**ЛАБОРАТОРНА РОБОТА № 7**

**Створення пристрою вимірювання концентрації чадного газу у повітрі з відображенням у хмарному сервісі ThingSpeak.**

Мета: Розробити програмний код для передавання вимірів концентрації чадного газу у повітрі аналоговим датчиком MQ-2 до хмарного сервісу ThingSpeak з використаннямплати NodeMCUv3 – ESP-12E***.***

**Короткі теоретичні відомості**

Датчик серії MQ-2 використовує невеликий нагрівач всередині з електрохімічним датчиком для вимірювання різних видів комбінації газів. Датчик газу MQ2 змінює значення внутрішнього опору RS відповідно до концентрації газу. Схематично датчик зображено на рис. 1. Послідовно з сенсором включається змінний резистор навантаження , який забезпечує регулювання чутливості датчика, опір якого змінюється від 2кОм до 47 кОм.

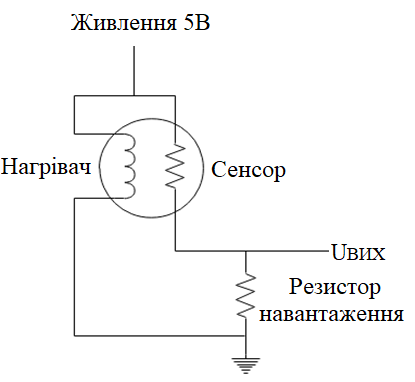


Рисунок 1 – Схема датчику MQ-2

Нагрівальний елемент всередині датчика MQ-2 потребує підключення зовнішнього джерела живлення з напругою 5 В та потужністю приблизно 1Вт. Для визначення внутрішнього опору сенсора необхідно врахувати, що вихід модуля датчика фактично являє дільник напруги джерела живлення , який створюється опором датчика та послідовним включенням резистора RL. Оскільки значення резистору навантаження RL відоме і стале, то на основі виміряного значення вихідної напруги датчика, можна розрахувати опір датчика RS:

|  |  |
| --- | --- |
|  | (1) |

На рис. 2 зображено графік, що показує концентрацію газу в частці на мільйон (ppm) від відношення опорів RS/R0, де R0 – опір датчика у чистому повітрі. Отже для визначення концентрації чадного газу необхідно визначити значення опору R0 датчика у чистому повітрі.

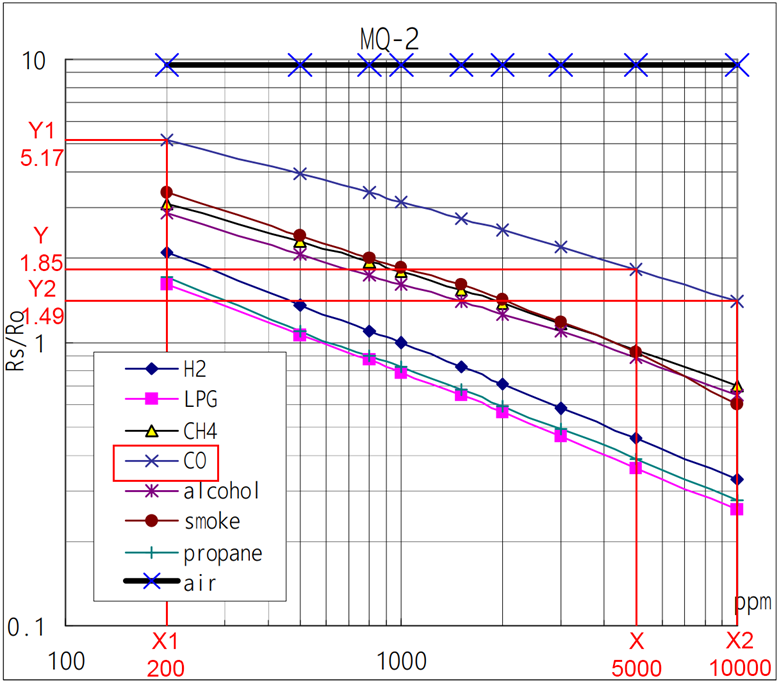


Рисунок 2 – Характеристики датчику MQ-2

Для цього за допомогою мультиметру виміряли опір резистору навантаження на платі датчика MQ-2, який становив Далі датчик було підключено до джерела живлення та залишено на прогрівання у середовищі з чистим повітрям протягом однієї години. Через одну годину було виміряно напругу на виході датчику, яка становила Далі було розраховано опір сенсора RS за виразом (1):

З рис. 2 видно, що у чистому повітрі Тоді значення опору R0 у чистому повітрі:

|  |  |
| --- | --- |
|  |  |

При відомому значенні опору датчику R0 у чистому повітрі можна, згідно рис. 2, визначити значення концентрації газу, яке відповідає розрахованому відношенню виміряного значення внутрішнього опору датчика до опору датчика у чистому повітрі .

Згідно (1) для розрахунку виміряного значення внутрішнього опору датчика RS необхідно виміряти поточне значення напруги на виході датчика. Для вимірювання напруги підключимо аналоговий вихід датчика газу MQ-2 до входу A0 вбудованого 10-ти бітного АЦП плати NodeMCUv3 – ESP-12E. Тоді поточне значення вихідної напруги датчика можна розрахувати за формулою:

|  |  |
| --- | --- |
|  | (2) |

де – цифровий код на виході АЦП;

– опорна напруга АЦП, яка дорівнює напрузі ;

– розрядність АЦП.

Графік залежності концентрації газу від відношення на рис. 2 подано у логарифмічному масштабі, тобто :

|  |  |
| --- | --- |
|  | (3) |

де – відношення опорів *RS*/*R0*;

– значення концентрації газу в ppm;

– нахил лінії;

– точка, де графік функції перетинає вісь Y.

Нахил лінії розраховується за виразом:

|  |  |
| --- | --- |
|  | (4) |

де *x1* – Х координата точки початку лінії на графіку;

*y1* –Y координата точки початку лінії на графіку;

*x2* –Х координата точки кінця лінії на графіку;

*у2* –Y координата точки кінця лінії на графіку.

Згідно рис. 2 початкова точка (*x1,y1*) має координати (200; 5,17), а координати кінцевої точки (*x2,y2*) становлять (10000;1,49). Тоді, згідно (4):

Після розрахунку нахилу лінії можна визначити з виразу (3) значення :

|  |  |
| --- | --- |
|  | (5) |

де – координати точки, що відповідає середині графіку.

Координати точки, що відповідає середині графіку (рис. 2), становлять (5000; 1,85). Тоді згідно (5):

Отже, як передує з виразу (3), а виміряне значення концентрації газу у ppm може бути визначено згідно виразу:

|  |  |
| --- | --- |
|  | (6) |

**Виконання роботи**

1. **Налаштовано програмне середовище Arduino IDE на роботу с платою ESP8266.**

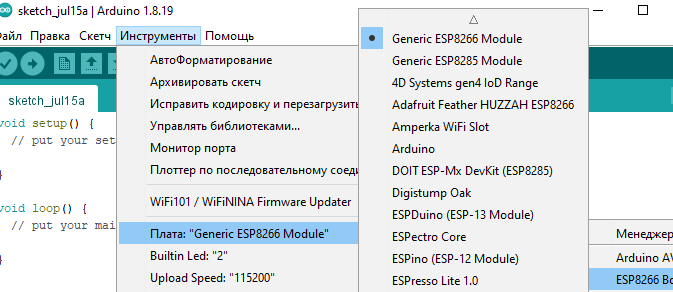


Рис 1.1

1. Завантажено програмний код «ThingSpeak\_temp\_hum.ino», який був раніше розроблений для передачі даних вимірів датчика DHT11 у хмарний сервіс ThingSpeak. В акаунті та створено новий канал для виведення вимірів СО у ppm. Змінено у коді ID каналу.
2. Додали до коду рядки, що визначають аналоговий вхід, до якого підключено датчик MQ-2 та типи змінних, що використовуються у розрахунках концентрації газу

#define adcPin A0 // Пін входу АЦП

float VCC = 5; // Напруга живлення датчику, В

float Vref = 5; // Опорна напруга АЦП, В

float RL = 2; // Номінал резистору RL, кОм

float RO = 7.2; // Відкаліброване значення R0

float curve[2] = { -0.32, 1.44}; // Розраховані значення функції для CO

int ppm; // Змінна для зберігання значень у ppm.

1. У **void loop()** додамо рядки, які дозволяють читати дані аналогового входу A0 та визначати значення виміряної напруги на виході датчика згідно виразу (2):

float = 0; // Змінна для зберігання значень

= analogRead(adcPin)\*Vref / 1024; // Розрахунок напруги на виході датчику.

На основі виміряного значення вихідної напруги датчика розрахуйте опір датчика RS згідно виразу (1) та відношення RS/RO. Використовуючи отримане значення RS/RO, розрахуйте виміряне значення концентрації газу CO у ppm згідно виразу (6) та використаємо функцію:

ppm = pow(10, (log10(ratio) - curve[1]) / curve[0]);

якщо отримане значення більше ніж 10000, його доцільно обмежити максимальним значенням, а саме:

if (ppm > 10000) { // Якщо розраховане ppm більше ніж 10000

ppm = 10000; // Обмежити до 10000 згідно даташиту

1. Додамо рядок для виведення розрахованого значення концентрації газу CO у створений канал хмарного сервісу ThingSpeak
2. Проведемо компіляцію створеного скетчу

#include <xc.h>

#include <stdint.h>

#include <math.h>

#include "supporing\_cfile/lcd.h"

#include "supporing\_cfile/adc.h"

#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = ON // Brown-out Reset Enable bit (BOR enabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

/\*

Program Flow related definition

\*/

#define gas\_detect PORTDbits.RD5

#define gas\_Detect\_Pin\_Direction TRISDbits.TRISD5

#define FIRST\_LINE 0x80

#define SECOND\_LINE 0xC0

#define RL\_VALUE (10) //define the load resistance on the board, in kilo ohms

#define RO\_VALUE\_CLEAN\_AIR (9.83) //(Sensor resistance in clean air)/RO,

//which is derived from the chart in datasheet

float MQ6\_curve[3] = {2.3,0.30,-0.41}; //two points from LPG curve are taken point1:(200,1.6) point2(10000,0.26)

//take log of each point (lg200, lg 1.6)=(2.3,0.20) (lg10000,lg0.26)=(4,-0.58)

//find the slope using these points. take point1 as reference

//data format:{ x, y, slope};

float Ro = 0; //Ro is initialized to 10 kilo ohms

#define \_XTAL\_FREQ 20000000 //20 Mhz

// System related definitions

void system\_init(void);

void introduction\_screen(void);

void clear\_screen(void);

//int GetPercentage(float rs\_ro\_ratio, float \*pcurve);

int gas\_plot\_log\_scale(float rs\_ro\_ratio, float \*curve);

float read\_mq();

float calculate\_resistance(int raw\_adc);

float SensorCalibration();

void main() {

system\_init();

clear\_screen();

lcd\_com(FIRST\_LINE);

lcd\_puts("Calibrating....");

Ro = SensorCalibration();

//clear\_screen();

lcd\_com(FIRST\_LINE);

lcd\_puts("Done! ");

//clear\_screen();

lcd\_com(FIRST\_LINE);

lcd\_print\_number(Ro);

lcd\_puts(" K Ohms");

\_\_delay\_ms(1500);

gas\_detect = 0;

while(1){

if(gas\_detect == 0){

lcd\_com(FIRST\_LINE);

lcd\_puts("Gas is present ");

lcd\_com(SECOND\_LINE);

lcd\_puts ("Gas ppm = ");

float rs = read\_mq();

float ratio = rs/Ro;

lcd\_print\_number(gas\_plot\_log\_scale(ratio, MQ6\_curve));

\_\_delay\_ms(1500);

clear\_screen();

}

else{

lcd\_com(FIRST\_LINE);

lcd\_puts("Gas not present ");

//lcd\_com(SECOND\_LINE);

// lcd\_print\_number(gas\_plot\_log\_scale(read\_mq()/Ro, MQ6\_curve));

}

}

}

void system\_init(){

TRISB = 0; // LCD pins set to out.

gas\_Detect\_Pin\_Direction = 1; //Configure RD0 as input

lcd\_init();

ADC\_Init();

introduction\_screen();

//dht11\_init();

}

/\*

This Function is for Clear screen without command.

\*/

void clear\_screen(void){

lcd\_com(FIRST\_LINE);

lcd\_puts(" ");

lcd\_com(SECOND\_LINE);

lcd\_puts(" ");

}

/\*

This Function is for playing introduction.

\*/

void introduction\_screen(void){

lcd\_com(FIRST\_LINE);

lcd\_puts("Welcome to");

lcd\_com(SECOND\_LINE);

lcd\_puts("circuit Digest");

\_\_delay\_ms(1000);

\_\_delay\_ms(1000);

clear\_screen();

lcd\_com(FIRST\_LINE);

lcd\_puts("MQ6 Sensor");

lcd\_com(SECOND\_LINE);

lcd\_puts("with PIC16F877A");

\_\_delay\_ms(1000);

\_\_delay\_ms(1000);

}

/\*

\* Sensor Related Functions

\*/

float SensorCalibration(){

int count; // This function assumes that sensor is in clean air.

float val=0;

for (count=0;count<50;count++) { //take multiple samples and calculate the average value

val += calculate\_resistance(ADC\_Read(0));

\_\_delay\_ms(500);

}

val = val/50;

val = val/RO\_VALUE\_CLEAN\_AIR; //divided by RO\_CLEAN\_AIR\_FACTOR yields the Ro

//according to the chart in the datasheet

return val;

}

float read\_mq()

{

int count;

float rs=0;

for (count=0;count<5;count++) { // take multiple readings and average it.

rs += calculate\_resistance(ADC\_Read(0)); // rs changes according to gas concentration.

\_\_delay\_ms(50);

}

rs = rs/5;

return rs;

}

float calculate\_resistance(int adc\_channel)

{ // sensor and load resistor forms a voltage divider. so using analog value and load value

return ( ((float)RL\_VALUE\*(1023-adc\_channel)/adc\_channel)); // we will find sensor resistor.

}

int gas\_plot\_log\_scale(float rs\_ro\_ratio, float \*curve)

{

return pow(10,(((log(rs\_ro\_ratio)-curve[1])/curve[2]) + curve[0]));

}

**Висновок:** В ході даної лабораторної роботи було створено пристрій вимірювання концентрації чадного газу у повітрі з відображенням у хмарному сервісі ThingSpeak. Розроблено програмний код для передавання вимірів концентрації чадного газу у повітрі аналоговим датчиком MQ-2 до хмарного сервісу ThingSpeak з використанням плати NodeMCUv3 – ESP-12E.