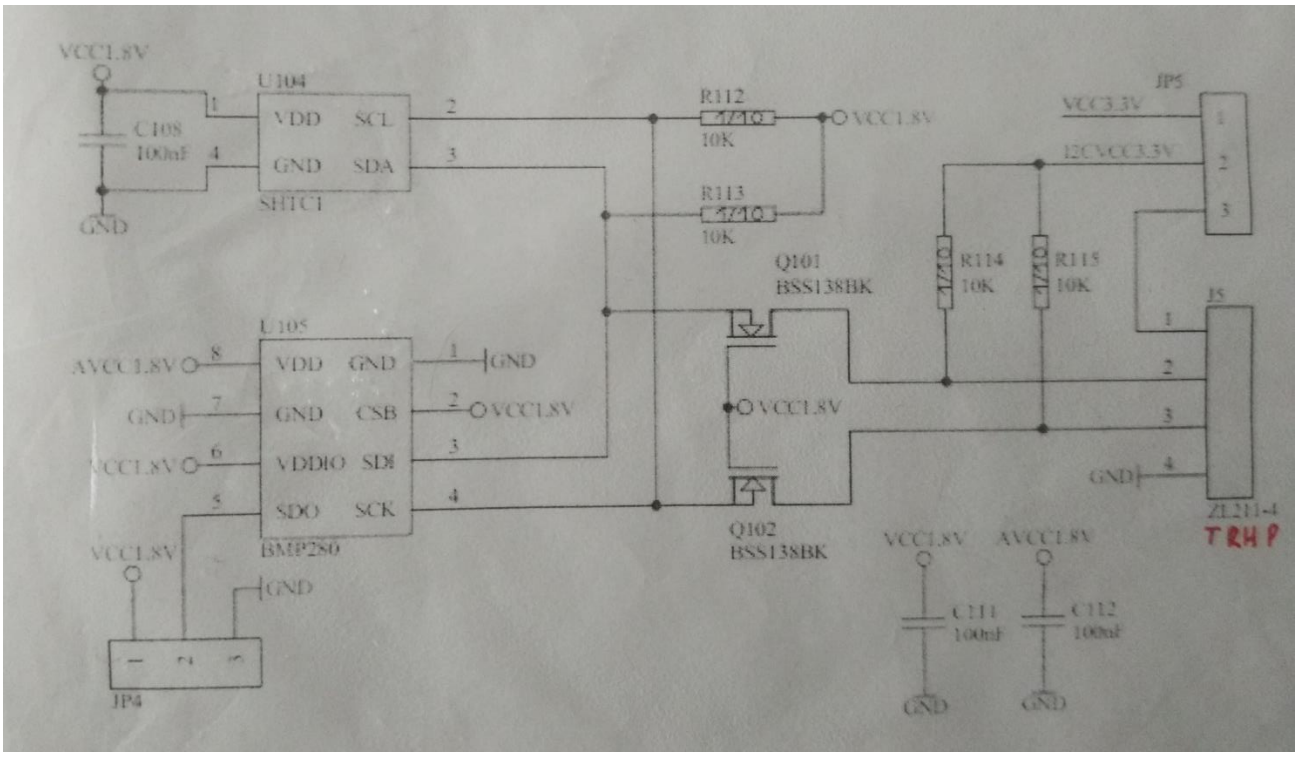


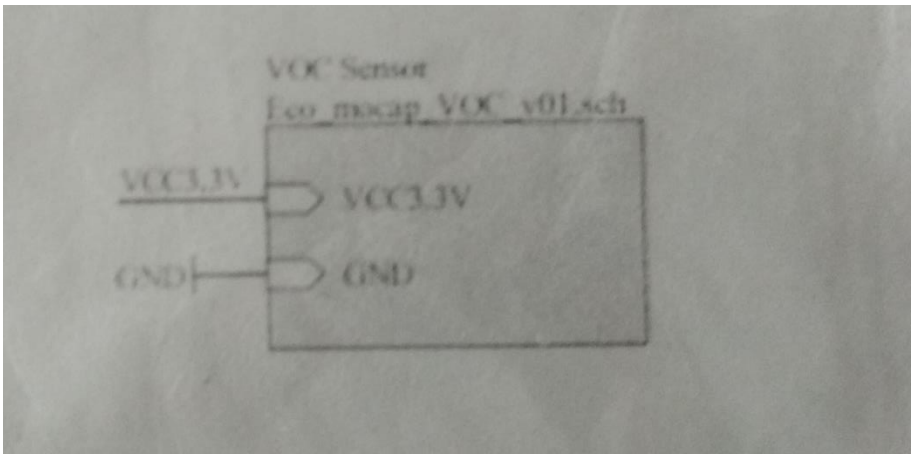


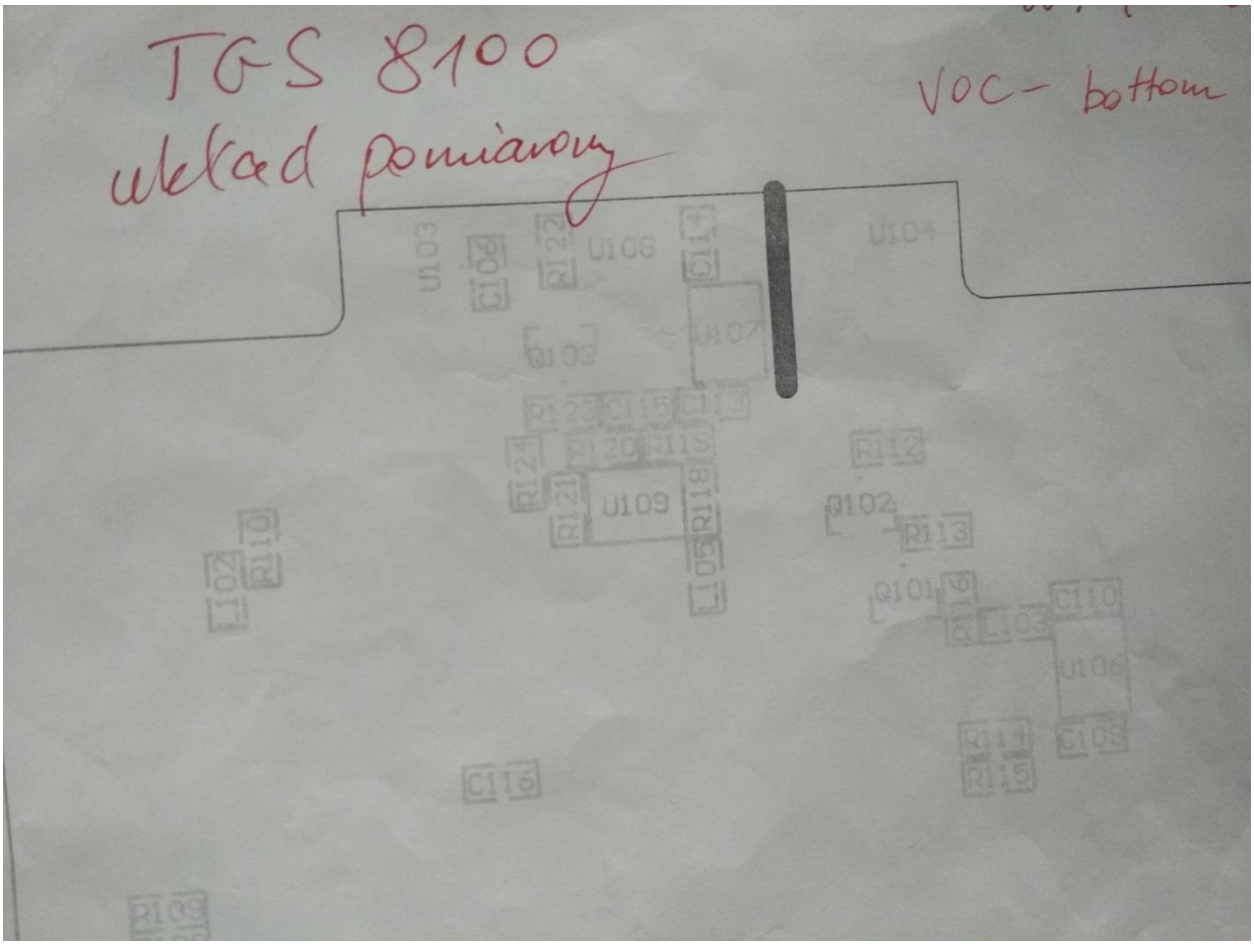

JP1
VCC3.3V
L102
70R
R110
47R
LVCC3.3V
U103
VDD
IOUT
VSS
NC
GB1
GB2
NOA1212
L_OUT
C106
1uF
C107
100nF
R111
10K
GND
J4
L_OUT
GND
ZL211-3
LIGHT SENSOR OUT
JP2
JP3

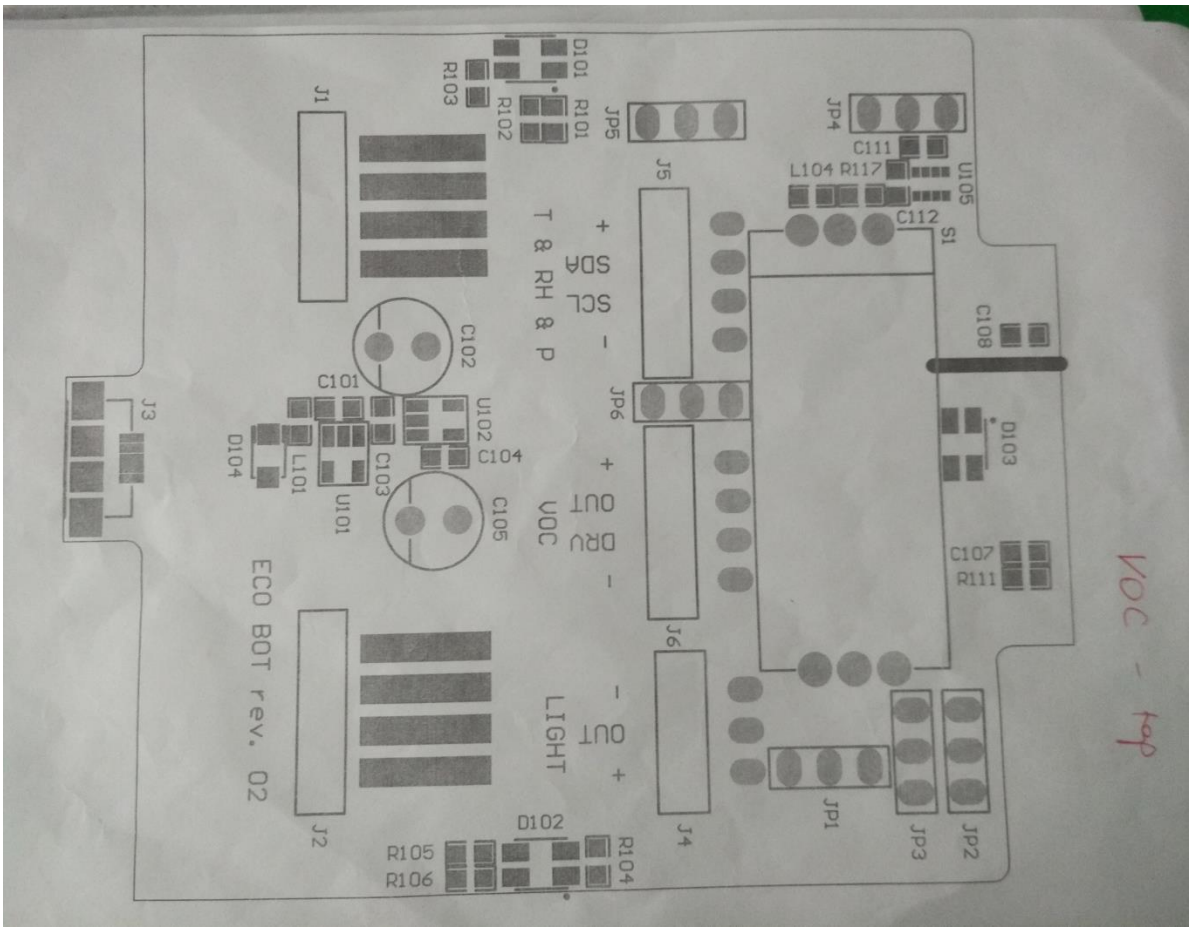

VCC5V
VCC3.3V
VIN
OUT
GND
EN
BYP
U102
AP131-33W
C105
100uF
6.3V
C104
100nF
GND

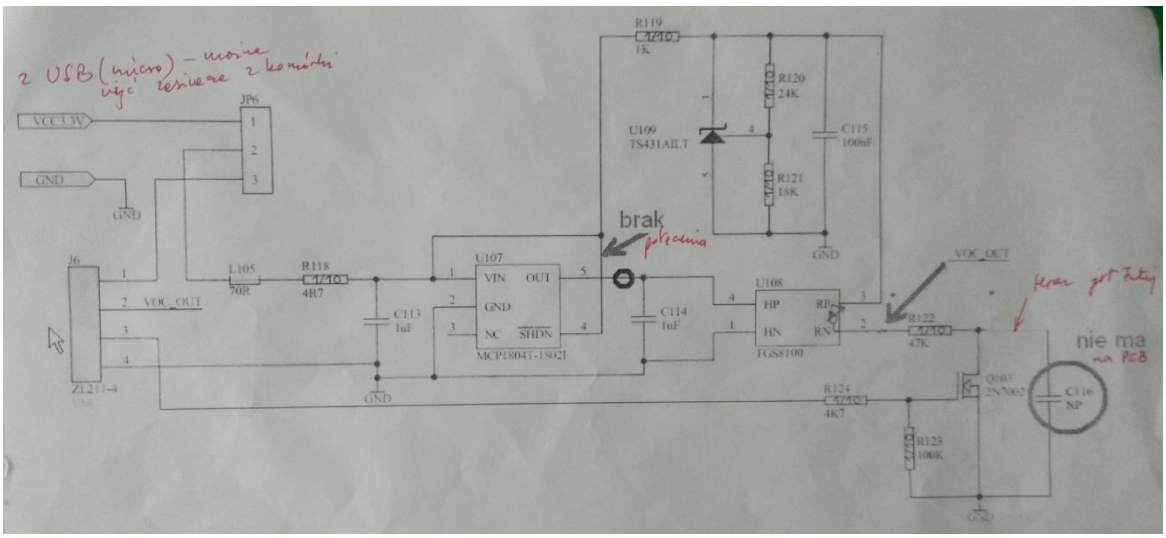

I2CVCC3.3V
U106
VIN
OUT
GND
NC
SHDN
MCP1804T-18021
C109
1uF
C110
1uF
L103
70R
R116
47R
VCC1.8V
L104
70R
R117
47R
AVCC1.8V
GND
VCC3.3V
GND

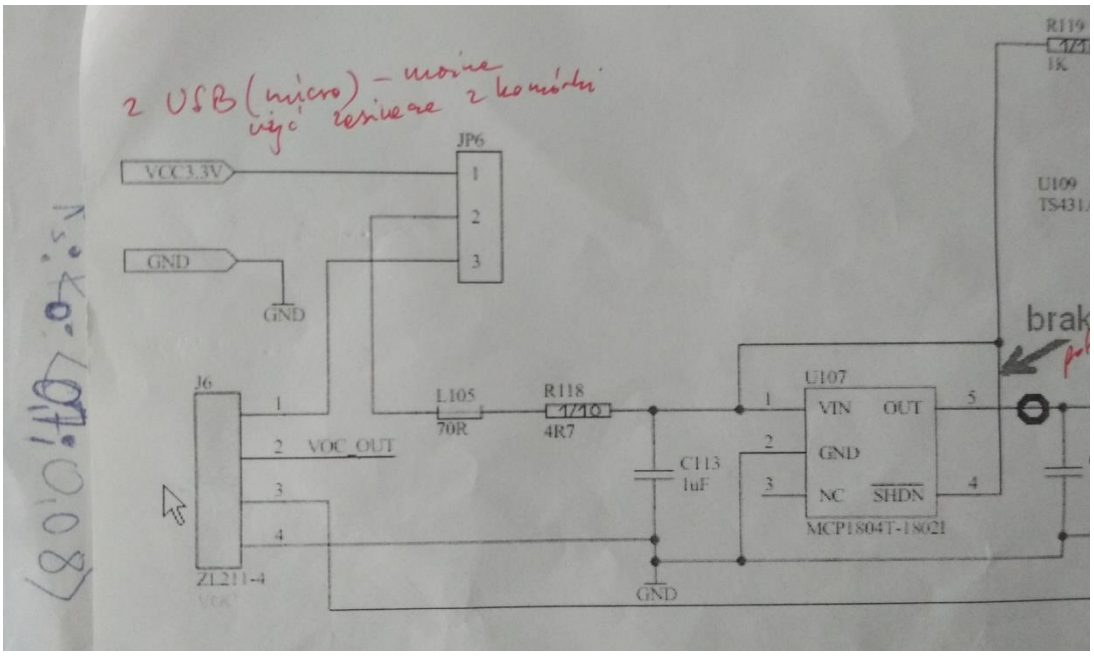

VCC1.8V
U104
C108
100nF
VDD
SCL
GND
SDA
SHTC1
R112
10K
R113
10K
VCC1.8V
JP5
VCC3.3V
I2C_VCC3.3V
Q101
BSS138BK
R114
10K
R115
10K
J5
U105
AVCC1.8V
GND
VDD
GND
CSB
VDDIO
SDI
SDO
SCK
BMP280
VCC1.8V
Q102
BSS138BK
C111
100nF
C112
100nF
AVCC1.8V
JP4
ZL211-4
TRHP

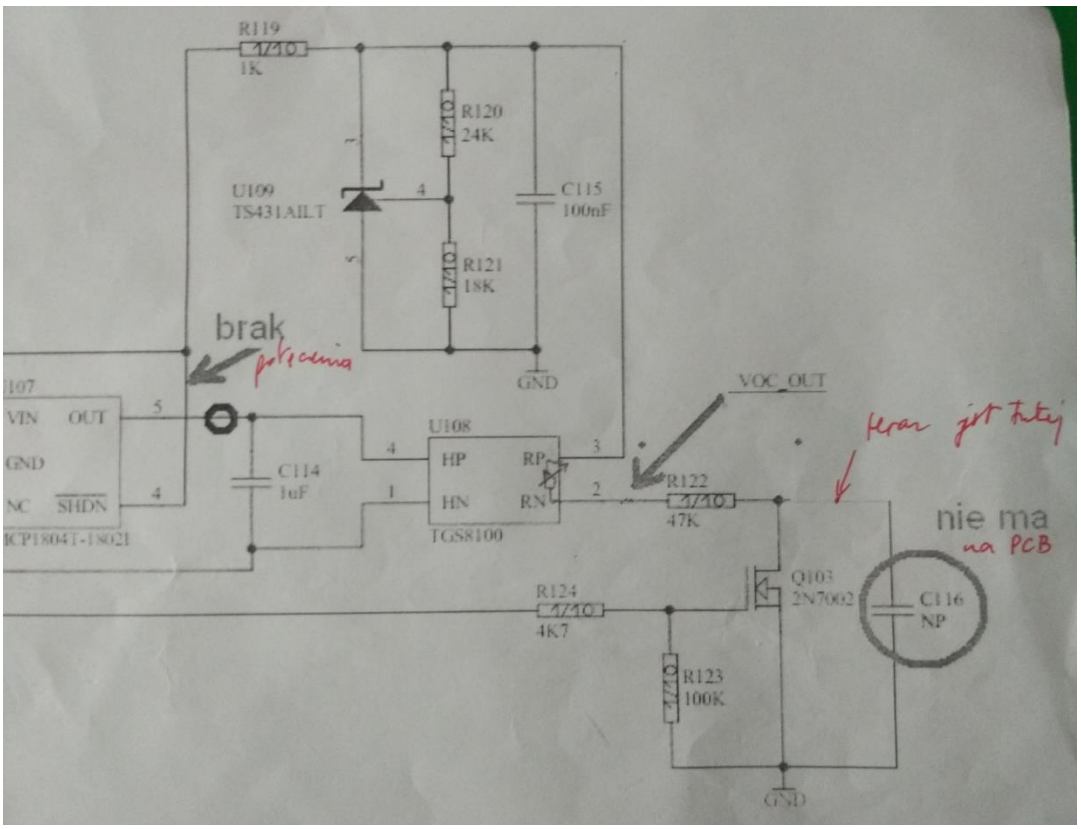

VOC Sensor
I_co_mosap_VOC_v01.sch
VCC3.3V
VCC3.3V
GND
GND

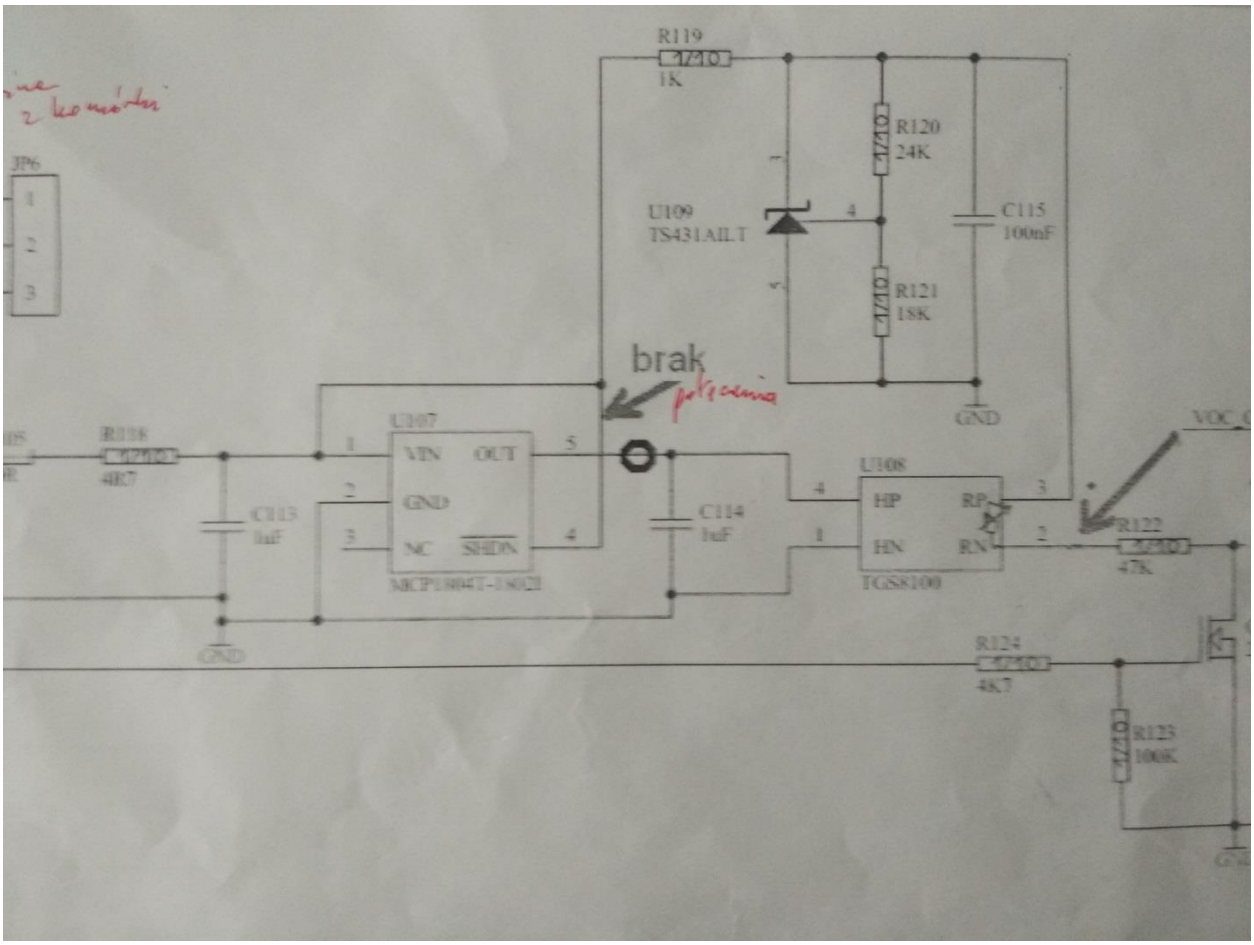

TGS 8100
układ pomiarowy
VOC - bottom
U103
C108
R122
U106
C114
U104
U107
Q103
R123
C115
C113
R120
R115
R124
R121
U109
R118
R112
Q102
R113
L102
R110
L105
Q101
C110
L103
U105
R114
C109
R115
C116
R103


ECO BOT rev. 02
T & RH & P
SCL
SDA
VOC
OUT
LIGHT
J1
J2
J3
J4
J5
J6
JP1
JP2
JP3
JP4
JP5
JP6
C101
C102
C103
C104
C105
C107
C108
C111
C112
D101
D102
D103
D104
L101
L104
R101
R102
R103
R104
R105
R106
R111
R117
S1
U101
U102
U105
VOC - top


2 USB (micro) – można
więc zasilać 2 komórki
JP6
VCC_3V
GND
J6
ZL211-4
VOC_OUT
L105
70R
R118
4R7
C113
1uF
U107
VIN
OUT
GND
NC
SHDN
MCP1804T-1802I
C114
1uF
brak
potrzeba
U109
TS431AILT
R119
1K
R120
24K
R121
18K
C115
100nF
U108
HP
HN
RP
RN
TGS8100
R122
47K
VOC_OUT
Ktoś jest zły
nie ma
na PCB
Q103
2N7002
C116
NP
R124
4K7
R123
100K


2 USB (micro) – można
użyć rezystora z komórki
JP6
VCC3.3V
GND
GND
R119
1K
U109
TS431
brak
J6
L105
70R
R118
4R7
U107
VIN
OUT
GND
NC
SHDN
MCP1804T-18021
C113
1uF
2 VOC_OUT
ZL211-4
GND


R119
1K
R120
24K
U109
TS431AILT
C115
100nF
R121
18K
brak
GND
VOC_OUT
U108
HP
RP
HN
RN
TGS8100
C114
1uF
R122
47K
Ekran jest tutaj
nie ma
na PCB
Q103
2N7002
C116
NP
R124
4K7
R123
100K
GND



Czujniki:

Czujnik Temperatury i ciśnienia atmosferycznego (BMA150) - Czujnik cyfrowy, z którym komunikujemy się za pomocą magistrali I2C. Czujnik ten ma możliwość zwracania wysokości nad poziomem morza wyliczając to na podstawie średniego ciśnienia atmosferycznego dla określonej wysokości oraz ciśnienia odbieranego z otoczenia.

Czujnik natężenia światła (NOA1212) - Czujnik analogowy, Nie jest obecny na jednej z płytek

Czujnik wilgotności powietrza i temperatury (SHTC1) – Czujnik cyfrowy, podpięty do tej samej magistrali I2C, przy której jest czujnik ciśnienia atmosferycznego i temperatury.

Czujnik zanieczyszczenia powietrza (TGS 8100) – Czujnik analogowy wykrywający dym i zanieczyszczenia gazowe.

Szczegółowe informacje na temat czujników odczytujemy z ich not katalogowych.

Przykładowy kod do zbierania danych z czujników:

```
#include <Wire.h>
#include "SPI.h" //Why? Because library supports SPI and I2C connection
#include <Adafruit_Sensor.h>
#include "Adafruit_BMP280.h"

//Setup connection of the sensor
Adafruit_BMP280 bmp; // I2C
/*//For SPI connection!
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10
//Adafruit_BMP280 bme(BMP_CS); // hardware SPI
//Adafruit_BMP280 bme(BMP_CS, BMP_MOSI, BMP_MISO,  BMP_SCK);
*/

//Variables
float pressure;   //To store the barometric pressure (Pa)
float temperature;  //To store the temperature (oC)
int altimeter;    //To store the altimeter (m) (you can also use it as a
float variable)

void setup() {
  bmp.begin();    //Begin the sensor
    Serial.begin(9600); //Begin serial communication at 9600bps
    Serial.println("Adafruit BMP280 test:");
}

void loop() {
  //Read values from the sensor:
  pressure = bmp.readPressure();
  temperature = bmp.readTemperature();
  altimeter = bmp.readAltitude (1050.35); //Change the "1050.35" to your city
current barrometric pressure (https://www.wunderground.com)

  //Print values to serial monitor:
  Serial.print(F("Pressure: "));
    Serial.print(pressure);
    Serial.print(" Pa");
    Serial.print("\t");
    Serial.print(("Temp: "));
    Serial.print(temperature);
    Serial.print(" oC");
  Serial.print("\t");
    Serial.print("Altimeter: ");
    Serial.print(altimeter); // this should be adjusted to your local forcase
    Serial.println(" m");

//    int sensorValue = analogRead(A0);
//  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0
- 5V):
//  float voltage = sensorValue * (5.0 / 1023.0);
```

```
//  // print out the value you read:
//  Serial.print("Volt ");
//  Serial.print(voltage);

    delay(5000); //Update every 5 sec
}
```

Kod jest w stanie zbierać wartości z czujnika cyfrowego komunikując się za pomocą magistrali I2C oraz z podłączonego wyjścia analogowego z 1 czujnika. Zbieranie wartości z reszty czujników robiona jest analogicznie, wystarczy podpiąć kolejne wejścia analogowe i je zainicjalizować analogicznie do przykładu powyżej a także ustalić adresy czujników podpiętych do magistrali I2C. Dla przykładu przedstawiony jest kod dla czujnika ciśnienia, analogicznie wykonuje się kod dla czujnika wilgotności.

Niektóre piny o szczególnych funkcjonalnościach jak GND lub 5V zostały oznaczone na powierzchni PCB, Oznaczone zostały również, ważniejsze złączki odpowiadające za zasilanie modułów.

Zostało wprowadzone sterowanie LEDami, jedyną możliwością ich konfiguracji jest wybór koloru lub stanu świecenia, ponieważ wszystkie podpięte są do wspólnej masy dla jednego koloru. Na przykład zapalając jeden niebieski wszystkie powinny świecić się na niebiesko, ale jedna dioda jest uszkodzona i brakuje jej koloru niebieskiego, ale za to przez jej złe połączenie świeci kolorem zielonym i nie świeci w ogóle, gdy wszystkie ustawiane są na kolor zielony.

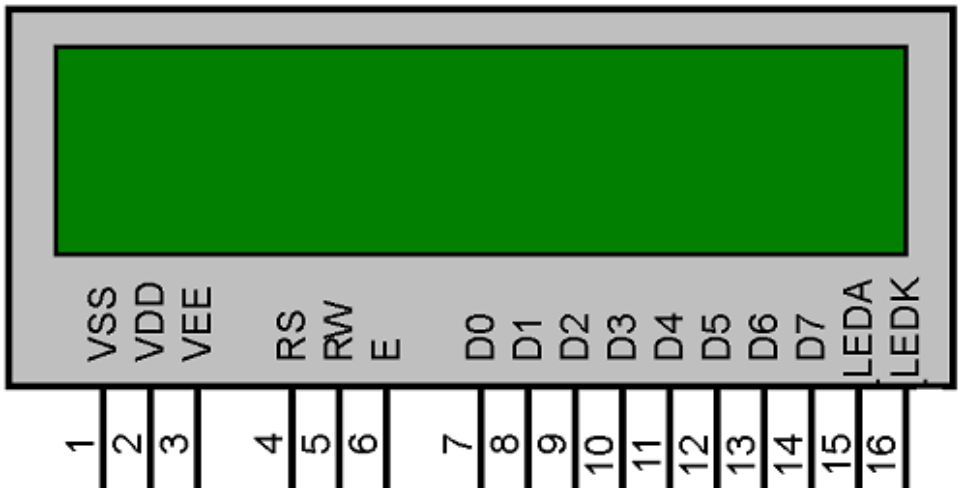
Na prywatnym repozytorium projektowym dostępny jest plik z Implementacją mrugania LEDów oraz zmianami ich koloru demonstrujący nieprawidłowe działanie jednej z diod.

Wyświetlacz:

LCD HD44780. Jest to popularny wyświetlacz, dostępny na rynku i wykorzystywany w wielu rodzajów projektach. Zaletą tego urządzenia jest możliwość komunikacji z mikrokontrolerem za pomocą magistrali I2C lub wykorzystując porty GPIO. W projekcie zastosowana została komunikacja za pomocą portów GPIO.

Dostępnych jest kilka rodzajów tego wyświetlacza, oferujących różną liczbę wierszy i kolumn oraz wymagane napięcie zasilania. W projekcie wykorzystana została wersja 2 × 16 (2 wiersze, 16 kolumn).

HD44780 posiada 16 pinów za pomocą, których możliwa jest komunikacja z mikrokontrolerem.



Rysunek 1. Wyświetlacz ze sterownikiem HD44780. [19]

Opis wykorzystywanych wejść wyświetlacza:

1) **VSS** - masa.
2) **VDD** - zasilanie urządzenia - 5V
3) **V0** – regulacja kontrastu. Przy użyciu potencjometru lub dzielnika napięcia składającego się z rezystorów należy ustawić odpowiednie napięcie na pinie w celu osiągnięcia pożądanego kontrastu.
4) **RS** – wybór rejestru. Za pomocą stanu logicznego na tym wejściu informujemy sterownik HD44780 o tym, jaka informacja zostanie wysłana. Stan wysoki (H) odpowiada wysyłaniu danych, stan niski (**L**) komendy.
5) **RW** – odczyt/zapis. Pin ten służy do wyboru odczytu lub zapisu. H - odczyt, L - zapis.
6) **E** – zegar. Pin ten służy do synchronizacji wysyłanych danych.
7) **D0 - D7** – linia danych. W 4 bitowym trybie przesyłania danych, transmisja danych odbywa się za pomocą wejść D4 - D7.
8) **A** – anoda, **K** – katoda. Piny obsługujące podświetlenie wyświetlacza. A łączymy z zasilaniem, K z masą.

Sterownik działa w trybie 4 oraz 8 bitowym, jako że wykorzystany został tryb 4 bitowy, piny D0-D3 nie są obsługiwane.

Sterowanie wyświetlaczem odbędzie się za pomocą portów GPIO. Odbywa się to poprzez ustawienia określonych stanów logicznych na pinach w odpowiedniej sekwencji.

Zależnie od stanu logicznego na pinie RW wybieramy rodzaj transmisji, w naszym przypadku będzie to tylko zapis do wyświetlacza, stąd pin ten zwarty jest do masy.

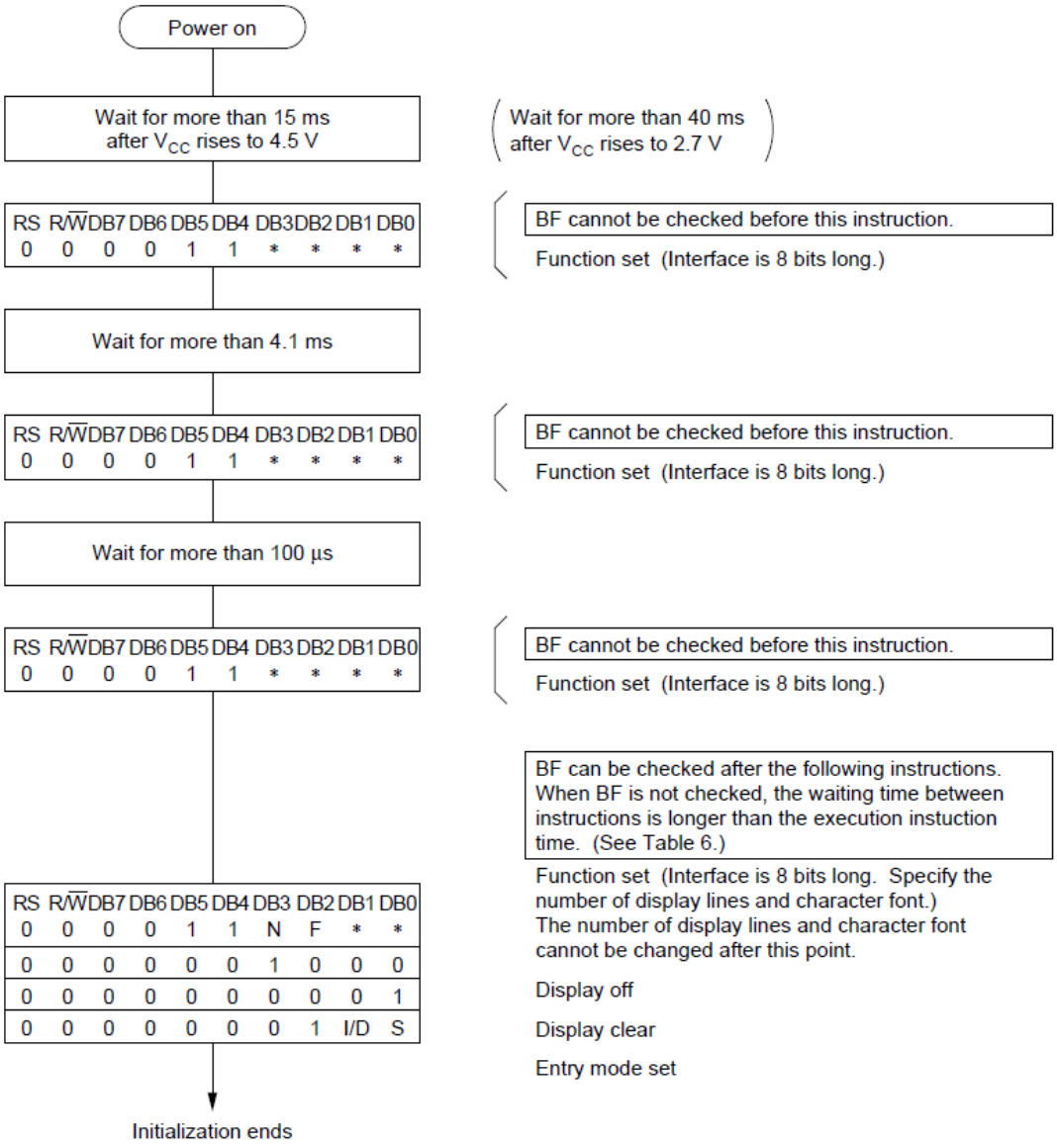
Wysyłanie 1 bajta danych w trybie 4-bitowym odbywa się w dwóch sekwencjach. Każda połowa bajta inicjalizowana jest zboczem narastającym zegara, a kończona zboczem opadającym. Dzięki zastosowanym maskom możemy uzyskać oczekiwany stan na odpowiednich liniach danych. Przesyłanie całości danych odbywa się poprzez dwukrotne wywołanie tej funkcji.

Do opisywanego wyświetlacza możemy wysłać komendę lub znak. Odbywa się to poprzez ustalenie odpowiedniego stanu logicznego na wejściu RS.

Tabela poniżej prezentuje, jaka sekwencja powinna być wysłana w celu otrzymania określonego znaku. Każda sekwencja powinna być poprzedzona ustawieniem stanu wysokiego na wejściu RS.

Inicjalizacja LCD

Najważniejszą częścią korzystania z wyświetlacza jest jego odpowiednia inicjalizacja. Operacja ta musi odbyć się zgodnie z sekwencją przedstawioną w nocie katalogowej urządzenia.



Konieczne jest przestrzeganie określonych ram czasowych. Zanim przedstawiony wcześniej schemat zostanie wywołany, konieczne jest ustawienie wszystkich wykorzystywanych pinów, jako wyjściowe. Następnie można wysłać przedstawioną wcześniej sekwencję.

Koniec części pierwszej wysłanej w trakcie semestru, w projekcie dokonano kilku usprawnień i zmian w porównaniu z wersją poprzednią.

Wersja Finalna

Krótki opis:

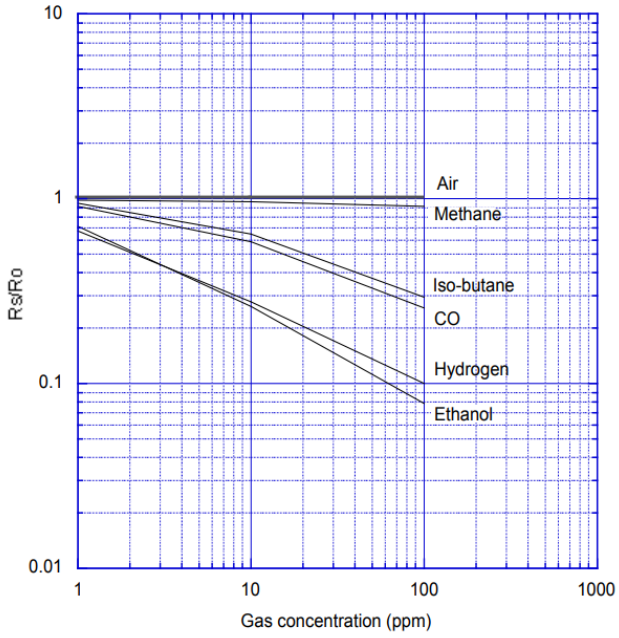
Nie dało się nawiązać komunikacji i2c z sensorem BMP280 ale udało się to z sensorem SHTC1. W konsekwencji BMP280 został zastąpiony czujnikiem MPL3115A2, który, tak jak BMP280, mierzy wysokość, ciśnienie i temperaturę. Zewnętrzny czujnik MPL został podpięty przez drugie wyjście I2C, ponieważ na płytce sensorycznej nie ma miejsca na podłączenie się do magistrali.

Wartość czujnika TGS 8100 firmy Figaro jest przetwarzana przez wewnętrzny ADC mikrokontrolera zgodnie ze wzorem podanym w nocie katalogowej:

$$Rs = \left(\frac{Vc}{Vout} - 1\right) * Rl$$

Gdzie Vc to napięcie zasilania
Vout to napięcie zmierzone na wyjściu czujnika
Rl jest równe rezystancji obciążenia (rezystor 47kΩ)

Sensitivity Characteristics:



Wartość zwracana przez czujnik jest przetwarzana na iloraz rezystancji Rs do Ro gdzie Ro jest to rezystancja TGS8100 na świeżym powietrzu i wynosi 33kΩ.

Niestety, po ustawieniu czujnika nad szklanką z alkoholem nie wykazał żadnej zmiany nie dając możliwości kalibracji i zobaczenia zmian w wynikach.
Inne testy są nie możliwe do przeprowadzenia, ze względu na brak dostępności tych gazów.

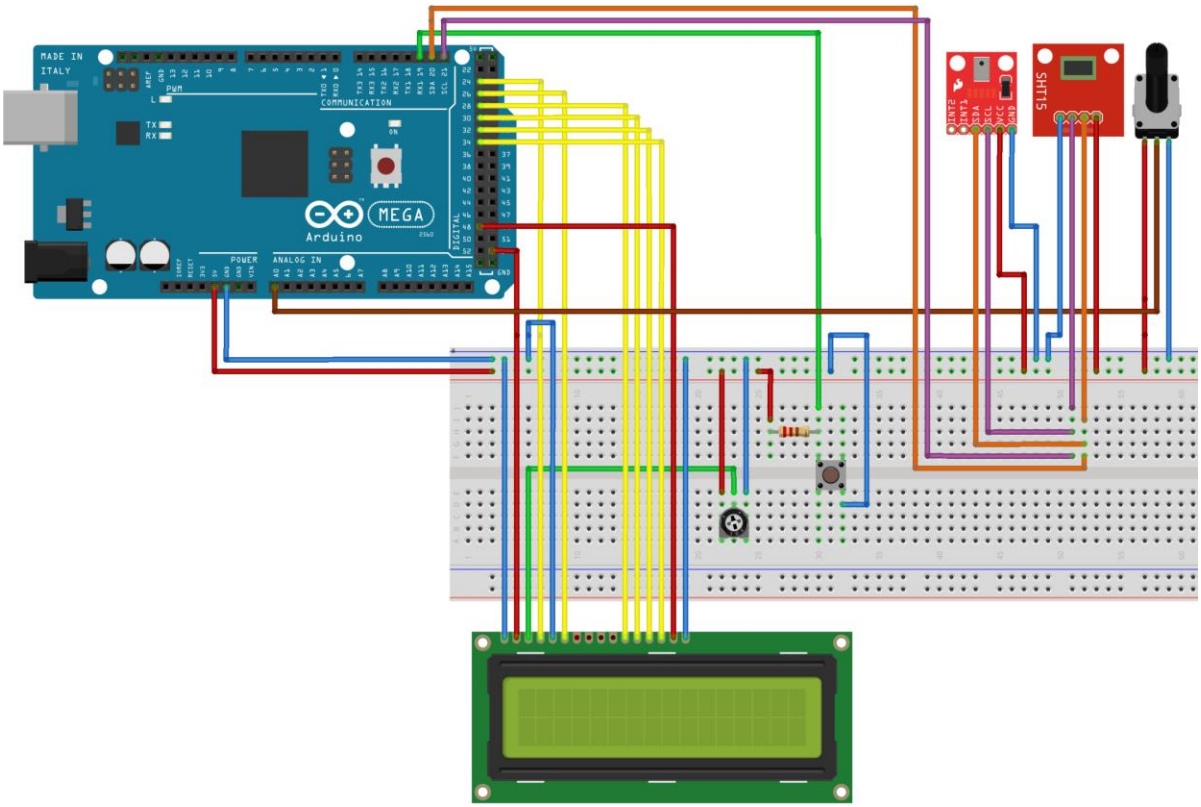
Dane przesyłane przez UART mogą być odczytane przez USB na komputerze za pomocą konwertera USB UART.

Przykładowe wyniki sensora MPL3115A2 i SHTC1, pokazują nieznaczne rozbieżności pomiędzy tymi czujnikami:

```
275.38 meters
28.69*C
OUT: 32604.31
  RH: 50.24
  T:  27.81
29.04 Inches (Hg)
275.00 meters
28.75*C
OUT: 32604.31
  RH: 50.27
  T:  27.82
29.04 Inches (Hg)
274.56 meters
28.87*C
OUT: 32604.31
  RH: 50.25
  T:  27.79
29.04 Inches (Hg)
274.69 meters
28.94*C
OUT: 32736.32
  RH: 50.24
  T:  27.81
OUT: 32604.31
  RH: 50.22
  T:  27.80
```

Wyświetlanie danych zostało zrobione na zasadzie przerwania zboczem opadającym. Gdzie po wykryciu przez pin mikrokontrolera ekran rozświetla się i odświeżona zostanie wartość wypisana na LCD.

Schemat połączeniowy:



Na tym schemacie czujniki płytki zielonej zastąpione zostały poprzez czujnik cyfrowy (SHT1S) odpowiadający czujnikowi SHTC1 i potencjometr odpowiadający czujnikowi analogowemu TGS8100.

Do magistrali I2C podpięty jest również barometr MPL3115A2 znajdujący się po lewej od wyżej wymienionych sensorów.

Przycisk został zrealizowany, jako pull up poprzez zastosowanie rezystancji 10kΩ.

Potencjometr kontroluje kontrast na LCD.

Szczegółowy opis funkcjonalności:

Obsługa LCD sprowadza się do wykorzystania biblioteki LiquidCrystal i realizowana jest w głównej mierze przez funkcję `LCDplay()`, która przekazuje do LCD 16x2 informacje o temperaturze i wilgotności względnej:

```
void LCDplay(void)
{
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("RH: ");
  lcd.print(humidity);
  lcd.print(" %");
  lcd.setCursor(0,1);
  lcd.print("T:  ");
  lcd.print(temperature);
  lcd.print(" C");

  if(warningflag == 1)
  {
  lcd.setCursor(0,0);
  lcd.print(warning);
  lcd.setCursor(0,1);
  lcd.print(airquality);
  }
}
```

Funkcja wywołana jest przez `ButtonHandler()` gdzie przerwanie spowodowane naciśnięciem przycisku załącza napięcie do podświetlenia LCD oraz wywołuje uaktualnienie danych:

```
void ButtonHandler(void){
  butstate = digitalRead(ButPIN);
  if (butstate == LOW)
  {
    digitalWrite(LCD_LIGHTPIN,HIGH);
    LCDplay();
    Serial.println("UP");
  }
  else if(butstate == HIGH){
  digitalWrite(LCD_LIGHTPIN,LOW);
 }
}
```

Funkcja `SHT_handler()` Obsługuje czujnik SHTC1. Komunikuje się przez magistralę I2C i przypisuje wyniki do zmiennych, z których korzysta wyświetlanie na LCD, oraz przesyła wyniki na zewnątrz przez UART. W przypadku odłączenia sensora przekazywany będzie ciągle ten sam wynik aż do ponownego jego podłączenia.

```
void SHT_handler(){
  if(init_sht == 1){
  sht.readSample();
    humidity = sht.getHumidity();
    humidity = humidity;
```

```
    Serial.print("  RH: ");
    Serial.print(humidity, 2);
    Serial.print("\n");

    temperature = sht.getTemperature();
    temperature = temperature;
    Serial.print("  T:  ");
    Serial.print(temperature, 2);
    Serial.print("\n");
  }
}
```

Funkcja `MPL_handler()` obsługuje komunikację z czujnikiem MLP3115A2. Komunikuje się przez magistralę I2C i przypisuje wyniki do zmiennych, które następnie wysłane są na zewnątrz przez UART. W przypadku odłączenia sensora funkcja nie będzie nadpisywać i wysyłać danych.

```
void MPL_handler(){
 if (! baro.begin())
 {
  init_mpl = 0;
 }
 else{
  init_mpl = 1;
  }

 if(init_mpl == 1){
  pascals = baro.getPressure();
  Serial.print(pascals/3377); Serial.println(" Inches (Hg)");

  altm = baro.getAltitude();
  Serial.print(altm); Serial.println(" meters");

  tempC = baro.getTemperature();
  Serial.print(tempC); Serial.println("*C");
 }
}
```

Dla analogowego sensora TGS8100 wyjście jest sprawdzane przez ADC mikrokontrolera, następnie przekształcane jest na stosunek rezystancji chwilowej do rezystancji w świeżym powietrzu.

```
 float sensorValue = analogRead(APIN);
 float voltage = sensorValue * (Vcc / 1023.0);// Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 3.3V):
 float resistance = (Vcc/voltage - 1)*Rl;
 float output = resistance/Ro; //Outputs  resistance by normal resistance for further analysis:
```

Jeżeli stosunek rezystancji spadnie poniżej 0.8 świadczy to o złym stanie powietrza. Niestety, nie udało się testować poszczególnych wartości dla różnych rodzajów zanieczyszczeń. Po czym ta wartość jest przesyłana na zewnątrz przez UART.

```
 if(output <= 0.8){
   warningflag = 1 ;
   Serial.println(warning);
   Serial.println(airquality);
 }
 else{
   warningflag = 0;
 }
 Serial.print("OUT: ");
 Serial.print(resistance);
 Serial.print("\n");
```

Pełen Kod:

```
#include <Wire.h>
#include "SHTSensor.h"
#include <Adafruit_MPL3115A2.h>
#include "String.h"
#include "LiquidCrystal.h"

#define Rl 47000
#define Vcc 3.3
#define Ro 33000
#define ButPIN 19 //Pullup resistor switch
#define LCD_PWRPIN 53
#define LCD_LIGHTPIN 48
#define APIN A0

SHTSensor sht;
Adafruit_MPL3115A2 baro = Adafruit_MPL3115A2();
LiquidCrystal lcd(24,26,28,30,32,34);  //LCD PIN mapping for Mega

String warning = "WARNING! BAD";
String airquality = "AIR QUALITY";
bool init_sht = 0;
bool init_mpl = 0;
float humidity = 0;
float temperature = 0;
bool warningflag = 0;
int butstate;
float pascals = 0;
float altm = 0;
float tempC = 0;

void LCDplay(void) //LCD Display function for temperature
{
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("RH: ");
  lcd.print(humidity);
  lcd.print(" %");
  lcd.setCursor(0,1);
  lcd.print("T:  ");
  lcd.print(temperature);
  lcd.print(" C");

  if(warningflag == 1)
```

```
  {
  lcd.setCursor(0,0);
  lcd.print(warning);
  lcd.setCursor(0,1);
  lcd.print(airquality);
  }
}

void ButtonHandler(void){
  butstate = digitalRead(ButPIN);
  if (butstate == LOW)
  {
    digitalWrite(LCD_LIGHTPIN,HIGH);
    LCDplay();
    Serial.println("UP");
  }
  else if(butstate == HIGH){
  digitalWrite(LCD_LIGHTPIN,LOW);
 }
}

void SHT_handler(){
  if(init_sht == 1){
  sht.readSample();
    humidity = sht.getHumidity();
    humidity = humidity;
    Serial.print("  RH: ");
    Serial.print(humidity, 2);
    Serial.print("\n");

    temperature = sht.getTemperature();
    temperature = temperature;
    Serial.print("  T:  ");
    Serial.print(temperature, 2);
    Serial.print("\n");
  }
}
void MPL_handler(){
  if (! baro.begin())
  {
    init_mpl = 0;
  }
  else{
    init_mpl = 1;
    }

  if(init_mpl == 1){
    pascals = baro.getPressure();
    Serial.print(pascals/3377); Serial.println(" Inches (Hg)");
```

```
    altm = baro.getAltitude();
    Serial.print(altm); Serial.println(" meters");

    tempC = baro.getTemperature();
    Serial.print(tempC); Serial.println("*C");
  }
}

void setup() {

  Wire.begin();
  Serial.begin(9600);
  delay(1000); // let serial console settle
/*Initialize SHT sensor */
  if (sht.init()) {
    Serial.print("init SHT: success\n");
    init_sht = 1;
  }
  else {
    Serial.print("init SHT: failed\n");
    init_sht = 0;
  }
  /*Initialize MPL sensor*/
  if (! baro.begin())
  {
    Serial.println("Couldnt find MPL sensor");
    init_mpl = 0;
  }
  else{
    Serial.println(" init MPL: success ");
    init_mpl = 1;
    }

    pinMode(ButPIN,INPUT_PULLUP);
    pinMode(LCD_LIGHTPIN,OUTPUT);
    pinMode(LCD_PWRPIN,OUTPUT);
    digitalWrite(LCD_LIGHTPIN,LOW);
    digitalWrite(LCD_PWRPIN,HIGH);

    attachInterrupt(digitalPinToInterrupt(ButPIN), ButtonHandler, FALLING);
// attached interrupt to pin for detecting button press

    lcd.begin(16, 2);
    lcd.clear();
}

void loop() {

  SHT_handler();
```

```
  MPL_handler();

  float sensorValue = analogRead(APIN);
  float voltage = sensorValue * (Vcc / 1023.0);// Convert the analog reading
(which goes from 0 - 1023) to a voltage (0 - 3.3V):
  float resistance = (Vcc/voltage - 1)*Rl;
  float output = resistance/Ro; //Outputs  resistance by normal resistance
for further analysis:

  if(output <= 0.8){
    warningflag = 1 ;
    Serial.println(warning);
    Serial.println(airquality);
  }
  else{
    warningflag = 0;
  }

  Serial.print("OUT: ");
  Serial.print(resistance);
  Serial.print("\n");
  ButtonHandler(); // Lights up LCD on button press

  delay(1000);
}
```