1. Subject name: Sonda Lambda

Sandu Elena
elenasandu1994@yahoo.com

Gavril Bogdan Alexandru
gavril.bogdan.alexandru@gmail.com

2. Summary:

Our project proposes to simulate the functionality of a Sonda Lambda sensor with a gas gas sensor. The project displays on a Graphic LCD a value which represents the level of gas from in the air and a message based on the value obtained. If the message is negativ one LED will blink to signal this.

3. Introduction:

The Lambda probe is a sensor located on the exhaust pipe and connected to the ECU, which consists of an electric current conductor whose intensity varies depending on the amount of oxygen passing through the probe. Inside, there is a porous ceramic material made of zirconium dioxide (ZrO2).

The current intensity through the zirconium plate varies depending on the number of oxygen molecules crossing the ceramic material. Because the probe works best only at high temperatures, "cold", until the exhaust gases reach temperatures of 400-500 oC, the probe is heated by a resistance inside it, after which the heat will be supplied even by the temperature of the exhaust gases .

Because one of the major sources of pollution is represented by combustion gases emitted by the engines in the atmosphere, the builders have set up systems to limit pollutant emissions.

In principle, the probe constantly measures the amount of oxygen in the exhaust gases and sends the signal as a voltage to the engine control unit. The ECU uses the signals received from the probe to adjust the mixture to obtain the ideal mix: 14.8kg air with 1kg of gasoline, for which the so-called Lambda factor is equal to 1. The sensor output values vary between 0.2 V (poor mix) and 0.8 V (rich mix), the ideal variation being around 0.45 V Optimizing the mix ensures maximum efficiency and lifetime of the catalyst.

4. Solution description:

Resurces:

Hardware:

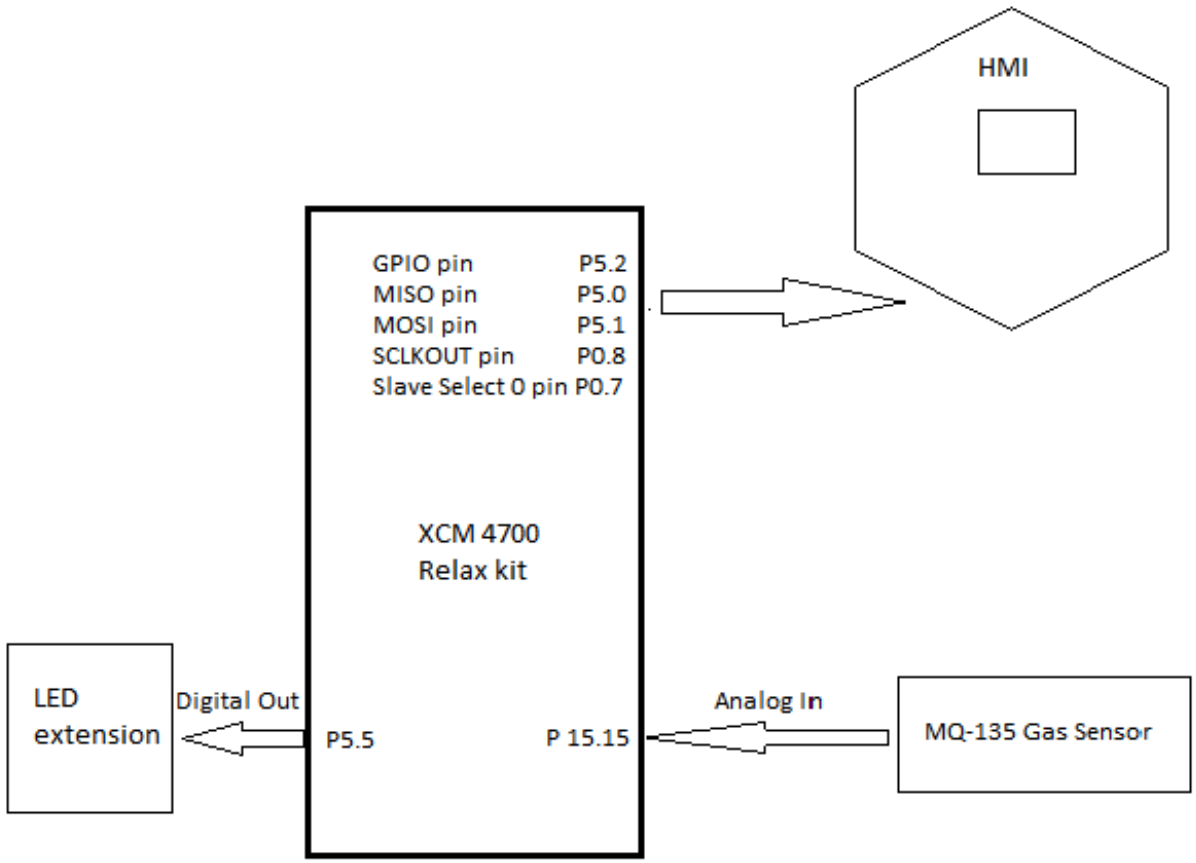
- XMC4700 relax kit
- HMI board
- Gas sensor MQ135
- LED
- breadboard
- connection cables

Software:

- Infineon DAVE 4.3.2

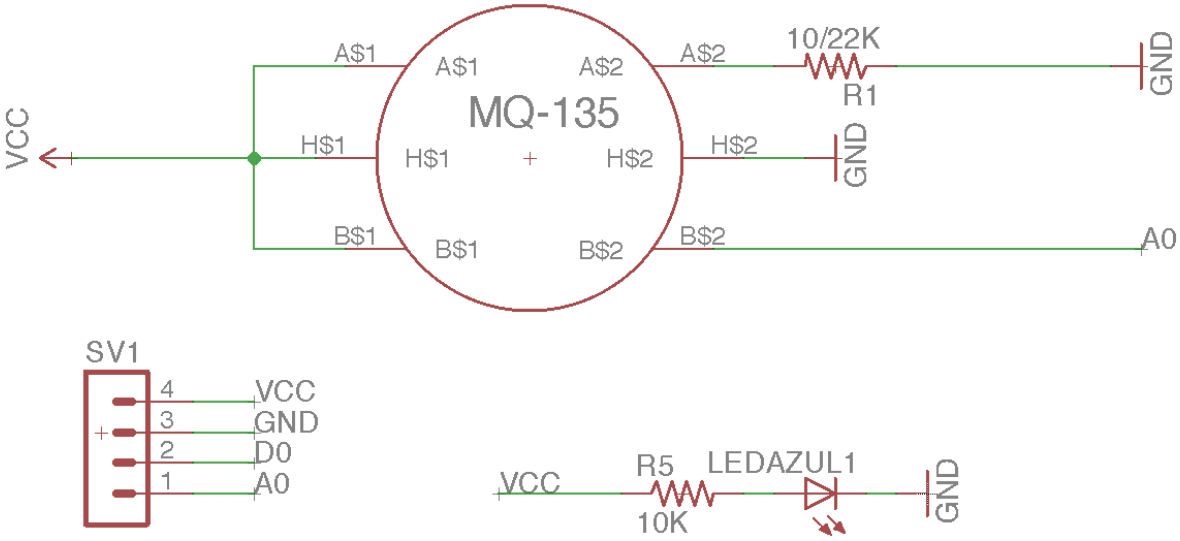
Schematics:

- The project modules are connected according to the following scheme:

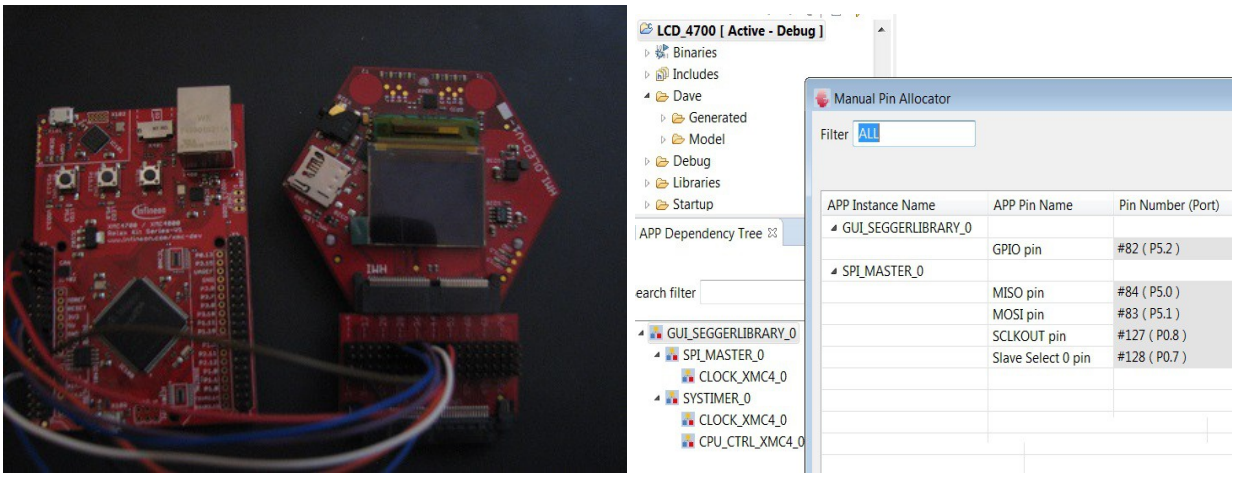


Components Connection

- The mode of operation for MQ135 sensor is described in the following diagram:



- Interconnection of HMI Board with XMC4700 and mapping pins is done according to the next pictures:



Initially, we connected all the hardware used with te development board and made the software settings to use the functions of the Segger emWin5 library. We have upgraded the GUI.h paths in prj-properties according to the GUI_Zagger_lib help specification and set the speed to 1000 and SPI mode with uplinked synchronization.

Since hardware resource do not allow the use of a real lambda prob, we will simulate the operation of one using a MQ135 gas sensor.

The sensor is connected to VCC, GND and an AnalogOut pin of the development board. The vlue of gas in the atmosphere read by the sensor is calculated according to a formula deduced from the sensor datasheet and displayed on the graphical display.

If reading gas value exceeds 1850 units, will be displayed the message: "High mixture!" with the red color and the value of gas from the atmosphere. This means that there is too little oxygen in the exhaust gases.

If reading gas value is below 1200 units, will be displayed the message: "Low mixture!" with the red color and the value of gas from the atmosphere. This means that there is too much oxygen in the exhaust gases.

If reading gas value is between 1200 and 1850 units, will be displayed the message: "Normal mixture!" with the green color and the value of gas from the atmosphere. This means that

the fuel-oxygen raport is on ideal value.

If the mixture is too high or to low, beside the message, a LED will blink to signal this.

5. Results presentation:

The results obtained are in accordance with the system design logic.

6. Source code:

```
#include <DAVE.h>
volatile float gasSense;
uint32_t timer_temp, timer_gas;
uint32_t sensorToRead = 0;
bool isOn=false;
uint32_t senz;
int gaz;

void delay()
{
int y;
        for (y=0;y<0x0fffff;y++);
}
int Adc_Measurement_Handler()
{
   static uint32_t result_gas;
      result_gas =
ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Channel_A_handle);
      gasSense = (result_gas / 4096.0) * 3300;
      return gasSense;
}
void getGas(void *args)
{
        if(isOn==true){
        if ((ADC_MEASUREMENT_t *)args == &ADC_SENSORS)
                ADC_MEASUREMENT_StartConversion(&ADC_SENSORS);
        }
}
int main(void)
{
 /* Dave Apps Initialization */
 DAVE_Init();
 /* Initialize the GUI */
 GUI_Init();
 while(1)
      {
   DIGITAL_IO_SetOutputLow(&led1);
   timer_gas = SYSTIMER_CreateTimer(3000000, SYSTIMER_MODE_PERIODIC, getGas,
&ADC_SENSORS);
   SYSTIMER_StartTimer(timer_gas);
   gasSense= Adc_Measurement_Handler();
                GUI_Clear();
                GUI_SetFont(&GUI_Font16B_1); // bfont
```

```
                isOn=true;
                if(isOn==true)
                {
                        if(gasSense>1850)
                        {
                                //gas detected
                                GUI_SetColor(GUI_RED);
                                DIGITAL_IO_ToggleOutput(&led1);
                                GUI_DispStringAt("Raport:", 63, 18);
                                GUI_DispStringAt("Amestec bogat!", 18, 36);
                                GUI_DispStringAt("Cantitatea este: ", 18, 54);
                                GUI_DispDec(gasSense,4);
                                GUI_DrawCircle  (70, 100, 15);
                                GUI_DrawLine(63,110,77,90);
                                GUI_DrawLine(77,110,63,90);
                        }
                        else if(gasSense<1200)
                        {
                                GUI_SetColor(GUI_RED);
                                DIGITAL_IO_ToggleOutput(&led1);
                                GUI_DispStringAt("Raport:", 63, 18);
                                GUI_DispStringAt("Amestec scazut!", 18, 36);
                                GUI_DispStringAt("Cantitatea este: ", 18, 54);
                                GUI_DispDec(gasSense,4);
                                GUI_DrawCircle  (70, 100, 15);
                                GUI_DrawLine(63,110,77,90);
                                GUI_DrawLine(77,110,63,90);
                        }
                        else
                        {
                                GUI_SetColor(GUI_GREEN);
                                DIGITAL_IO_SetOutputLow(&led1);
                                GUI_DispStringAt("Raport:", 63, 18);
                                GUI_DispStringAt("Amestec normal!", 18, 36);
                                GUI_DispStringAt("Cantitatea este: ", 18, 54);
                                GUI_DispDec(gasSense,4);
                                GUI_DrawCircle  (70, 100, 15);
                                GUI_DrawLine(70,110,63,100);
                                GUI_DrawLine(70,110,77,95);
                        }
                }
                else
                {
                }
    }
}
```

6. Bibliography.

[1]http://www.pilotauto.ro/tuning-magazin/sonda-lambda-10-intrebari-si-raspunsuri-esentiale/

[2] http://www.embedac.ro/CI/Lab/L8/Laborator8.htm

[3]https://www.infineon.com/dgdl/Infineon-Board_User_Manual_XMC4700_XMC4800_Relax_Kit_Series-UM-v01_02-EN.pdf?

fileId=5546d46250cc1fdf01513f8e052d07fc
[4]https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf
[5]https://github.com/SnipeHunting/MQ135-Air-Quality-with-ST7735-Display/blob/SnipeHunting-patch-1/MQ135-master/MQ135.h