

Arduino Code

For Carbon Dioxide detection in air

```
/*
// This code communicates with the MQ135 air quality sensor. The sensor is
supposed to preheat for 2 mins before taking readings
// Once the code runs, it prints out the concentration of detected gases
in ppm on a serial monitor and a LCD 20 x 4 screen
//An alarm system (LED light) is also set to print out messages saying if
the air is of a good quality relying on a predefined threshold value
*/
// digital output value is converted to ppm value using CO2 gas as
parameter
#include "MQ135.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h> //Header file for LCD

LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to x27 for a 16 chars
and 2 line display

#define led          9                //led on pin 9
const int gas_pin = A0;              //analog feed from MQ135
MQ135 gasSensor = MQ135(gas_pin);

void setup(){

    lcd.init();                      // initialize the lcd
    lcd.begin(16,2);                 // consider 16 chars + 2 lines lcd
    lcd.backlight();                 // illuminate to produce visible reading
    lcd.clear();                     // clear lcd
    lcd.setCursor(4,0);              //set cursor of lcd to 1st row and 5th
column
    lcd.print("Group L");            // print as a sentence on lcd

    pinMode(gas_pin,INPUT);          //MQ135 analog feed set for input
    pinMode(led,OUTPUT);             //led set for output
    Serial.begin(9600);              //serial comms for debugging
}

void loop(){
    float ppm = gasSensor.getPPM();
    Serial.println(ppm);              // print ppm on serial monitor
    delay(1000);
    lcd.clear();                     // clear lcd
    lcd.setCursor(0,0);              // set cursor of lcd to 1st row and 1st
column
    lcd.print("Air Quality: ");      // print as a sentence on lcd
}
```

```
    lcd.print(ppm);           // print value of MQ135
    if(ppm>999){              //if co2 ppm > 1000
        digitalWrite(led,HIGH); //turn on led
        lcd.setCursor(2,1);    // set cursor of lcd to 2nd row and 3rd
column
        lcd.print("AQ Level BAD"); //print as a sentence on lcd
    }
    else{
        digitalWrite(led,LOW); //turn off led
        lcd.setCursor(1,1);    // set cursor of lcd to 2nd row and 2nd
column
        lcd.print ("AQ Level Good"); // print as a sentence on lcd
    }
}
```

****For noise disturbance detection in the environment****

```
/* This code is meant to monitor the sound intensity using LM393 sensor
connected to Arduino UNO board.
//The used sensor has only a digital output. Therefore, the number of times
the sensor detects a sound is summed up over a sampling time called
"SAMPLE_TIME".
//Then the sum called "sampleBufferValue" is printed on a Serial Monitor
(laptop), and visualized with the Serial Plotter.
// The code allows to communicate with a LED in order to provide a visual
alarm if the "sampleBufferValue" surpasses a preset Threshold "Threshold"
*/

// 0 means silence and 1 means noise

const int OUT_PIN = 12;           // The OUTPUT of the sound sensor is
connected to the digital pin D12 of the Arduino
const int SAMPLE_TIME = 10;      // The sampling time in milliseconds, can
be set differently if required
const int Threshold = 900;       // Threshold on cumulative counts for LED
switching ON, the value has been optimized with respect to the used sampling
time (900 digital outputs  $\approx$  90 dB from "Schall")

unsigned long millisCurrent;      // current time
unsigned long millisLast = 0;     //previous time
unsigned long millisElapsed = 0; // difference between current time and
previous time (time interval)

int sampleBufferValue = 0;        // initiate the sum of digital outputs
over the sampling time
int led = 8;                     // LED on pin 4 of Arduino
int dB = 0;                      //initiate sound intensity dB value
```

```
void setup() {  
    Serial.begin(9600);           //Arduino starts serial communication with  
    baud rate 9600  
    pinMode(led,OUTPUT);         // the LED is connected as output for alarm  
    purpose  
}  
  
void loop() {  
    millisCurrent = millis();           //the current time is  
    assigned to the dedicated variable  
    millisElapsed = millisCurrent - millisLast; //the elapsed time is updated  
    if(digitalRead(OUT_PIN) == HIGH){    //HIGH means noise  
        sampleBufferValue++;           //increments the sum variable  
        by 1  
    }  
    if (millisElapsed > SAMPLE_TIME) {    //if the elapsed time surpasses  
    the sampling time, print the sampleBufferValue and test threshold for alarm  
        dB = 0.0666 *(sampleBufferValue) + 30.223; //linear regression to  
        calculate the decibel value based of the rough calibration of the sensor  
        response  
        Serial.println(dB);           // print decibel values on the  
        Serial Monitor  
        Serial.print("dB");           // print sound unit decibel  
        if (sampleBufferValue > Threshold) {    // test if the threshold is  
        surpassed  
            digitalWrite(led, HIGH);           //blink LED 2 ms ON and 1 ms OFF  
            delay(2);  
            digitalWrite(led, LOW);  
            delay(1);  
        }  
        digitalWrite(led, LOW);           // the LED is turned off to be  
        ready for the next sample  
        sampleBufferValue = 0;           // re-initialization of the  
        sampleBufferValue variable for the new sampling time  
        millisLast = millisCurrent;       // update the previous time to be  
        the start for the next sample  
    }  
}
```

From:

<https://student-wiki.eolab.de/> - **HSRW EOLab Students Wiki**

Permanent link:

<https://student-wiki.eolab.de/doku.php?id=amc2021:code:start&rev=1630616643>

Last update: **2023/01/05 14:38**

