**ПМГ „Акад. Боян Петканчин“ – гр. Хасково**

**НП „Обучение за ИТ Кариера“**

**Модул 8**

**Метеорологична станция**

**Документация**

Димитър Бялков

# Съдържание

[Описание на проекта 3](#_Toc1842189566)

[Блокова схема 3](#_Toc1911322274)

[Електрическа схема 4](#_Toc1983009294)

[Монтажна схема 5](#_Toc1816934655)

[Списък съставни части 5](#_Toc576601087)

[Прототип 6](#_Toc70602498)

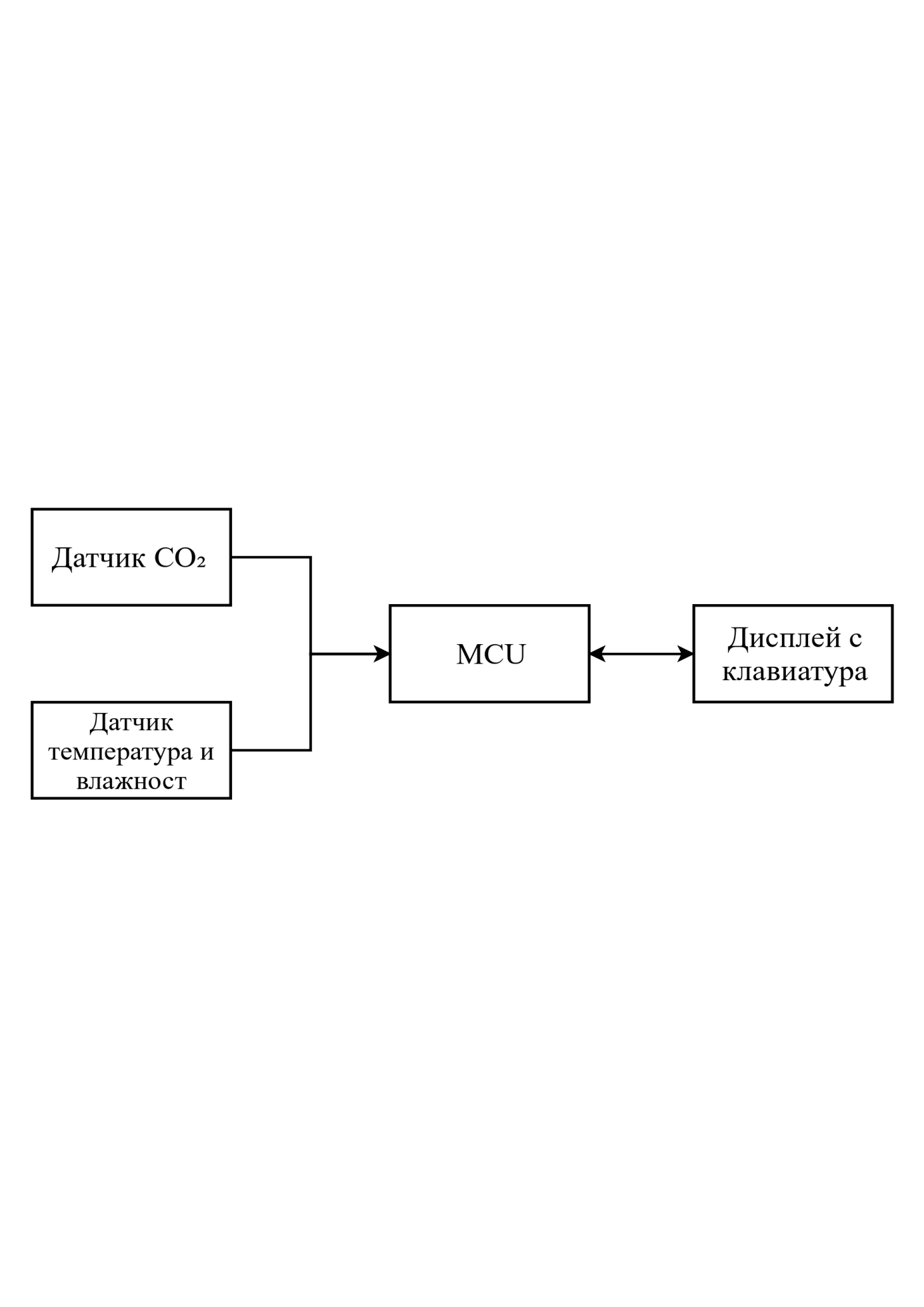
[Изходен код 6](#_Toc2143769433)

[Заключение 15](#_Toc2088391921)

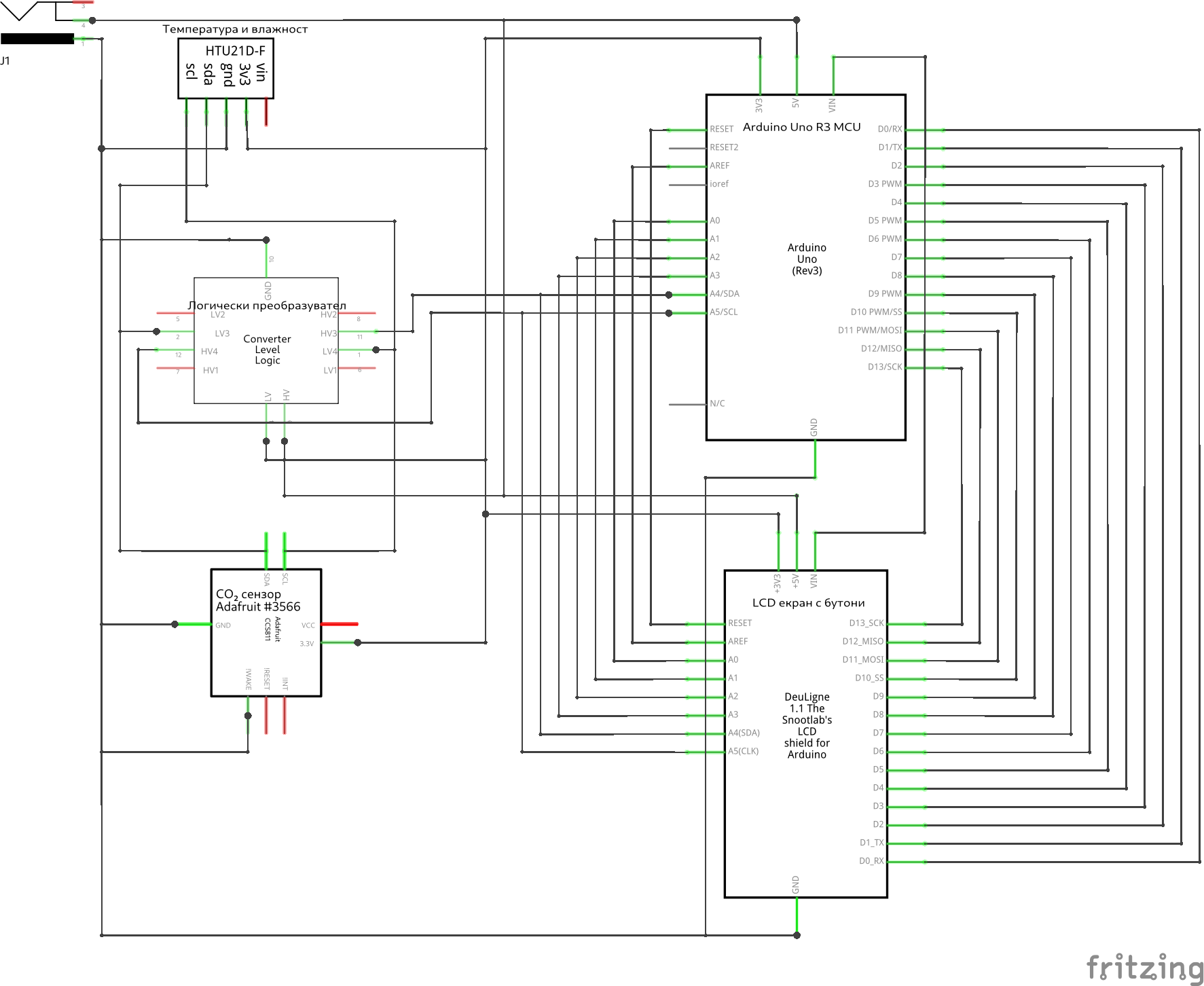
# Описание на проекта

Проектът представлява метеоролигна станция, която показва на потребителя текущи данни за температурата и качеството на въздуха в помещението посредством сензорите и известява, когато показателите бъдат превишени или са под норма. Чрез менюто за настройка се задават границите за стойностите. Когато датчиците засекат превишени или поднормени стойности на екрана се извежда предупреждение, което потребителя може да отхвърли за десет секунди.

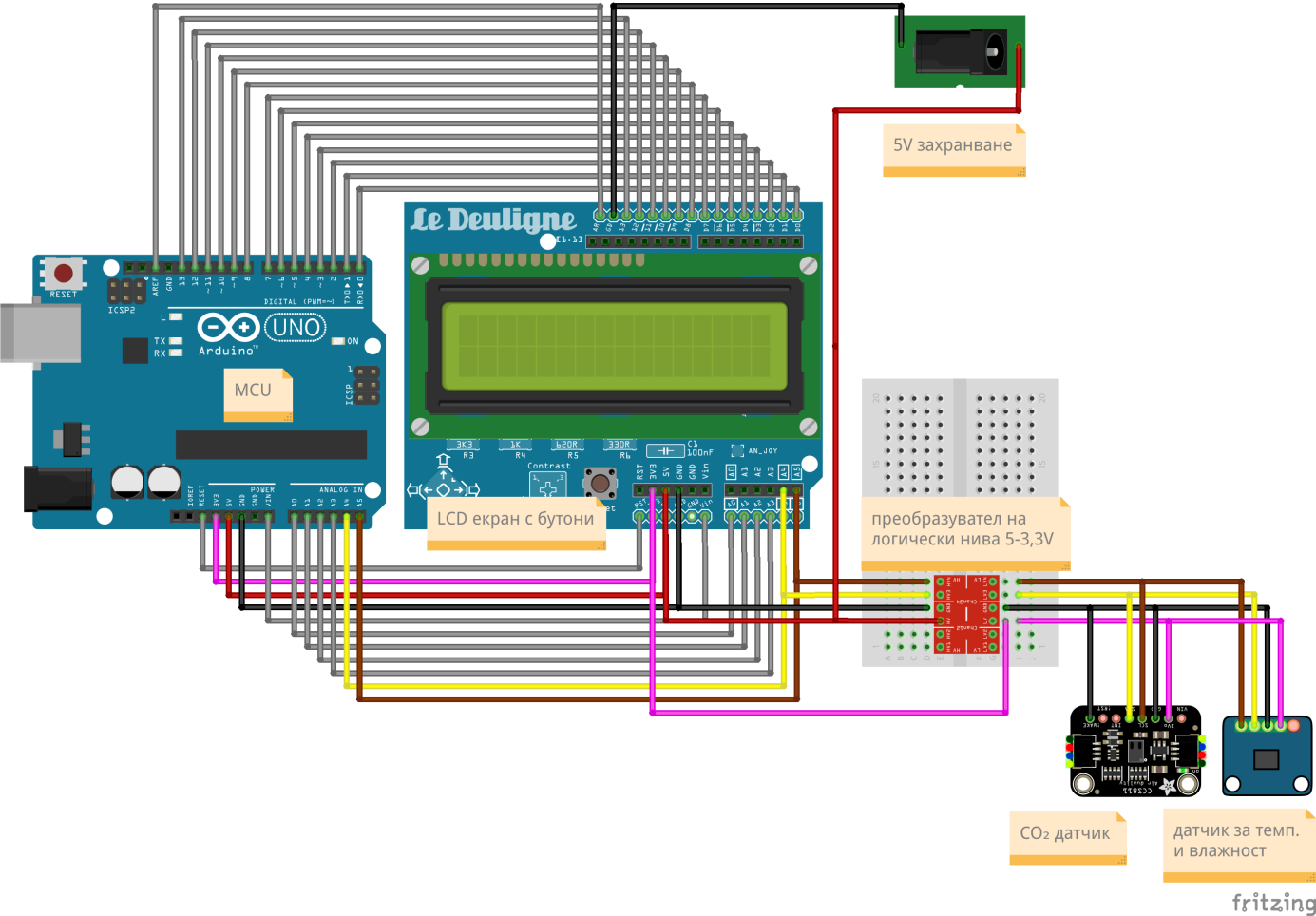
# Блокова схема



# Електрическа схема



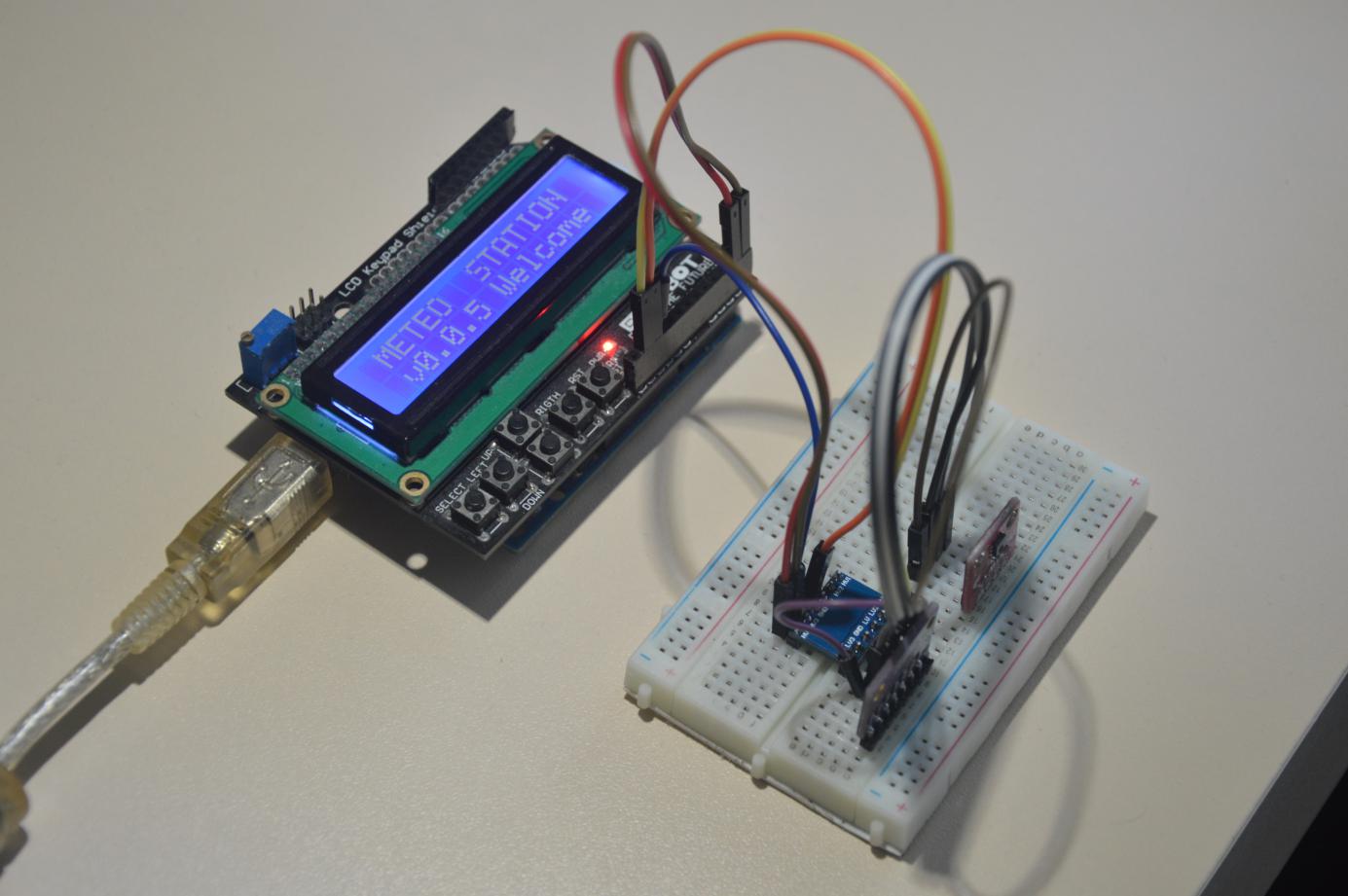
# Монтажна схема



# Списък съставни части

* Arduino UNO R3
* LCD TC1602 (DFRobot LCD Keypad Shield)
* 5 бутона (DFRobot LCD Keypad Shield)
* 4-канален конвертор на логически нива
* HTU21D I2C цифров датчик за температура и влажност
* CCS811 I2C цифров датчик за концентрация на CO2 във въздуха

# Прототип



# Изходен код

#include <Adafruit\_CCS811.h>

#include <LiquidCrystal.h>

#include <Adafruit\_HTU21DF.h>

int lcd\_key = 0;

int adc\_key\_in = 0;

float tempUpHold = 44.5;

float tempDownHold = -10.5;

float humiUpHold = 95.0;

float humiDownHold = 10.0;

int gasUpHold = 9000;

int gasDownHold = 0;

#define btnRIGHT 0

#define btnUP 1

#define btnDOWN 2

#define btnLEFT 3

#define btnSELECT 4

#define btnNONE 5

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // lcd init

Adafruit\_HTU21DF htu = Adafruit\_HTU21DF(); // htu21d init

Adafruit\_CCS811 ccs; // ccs811 init

// button handle method

int read\_LCD\_buttons() {

adc\_key\_in = analogRead(0); // read value from the sensor

// threshold manually adjusted

if (adc\_key\_in > 1000) return btnNONE;

if (adc\_key\_in < 50) return btnRIGHT;

if (adc\_key\_in < 250) return btnUP;

if (adc\_key\_in < 300) return btnDOWN;

if (adc\_key\_in < 500) return btnLEFT;

if (adc\_key\_in < 850) return btnSELECT;

return btnNONE;

}

void setup() {

Serial.begin(9600);

if (!ccs.begin()) {

while (1);

}

// gas sensor init

while (!ccs.available());

ccs.begin();

// htu init

htu.begin();

// lcd init (16x2)

lcd.begin(16, 2);

// splash screen

lcd.print(" METEO STATION ");

lcd.setCursor(0, 1);

lcd.print(" v1.0.0 Welcome ");

delay(2000);

lcd.clear();

Serial.println("Ready");

}

int selectedMenu = 0;

bool selectedOption = false; // because the display is only 16x2 I can get away

// with using just a bool variable

bool splashShown = false;

bool clearScreen = false;

bool exitSettings = false;

String typeAlert = " ";

bool tempAlert = false;

bool humiAlert = false;

bool gasAlert = false;

bool alertOverride = false;

int alertStart = 0;

int alertLength = 10;

void readButtons() {

lcd\_key = read\_LCD\_buttons(); // read the buttons

switch (lcd\_key) { // depending on which button was pushed, we perform an action

case btnRIGHT: {

if (gasAlert) {

gasAlert = false;

alertOverride = true;

alertStart = millis() / 1000;

}

if (humiAlert) {

humiAlert = false;

alertOverride = true;

alertStart = millis() / 1000;

}

if (tempAlert) {

tempAlert = false;

alertOverride = true;

alertStart = millis() / 1000;

}

switch (selectedMenu) {

case 0: {

break;

}

case 1: {

if (selectedOption) {

tempDownHold += 0.5;

} else {

tempUpHold += 0.5;

}

break;

}

case 2: {

if (selectedOption) {

humiDownHold += 0.5;

} else {

humiUpHold += 0.5;

}

break;

}

case 3: {

if (selectedOption) {

gasDownHold += 1;

} else {

gasUpHold += 1;

}

break;

}

case 4: {

if (!selectedOption) {

exitSettings = true;

}

break;

}

}

break;

}

case btnLEFT: {

switch (selectedMenu) {

case 0: {

break;

}

case 1: {

if (selectedOption) {

tempDownHold -= 0.5;

} else {

tempUpHold -= 0.5;

}

break;

}

case 2: {

if (selectedOption) {

humiDownHold -= 0.5;

} else {

humiUpHold -= 0.5;

}

break;

}

case 3: {

if (selectedOption) {

gasDownHold -= 1;

} else {

gasUpHold -= 1;

}

break;

}

case 4: {

if (!selectedOption) {

exitSettings = false;

}

break;

}

}

break;

}

case btnUP: {

switch (selectedMenu) {

case 0: {

break;

}

case 1: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 2: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 3: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 4: {

selectedOption = false;

break;

}

}

break;

}

case btnDOWN: {

switch (selectedMenu) {

case 0: {

break;

}

case 1: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 2: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 3: {

if (selectedOption) {

selectedOption = false;

} else {

selectedOption = true;

}

break;

}

case 4: {

selectedOption = false;

break;

}

}

break;

}

case btnSELECT: {

switch (selectedMenu) {

case 0: {

selectedMenu = 1;

splashShown = false;

break;

}

case 1: {

selectedMenu = 2;

splashShown = false;

break;

}

case 2: {

selectedMenu = 3;

splashShown = false;

break;

}

case 3: {

selectedMenu = 4;

splashShown = false;

break;

}

case 4: {

selectedMenu = 0;

splashShown = false;

clearScreen = true;

break;

}

}

break;

}

case btnNONE: {

break;

}

}

}

void selectionShow() {

lcd.setCursor(0, 0);

if (!selectedOption) {

lcd.print('>');

} else {

lcd.print(" ");

}

lcd.setCursor(0, 1);

if (selectedOption) {

lcd.print('>');

} else {

lcd.print(" ");

}

}

void status1() {

// gas sensor print

if (!ccs.readData()) {

lcd.setCursor(0, 0);

lcd.print(ccs.geteCO2());

lcd.print("ppm");

lcd.setCursor(10, 0);

lcd.print(ccs.getTVOC());

}

float h = htu.readHumidity();

// Read temperature as Celsius

float t = htu.readTemperature();

lcd.setCursor(0, 1);

lcd.print(t, 1);

lcd.print(char(223));

lcd.print('C');

lcd.setCursor(10, 1);

lcd.print(h, 1);

lcd.print("%");

}

void alertChecker() {

if (ccs.geteCO2() > gasUpHold || ccs.geteCO2() < gasDownHold) {

gasAlert = true;

typeAlert += "Gas ";

}

if (htu.readHumidity() > humiUpHold || htu.readHumidity() < humiDownHold) {

humiAlert = true;

typeAlert += "Hum. ";

}

if (htu.readTemperature() > tempUpHold || htu.readTemperature() < tempDownHold) {

tempAlert = true;

typeAlert += "Temp. ";

}

if (alertOverride && alertStart + alertLength == millis() / 1000) {

alertOverride = false;

alertStart = 0;

typeAlert = " ";

}

}

void alert() {

lcd.setCursor(0, 0);

lcd.print(typeAlert);

lcd.setCursor(0, 1);

lcd.print("Alert! > to clr");

}

void settings1() {

selectionShow();

lcd.setCursor(1, 0);

lcd.print("T" + String(char(223)) + " UP ");

lcd.print(tempUpHold, 1);

lcd.print(String(char(223)) + "C");

lcd.setCursor(1, 1);

lcd.print("T" + String(char(223)) + " DOWN ");

lcd.print(tempDownHold, 1);

lcd.print(String(char(223)) + "C");

delay(150);

}

void settings2() {

selectionShow();

lcd.setCursor(1, 0);

lcd.print("H% UP ");

lcd.print(humiUpHold, 1);

lcd.setCursor(1, 1);

lcd.print("H% DOWN ");

lcd.print(humiDownHold, 1);

delay(150);

}

void settings3() {

selectionShow();

lcd.setCursor(1, 0);

lcd.print("Gas UP ");

lcd.print(gasUpHold, 1);

lcd.setCursor(1, 1);

lcd.print("Gas DOWN ");

lcd.print(gasDownHold, 1);

delay(150);

}

void settings4() {

selectionShow();

lcd.setCursor(0, 0);

lcd.print("Exit? ");

/\*

if (exitSettings) {

lcd.print("YES");

}

else {

lcd.print("NO");

}

\*/

lcd.setCursor(1, 1);

lcd.print("Press > to exit");

clearScreen = true;

delay(150);

}

void loop() {

readButtons();

alertChecker();

if (exitSettings) {

selectedMenu = 0;

exitSettings = false;

}

if ((gasAlert || tempAlert || humiAlert) && !alertOverride) {

alert();

clearScreen = true;

typeAlert = "";

} else {

switch (selectedMenu) {

case 0: {

if (clearScreen) {

lcd.clear();

clearScreen = false;

}

if ((millis() / 1000) % 4 == 0) {

status1();

}

break;

}

case 1: {

if (!splashShown) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temperature");

lcd.setCursor(0, 1);

lcd.print("settings");

delay(1000);

lcd.clear();

splashShown = true;

}

settings1();

break;

}

case 2: {

if (!splashShown) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Humidity");

lcd.setCursor(0, 1);

lcd.print("settings");

delay(1000);

lcd.clear();

splashShown = true;

}

settings2();

break;

}

case 3: {

if (!splashShown) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Gas");

lcd.setCursor(0, 1);

lcd.print("settings");

delay(1000);

lcd.clear();

splashShown = true;

}

settings3();

break;

}

case 4: {

if (!splashShown) {

lcd.clear();

splashShown = true;

selectedOption = false;

}

settings4();

break;

}

}

}

}

# Заключение

Този проект може да намери приложение за наблюдение на контролирани среди, където отклонения в метеорологичните условия са недопустими. Може да се използва и в домашни условия, като инструмент за наблюдение на качеството на въздуха и да предупреждава при занижени параметри. Потребителите на този уред ще могат да да управляват въздуха, спрямо своите предпочитания за влажност, топлина и наситеност на CO2.