Microprocessor Lab Project

Helia Shakeri

January 3, 2025

Abstract

In this project, a gas sensor was programmed and calibrated.

1 Sensor to Micro

To connect the module of the sensor to the microprocessor, 3 wires were used. The first connected the VCC input of the module to the 3.3V output of the micro, the second connected the ground pins, and the last connected the AO (Analog Output) of the sensor to the PA4 pin of the micro. When the micro was connected to the laptop two LEDs, the power LED and the digital output LED were turned on. The digital output is turned on when the concentration reaches a level determined by a potentiometer on the module, which was left untouched because we only needed the analog output.

2 Pinout & Configuration

The clock is set to internal and the debugger to Serial Wire. Then the PA15 pin is set to `GPIO_Output` to drive LED1 and channel IN4 of ADC1 was ticked on to receive information from the sensor. No other setting was changed.

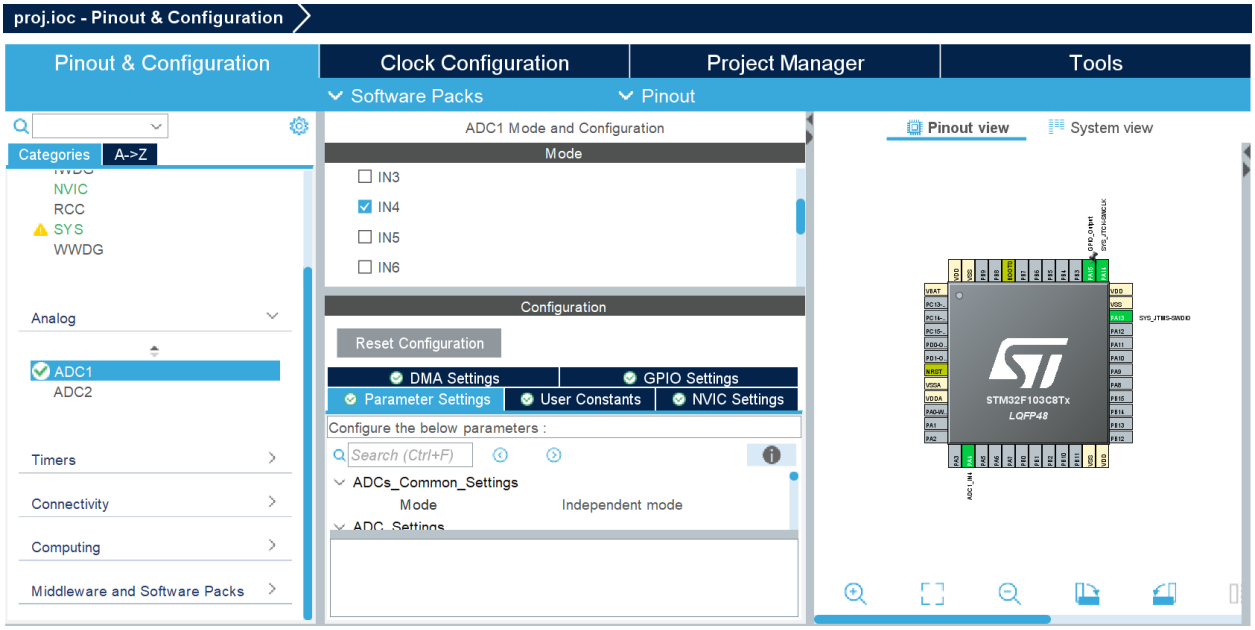

Figure 1: Pinout

3 Code & Calibration

I decided to use the same moving average filter than I used in the 5th experimented and explained in its report, so I copied the files for the filter function to my project folder and included the `moving_average.h` file in the beginning of the code. The `math.h` library was also included because the calibration necessitated the use of the `pow()` function from this library.

As the variables needed to be monitored by the CubeMonitor software, they were declared globally and outside the while loop, as shown in figure 2. The supply voltage V_{CC} and the load resistance R_L were interred from the datasheet but R_0 is the sensor's resistance at normal conditions which was found from observation of the resistance in room conditions.

```
/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;

/* USER CODE BEGIN PV */

uint16_t raw;
FilterTypeDef filterStruct;
float Vcc = 3.3, Rl = 10000, R0=11500;
float voltage, temp, Rs, concentration;

/* USER CODE END PV */
```

Figure 2: The variables

The code in the while loop is shown in figure 3. The first lines start the ADC and wait until the conversion is finished. Then the value from the ADC is copied onto the variable

`raw` and sent to the moving average function, which outputs the average of 10 consecutive samples in the variable `temp`. To turn the ADC output to the analog voltage reported by the module, `temp` was divided by the ADC's number of bits and then multiplied by the supply voltage.

The next step in the calibration process is finding the resistance of the sensor using the relations below:

$$V_{out} = V_{CC}\frac{R_L}{R_s + R_L} \quad \Rightarrow \quad R_s = R_L\frac{V_{CC} - V_{out}}{V_{out}}$$

To find the concentration from the sensor resistance, I used the graph in figure 4, taken from the datasheet. As I was using a lighter to stimulate the sensor, which most typically releases LPG gas, I used the first and last points of the LPG curve in the graph and found the logarithmic relation between the ratio of R_s to R_0 and concentration, as shown in the equation below. To find R_0 I ran the incomplete code that only calculated the sensor resistance and watched the resistance on CubeMonitor and found that $11.5\,k\Omega$ is an appropriate value for the sensor's resistance at normal conditions.

$$\log\left(\text{Concentration}\right) = m.\log\left(\frac{R_s}{R_L}\right) + b \quad \Rightarrow \quad \text{Concentration (ppm)} = \left(\frac{R_s/R_L}{23.93}\right)^{-2.136}$$

Then for the last part, I chose a threshold of 1000 ppm and the code turns on LED1 if this threshold is passed (which only happens after moments of holding the lighter gas in front of the sensor) and remains off if it's not.

```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
      raw = HAL_ADC_GetValue(&hadc1);
      temp = Moving_Average_Compute(raw, &filterStruct);
      voltage = temp/4095*Vcc;
      Rs = Rl*(Vcc-voltage)/voltage;
      concentration = pow((Rs/R0/23.93),-2.136);
      if (concentration>1000){
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);
      }
      else {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);
      }
      HAL_Delay(500);
  }
  /* USER CODE END 3 */
}
```

Figure 3: The while loop

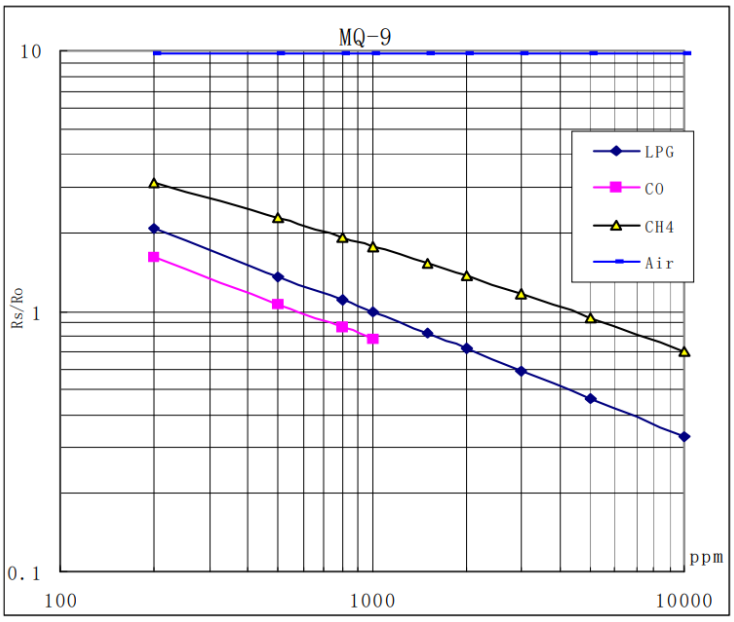

Figure 4: The sensitivity curve from the datasheet

4 CubeProgrammer

First, I changed the project's settings to produce the `.bin` binary file necessary for programming the microprocessor, and then I selected `Build Project`. In the CubeProgrammer software, after connecting the micro to my laptop, I selected `Connect` , downloaded the binary file from my project onto the board, and disconnected it again.

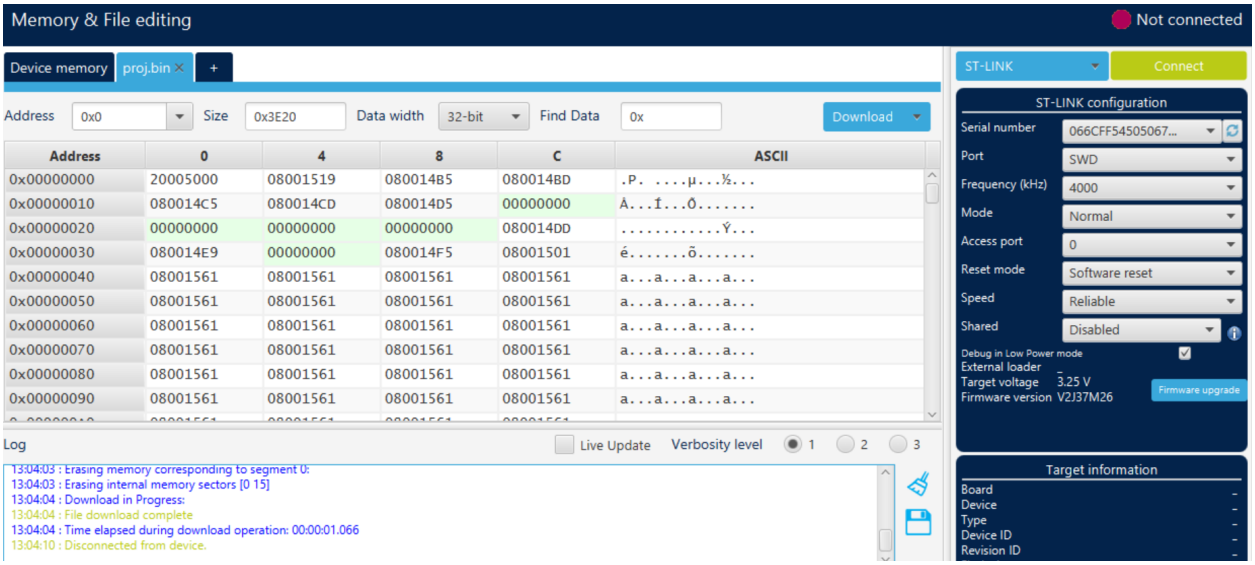

Figure 5: CubeProgrammer

5 CubeMonitor

In the CubeMonitor, first I selected the variables by changing the folder path to my project and selecting the `.elf` file from that folder. Then I chose the variable `concentration` and deployed the monitor, opened the dashboard, and started the acquisition. Figure 6 shows the concentration of lighter gas when the lighter is held near the sensor for a moment and figure 7 shows that LED1 is turned on at that moment, so the outcomes of the project are met.

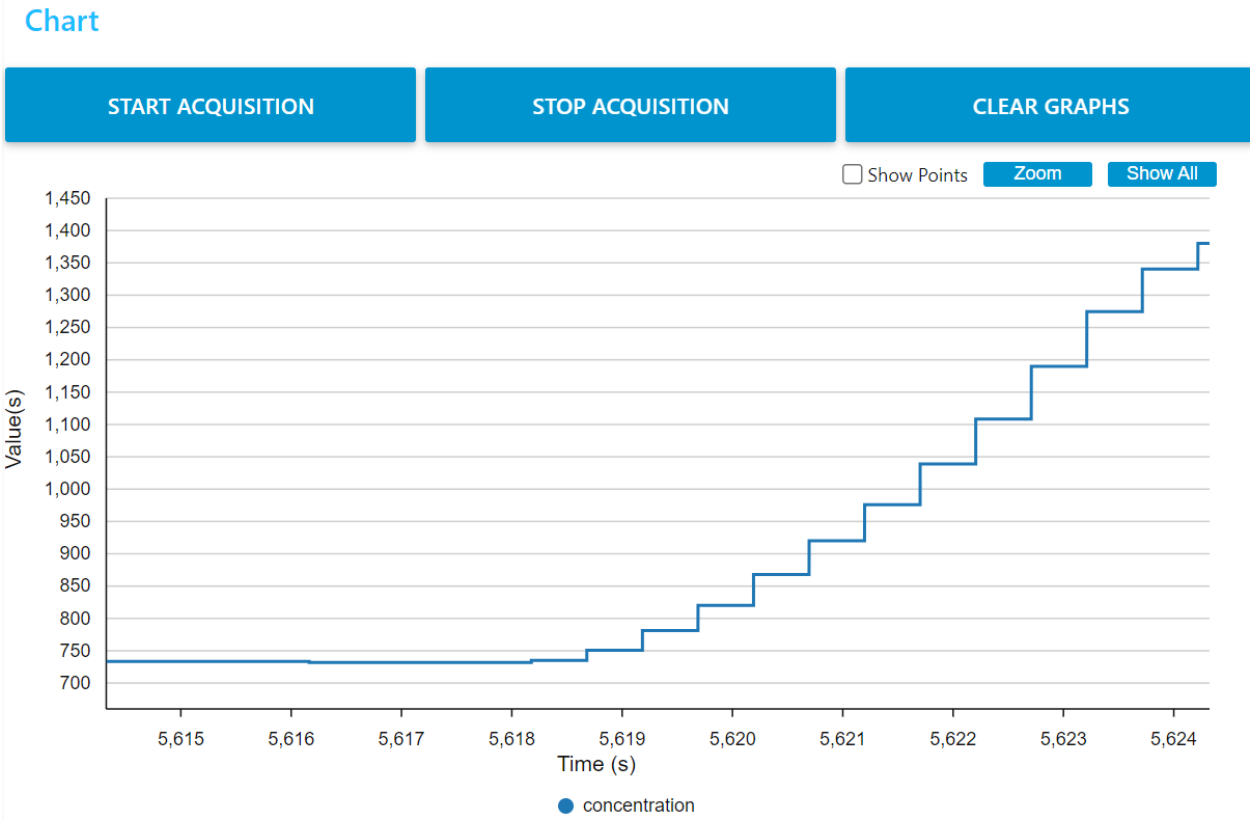

Figure 6: The concentration of LPG

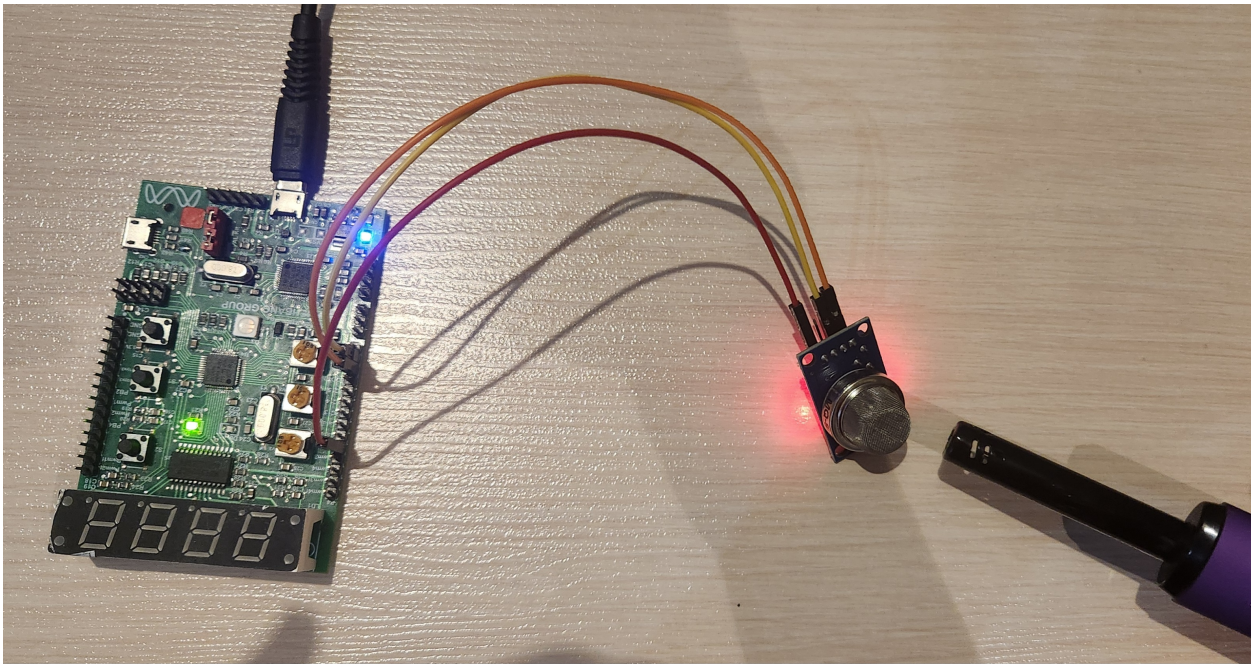

Figure 7: LED1 is turned on.