Building a CO2 meter for ESP32 – base setup

By Pedro M.J. Wyns – pedro.wyns@thomasmore.be - 20220316

A good lead with info on the sensor basics is this article:

http://ihormelnyk.com/mh-z19_co2_meter

Unfortunately he uses the quasi analog PWM output to display the value. I prefer the serial data link with the device. So below my setup and an example program, *soon to be updated towards ESP32 with a blynk remote link to a cellphone and alarms via Telegram.*

Data sheet:

The data sheet of the **sensor MH-Z19** can be found here:

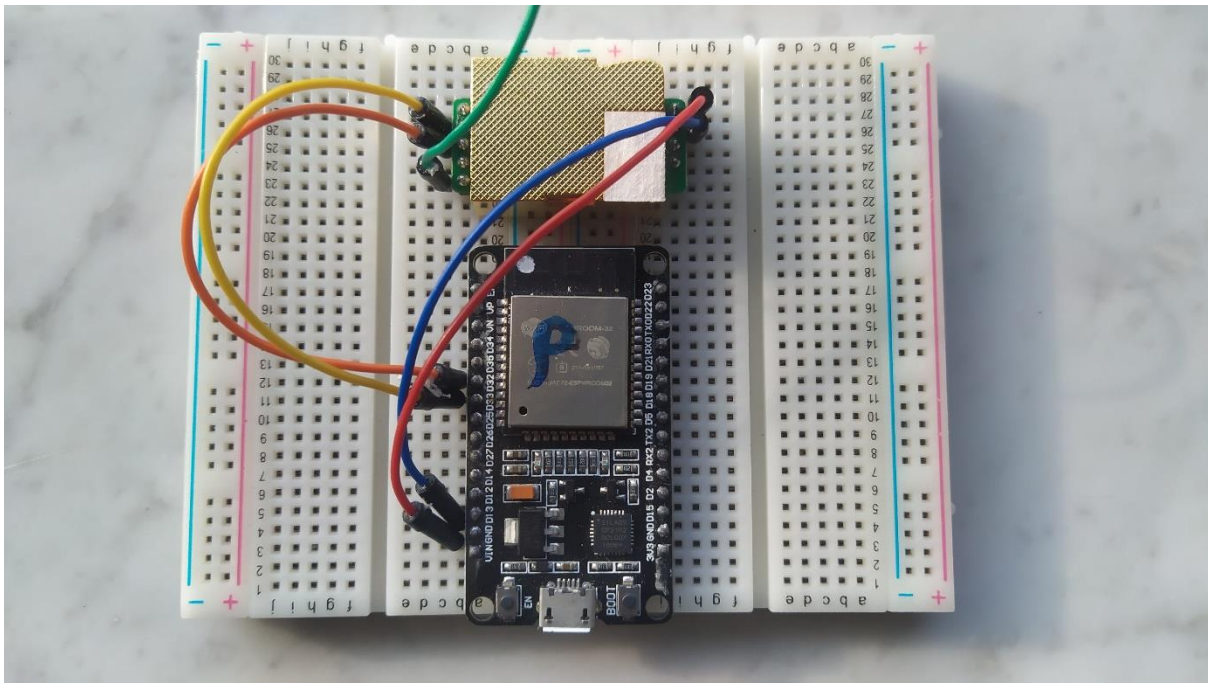
https://www.winsen-sensor.com/d/files/PDF/Infrared%20Gas%20Sensor/NDIR%20CO2%20SENSOR/MH-Z19%20CO2%20Ver1.0.pdf

Pinout:

PIN	Description
Pin 6	Vin (voltage input)
Pin 7	GND
Pin 1	Vout (output voltage 3.3V, output current lower than 10mA)
Pin 9	PWM
Pin 5	HD (zero calibration, low level above 7 seconds) (Factory Reserved)
Pin 2	UART (RXD) 0~3.3V digital input
Pin 3	UART (TXD) 0~3.3V digital output
Pin 4	SR (Factory Reserved)
Pin 8	AOT (Factory Reserved)

Connections of the sensor on the ESP32:

Red is 5V to feed the sensor, blue is ground, yellow and orange for datatransmission, normally 3,3V but according to data sheet 5V tolerant. GPIO 32 and 33 are used here. Green wire is pin 5 used for manual calibration (by grounding it for 7 seconds or more)



Test program: (adapted from the library MHZ19.h by Jonathan Dempsey jdwifwaf@gmail.com examples page)

```
// test program by Pedro M.J. Wyns from Jonathan Dempsey <jdwifwaf@gmail.com> library example adapted for ESP32

//#include <Arduino.h>
#include "MHZ19.h"
#include <SoftwareSerial.h>                    // Remove if using HardwareSerial

#define RX_PIN (32)                            // ESP32 GPIO pin which the MHZ19 Tx pin is attached to
#define TX_PIN (33)                            // ESP32 GPIO pin which the MHZ19 Rx pin is attached to
#define BAUDRATE 9600                          // Device to MH-Z19 Serial baudrate (should not be changed)

MHZ19 myMHZ19;                                 // Constructor for library
SoftwareSerial mySerial(RX_PIN, TX_PIN);       // (Uno example) create device to MH-Z19 serial

unsigned long getDataTimer = 0;

void setup()
```

```
{

  Serial.begin(9600);                          // Device to serial monitor feedback

  mySerial.begin(BAUDRATE);                    // (Uno example) device to MH-Z19 serial start
  myMHZ19.begin(mySerial);                     // *Serial(Stream) refence must be passed to library begin().

  myMHZ19.autoCalibration();                   // Turn auto calibration ON (OFF autoCalibration(false))
}

void loop()
{
  if (millis() - getDataTimer >= 5000)
  {
    int CO2;

    /* note: getCO2() default is command "CO2 Unlimited". This returns the correct CO2 reading even
    if below background CO2 levels or above range (useful to validate sensor). You can use the
    usual documented command with getCO2(false) */

    CO2 = myMHZ19.getCO2();                    // Request CO2 (as ppm)

    Serial.print("CO2 (ppm): ");
    Serial.println(CO2);

    int8_t Temp;
    Temp = myMHZ19.getTemperature();           // Request Temperature (as Celsius)
    Serial.print("Inside Sensor Temperature (C): ");
    Serial.println(Temp);

    getDataTimer = millis();
  }
}
```

Calibration of the sensor:

The sensor has an autocalibration routine that sets after some considerable time the lowest measured values towards 400 ppm.

400 ppm is the value op CO2 in open uncontaminated air.

You can speedup things by taking the sensor outside and grounding pin 5 (green wire on picture) for more than 7 second. Then the sensor will be calibrated to 400. It will keep the auto calibration running to deal with (considerable) sensor drift.

The temperature is not the real outdoor temperature but the temperature inside the housing only used for sensor calibration purposes so not interesting.

General remarks:

The MH-Z19C is a sensor made for 24/7 use. So readings during the first hours are not reliable. Best is to let it run for a few hours outside and then do a manual calibration. It would be a good idea to run pin 5 to an ESP32 output so you can do a remote calibration too via Blynk.