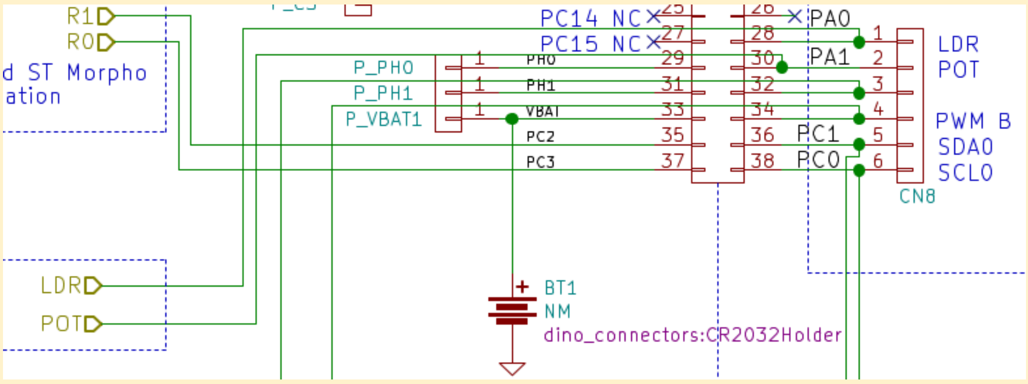
Mark	**1/11**

Team name:	*B5*		
Homework number:	*HOMEWORK 06*		
Due date:	29/10/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

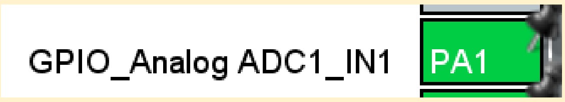
Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1: 3a - ADC acquire 3 voltages via DMA (potentiometer, temperature sensor, Vref) and send to pc via UART DMA

- Setup the PIN of the potentiometer
  - POT is connected to the PA1 pin:

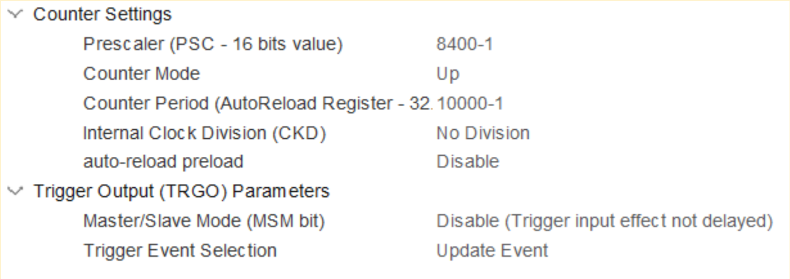


- Setup the pin PA1 to be connected with Input 1 of the ADC:



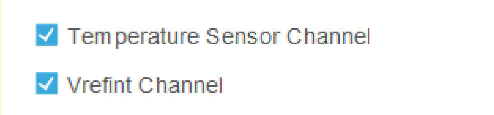
GPIO_Analog ADC1_IN1 PA1

- Setup the timer TIM2:
  - it operates at 1hz, and generate a TRGO event:



Counter Settings	
Prescaler (PSC - 16 bits value)	8400-1
Counter Mode	Up
Counter Period (AutoReload Register - 32	10000-1
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Update Event

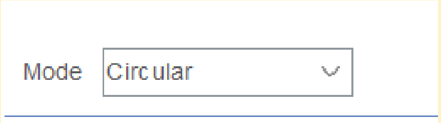
- Setup the ADC:
  - enable the Input 1, the temperature sensor, and the Vref



☑ Temperature Sensor Channel

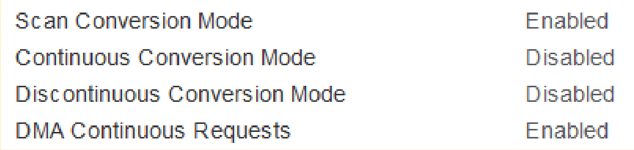
☑ Vrefint Channel

  - Enable the DMA in circular mode:



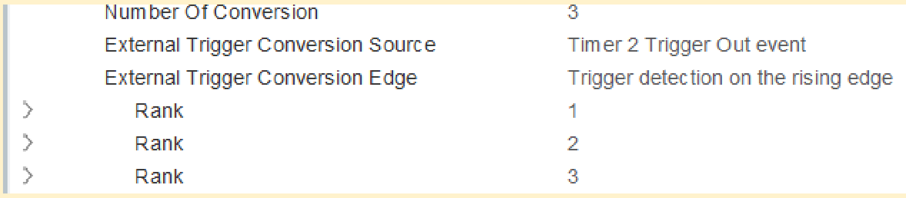
Mode Circular

  - it operates in scan conversion mode, with DMA continuous request



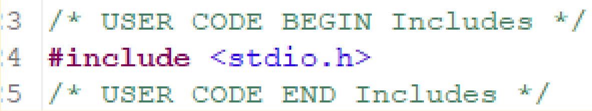
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled

  - the ADC acquisition is triggered by the timer 2.
    The number of conversions is set to 3 because it needs to convert the 3 inputs. Every Rank is set to 480 cycles in order to be sure that the analog value is converted correctly.



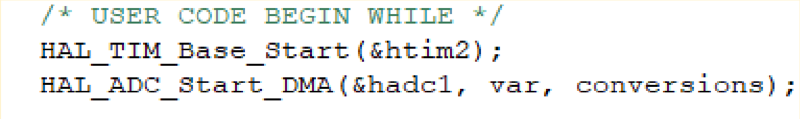
Number Of Conversion	3
External Trigger Conversion Source	Timer 2 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Rank	2
Rank	3

  - For rank 1 leave the channel 1, for rank 2 select the temperature, for rank 3 select Vref
  - Enable the ADC1 global interrupt
- Setup the USART
  - Enable the DMA and set USART2_TX, in order to send the data to the terminal
  - Enable the USART2 global interrupt
- How we implemented the code:
  - include the library in order to use the snprintf later



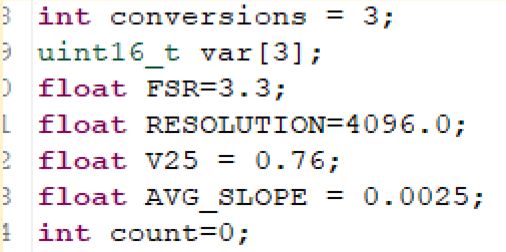
```
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */
```

- in the main, call the starting function of timer2 and of ADC in DMA mode



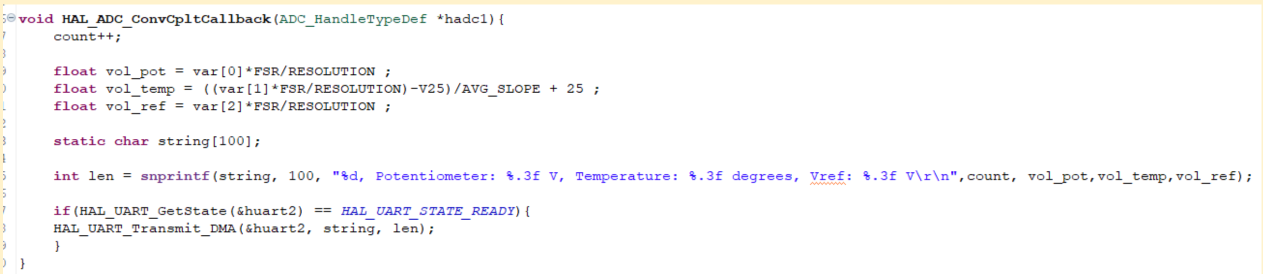
```
/* USER CODE BEGIN WHILE */
HAL_TIM_Base_Start(&htim2);
HAL_ADC_Start_DMA(&hadc1, var, conversions);
```

- declare the variable useful for the computation, as seen in the slides formulas:



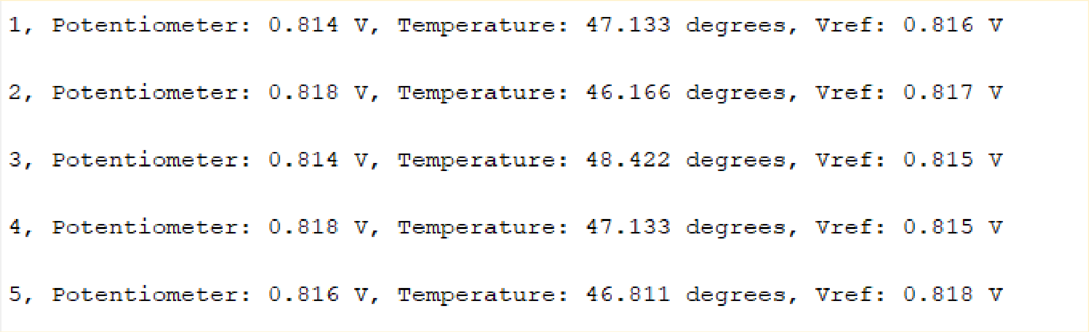
```
int conversions = 3;
uint16_t var[3];
float FSR=3.3;
float RESOLUTION=4096.0;
float V25 = 0.76;
float AVG_SLOPE = 0.0025;
int count=0;
```

- redefine the ADC conversion complete callback:
    - it computes the voltages of the potentiometer, temperature sensor and Vref
    - it uses the variable count to keep track of the messages
    - then, if the state of the UART is READY, it send the string



```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc1){
    count++;

    float vol_pot = var[0]*FSR/RESOLUTION ;
    float vol_temp = ((var[1]*FSR/RESOLUTION)-V25)/AVG_SLOPE + 25 ;
    float vol_ref = var[2]*FSR/RESOLUTION ;

    static char string[100];

    int len = snprintf(string, 100, "%d, Potentiometer: %.3f V, Temperature: %.3f degrees, Vref: %.3f V\r\n",count, vol_pot,vol_temp,vol_ref);

    if(HAL_UART_GetState(&huart2) == HAL_UART_STATE_READY){
    HAL_UART_Transmit_DMA(&huart2, string, len);
    }
}
```

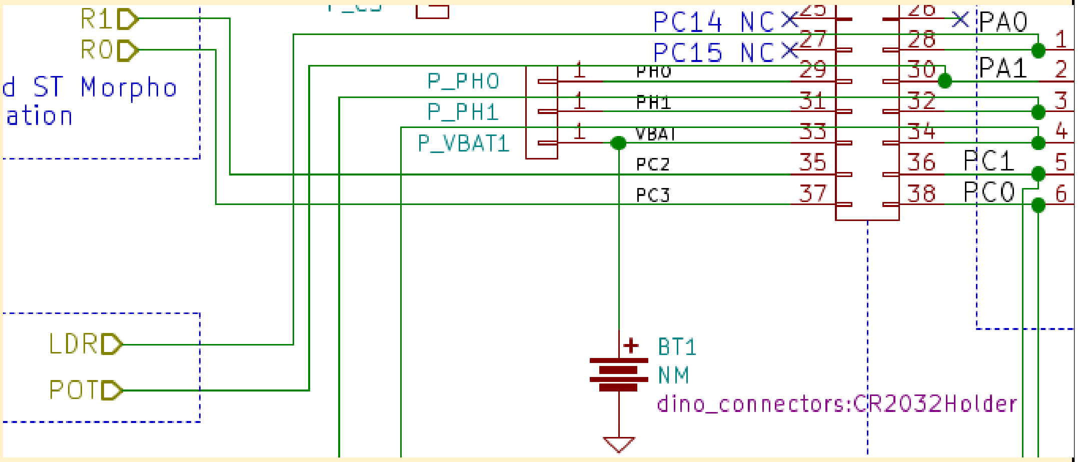
- The output of this project in MATLAB:



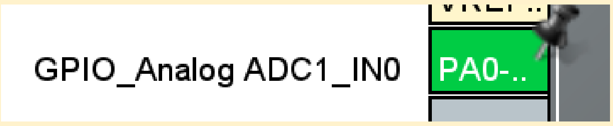
```
1, Potentiometer: 0.814 V, Temperature: 47.133 degrees, Vref: 0.816 V

2, Potentiometer: 0.818 V, Temperature: 46.166 degrees, Vref: 0.817 V

3, Potentiometer: 0.814 V, Temperature: 48.422 degrees, Vref: 0.815 V

4, Potentiometer: 0.818 V, Temperature: 47.133 degrees, Vref: 0.815 V

5, Potentiometer: 0.816 V, Temperature: 46.811 degrees, Vref: 0.818 V
```

Part 2: 3b - acquire the LDR (light dependent resistor), with ADC half conversion complete callback

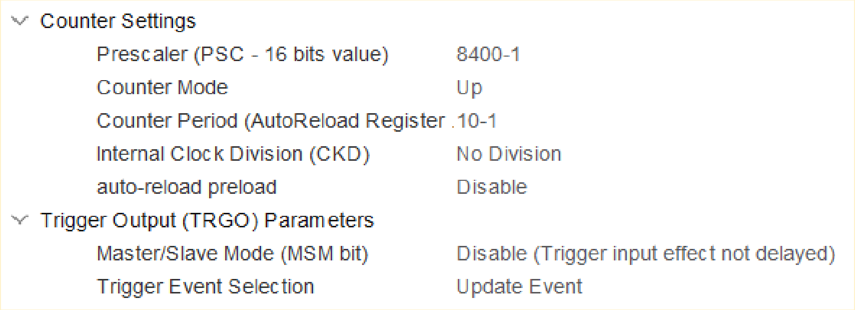
- Setup the PIN
  - LDR is connected to pin PA0



  - Setup the pin PA0 to be connected with Input 0 of the ADC:

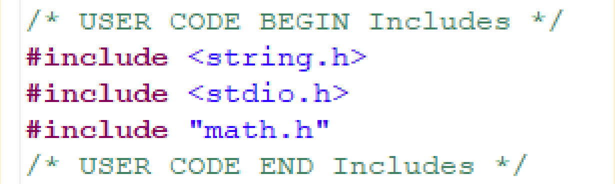

GPIO_Analog ADC1_IN0 PA0-..

- Setup the timer TIM2:
  - Since it is asked to acquire the value every millisecond, the timer operates at 1000hz, and generate a TRGO event:



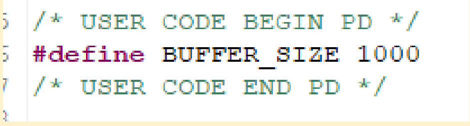
Counter Settings	
Prescaler (PSC - 16 bits value)	8400-1
Counter Mode	Up
Counter Period (AutoReload Register .	10-1
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Update Event

- Setup the ADC like before except disabling the "Scan Conversion Mode”, except this time we only have Input 0 (IN0)
- Setup the USART like before
- How we implemented the code:
  - include the library in order to use the snprintf, the memset, and the pow functions later



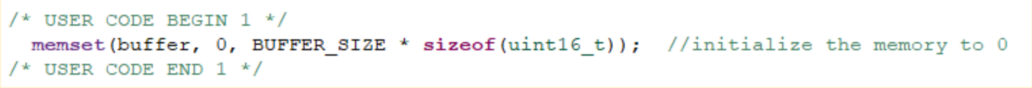
```
/* USER CODE BEGIN Includes */
#include <string.h>
#include <stdio.h>
#include "math.h"
/* USER CODE END Includes */
```

  - define the dimension of the buffer: 1000 since it needs to store 1000 values



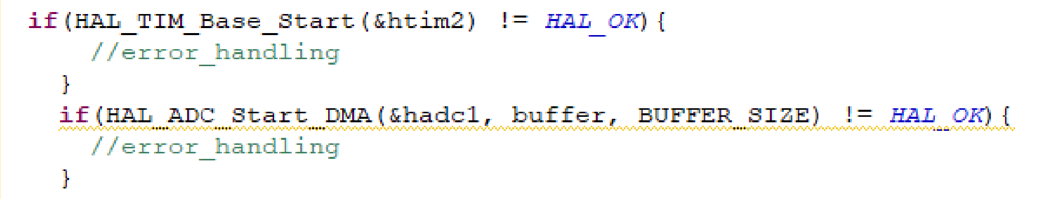
```
/* USER CODE BEGIN PD */
#define BUFFER_SIZE 1000
/* USER CODE END PD */
```

  - in the main, the buffer is initialized to 0

- 

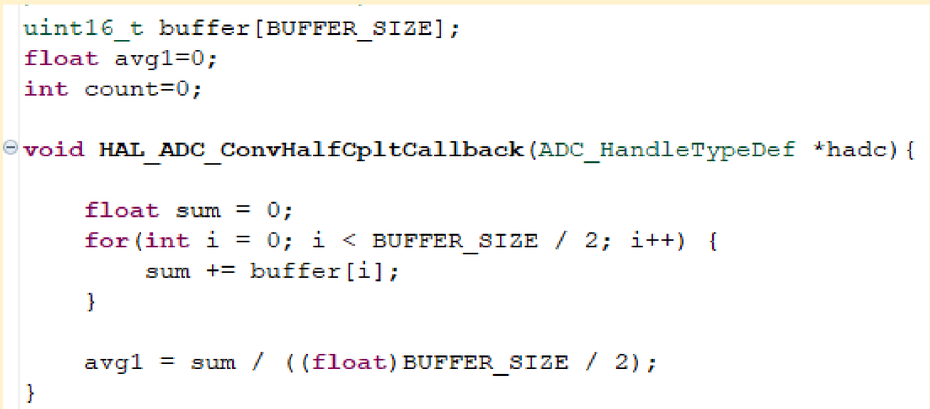
```
/* USER CODE BEGIN 1 */
  memset(buffer, 0, BUFFER_SIZE * sizeof(uint16_t));  //initialize the memory to 0
/* USER CODE END 1 */
```

- in the main, call the starting function of timer2 and of ADC in DMA mode

- 

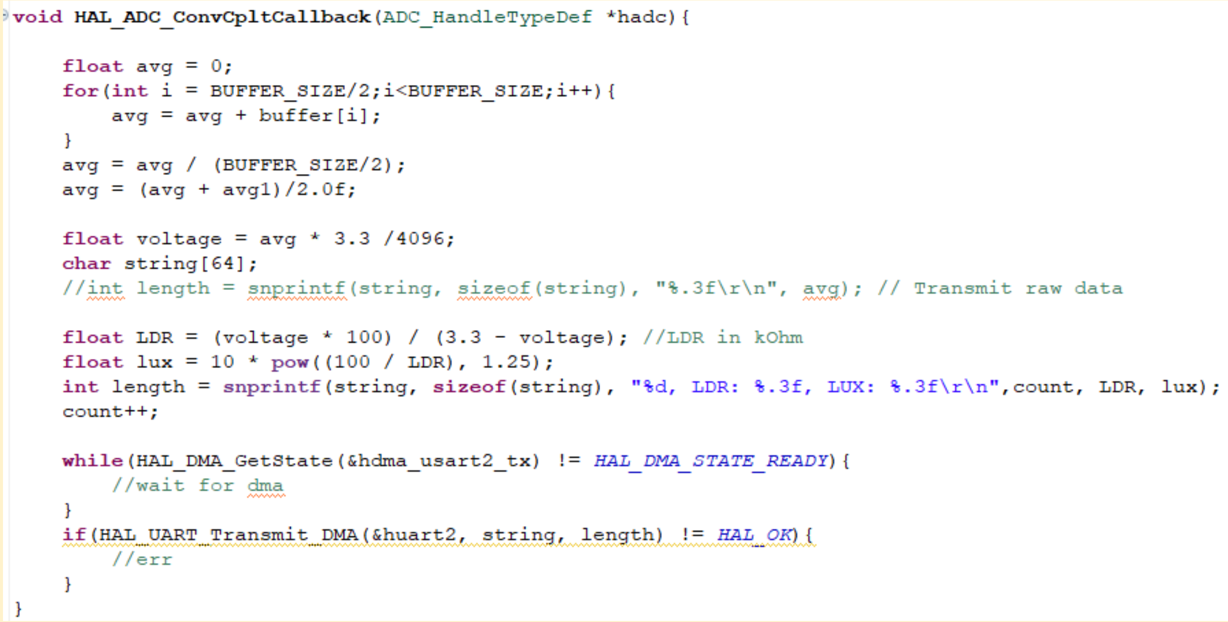
```
 if(HAL_TIM_Base_Start(&htim2) != HAL_OK){
     //error_handling
   }
   if(HAL_ADC_Start_DMA(&hadc1, buffer, BUFFER_SIZE) != HAL_OK){
     //error_handling
   }
```

- We use the Half conversion callback in order to get the first 500 values and compute the average. This is useful because otherwise, if we only used the Conversion complete callback, the first data of the buffer would be overwritten by new data before they could be used to compute the average value.
    - the first 500 values are added in the sum variable and then the first average avg1 is computed

- 

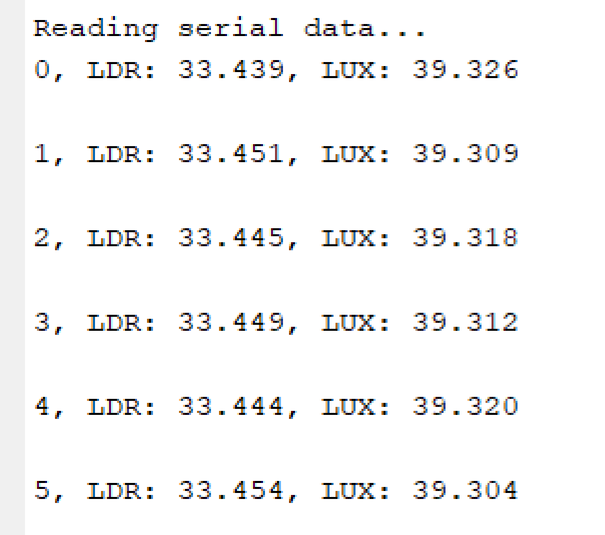
```
uint16_t buffer[BUFFER_SIZE];
float avg1=0;
int count=0;

void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc){

    float sum = 0;
    for(int i = 0; i < BUFFER_SIZE / 2; i++) {
        sum += buffer[i];
    }

    avg1 = sum / ((float)BUFFER_SIZE / 2);
}
```

- The conversion complete callback is used to compute the average value of the last 500 values. Then the overall average is computed using the 2 average values.
    - The LDR and Lux values are computed with the formulas shown in the slides.
    - The variable counter is used to keep track of the number of data sent.
    - Then if the state of the USART is READY, the string is transmitted



```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){

    float avg = 0;
    for(int i = BUFFER_SIZE/2;i<BUFFER_SIZE;i++){
        avg = avg + buffer[i];
    }
    avg = avg / (BUFFER_SIZE/2);
    avg = (avg + avg1)/2.0f;

    float voltage = avg * 3.3 /4096;
    char string[64];
    //int length = snprintf(string, sizeof(string), "%.3f\r\n", avg); // Transmit raw data

    float LDR = (voltage * 100) / (3.3 - voltage); //LDR in kOhm
    float lux = 10 * pow((100 / LDR), 1.25);
    int length = snprintf(string, sizeof(string), "%d, LDR: %.3f, LUX: %.3f\r\n",count, LDR, lux);
    count++;

    while(HAL_DMA_GetState(&hdma_usart2_tx) != HAL_DMA_STATE_READY){
        //wait for dma
    }
    if(HAL_UART_Transmit_DMA(&huart2, string, length) != HAL_OK){
        //err
    }
}
```

- The output of this project in MATLAB:



```
Reading serial data...
0, LDR: 33.439, LUX: 39.326

1, LDR: 33.451, LUX: 39.309

2, LDR: 33.445, LUX: 39.318

3, LDR: 33.449, LUX: 39.312

4, LDR: 33.444, LUX: 39.320

5, LDR: 33.454, LUX: 39.304
```

Professor comments:

Project 1: Very strange values of temperature (very hot room!!!), and Vref should be 1.2V. I don't see from the report how you configured the 3 ranks, but maybe you are always sampling the potentiometer...