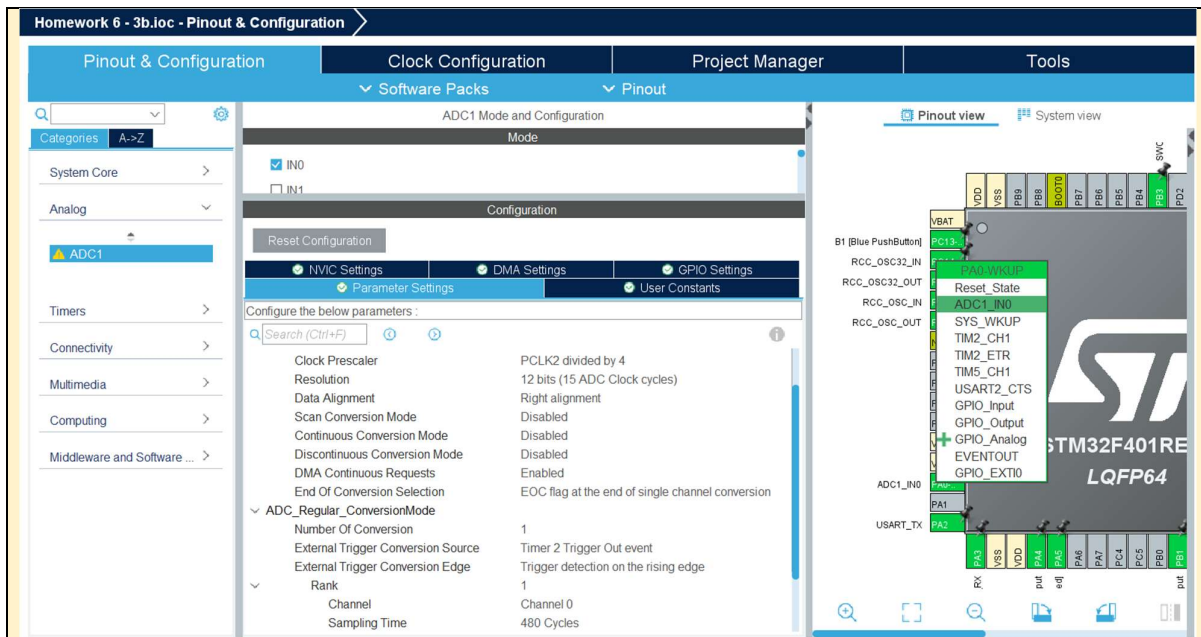


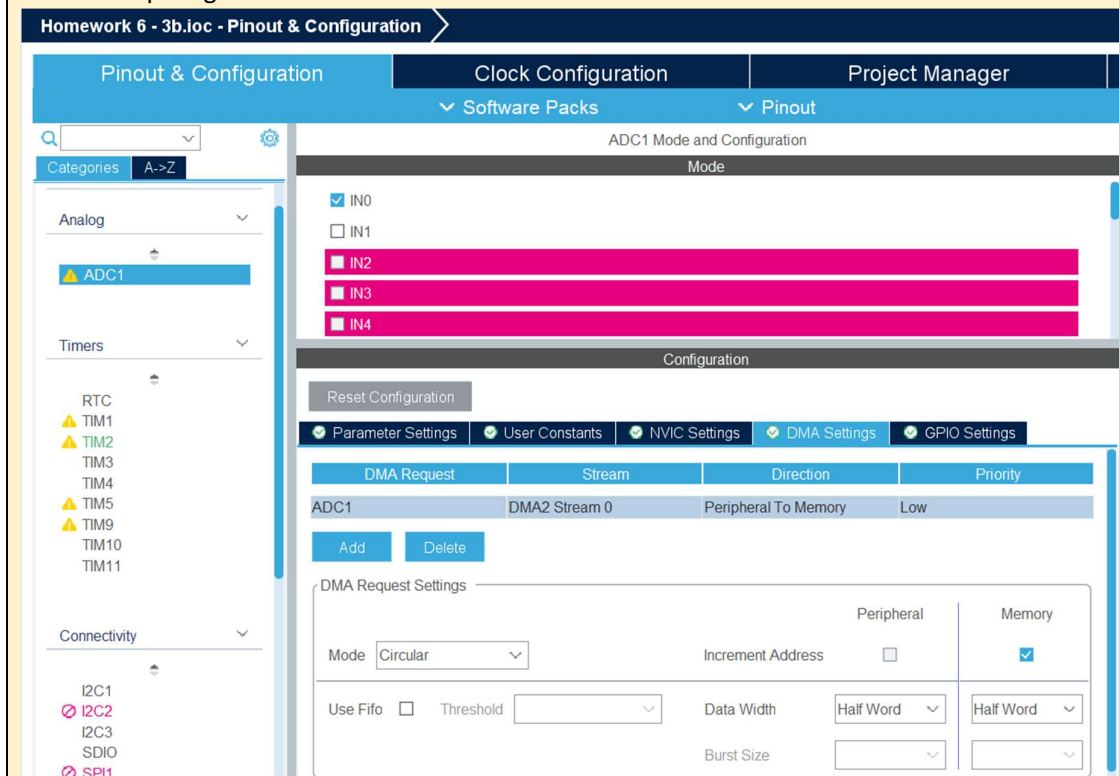
Mark	/11
------	-----

Team name:	A14		
Homework number:	HOMEWORK 06		
Due date:	26/10/25		
Contribution	NO	Partial	Full
Mattia Di Mauro			x
Francesca Biondi			x
Lorenzo Castelli			x
Carmen Maria Niro			x
Pietro Albrigi			x
Notes: none			

Project name	HOMEWORK 6		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x
<b>Project 2b:</b> We identified the LDR pin (PA0) in the Hands-on Lab schematics and configured it in the IOC file as ADC1_IN0. We enabled the DMA Continuous Request and selected Timer 2 Trigger Output Event as the external trigger conversion source, with trigger detection set on the rising edge. The sampling time was set to 480 cycles, ensuring stable and accurate readings from the LDR:			



In the DMA configuration, ADC1 was added with mode set to circular, allowing continuous data transfers without requiring CPU intervention:



For Timer 2, we selected the internal clock source and configured a prescaler of 8399, an auto-reload register (ARR) value of 9 and Update Event as Trigger Event Selection. This means that Timer 2 generates an update event, and thus a trigger for the ADC, every millisecond. Since the ADC is configured to start a conversion on each rising edge of the trigger, this setup results in one ADC conversion being started every millisecond, precisely synchronized with the timer overflow.

Homework 6 - 3b.ioc - Pinout & Configuration

Pinout & Configuration

Clock Configuration

Project Manager

Software Packs

Pinout

Categories

A-Z

Timers

RTC

TIM1

TIM2

TIM3

TIM4

TIM5

TIM9

TIM10

TIM11

Connectivity

I2C1

I2C2

I2C3

SDIO

SPI1

SPI2

SPI3

USART1

USART2

USART6

TIM2 Mode and Configuration

Mode

Slave Mode

Disable

Trigger Source

Disable

Clock Source

Internal Clock

Channel1

Disable

Channel2

Disable

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)

8399

Counter Mode

Up

Counter Period (AutoReload Register - 32 bits value )

9

Internal Clock Division (CKD)

No Division

auto-reload preload

Disable

Trigger Output (TRGO) Parameters

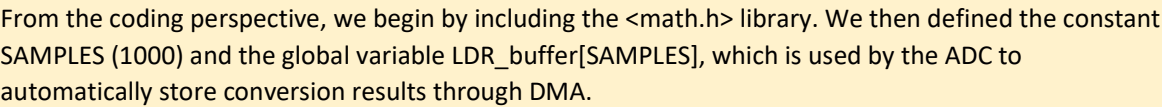
Master/Slave Mode (MSM bit)

Disable (Trigger input effect not delayed)

Trigger Event Selection

Update Event

We also enabled USART communication using DMA at a baud rate of 9600:



In the main() function, we start Timer 2 using HAL\_TIM\_Base\_Start(&htim2) and initialize the ADC in DMA mode with the call HAL\_ADC\_Start\_DMA(&hadc1, &LDRBuffer, SAMPLES). This ensures that one sample is transferred to LDR\_buffer every millisecond without any CPU overhead.

```
132  /* USER CODE BEGIN 2 */
133  HAL_TIM_Base_Start(&htim2);
134  HAL_ADC_Start_DMA(&hadc1, &LDR_buffer, SAMPLES);
135  /* USER CODE END 2 */
```

Outside the main() function, two callback functions are defined: HAL\_ADC\_ConvHalfCpltCallback() and HAL\_ADC\_ConvCpltCallback(). These functions are automatically invoked by the HAL library when the ADC, operating in DMA mode, completes half or all of the buffer conversions, respectively. The half-complete callback is triggered when half of the ADC buffer is filled, enabling intermediate data processing while maintaining continuous sampling through DMA.

In both callbacks, the ADC samples stored in the LDR\_buffer array are first converted into a corresponding voltage based on the ADC's 12-bit resolution and the 3.3 V reference.

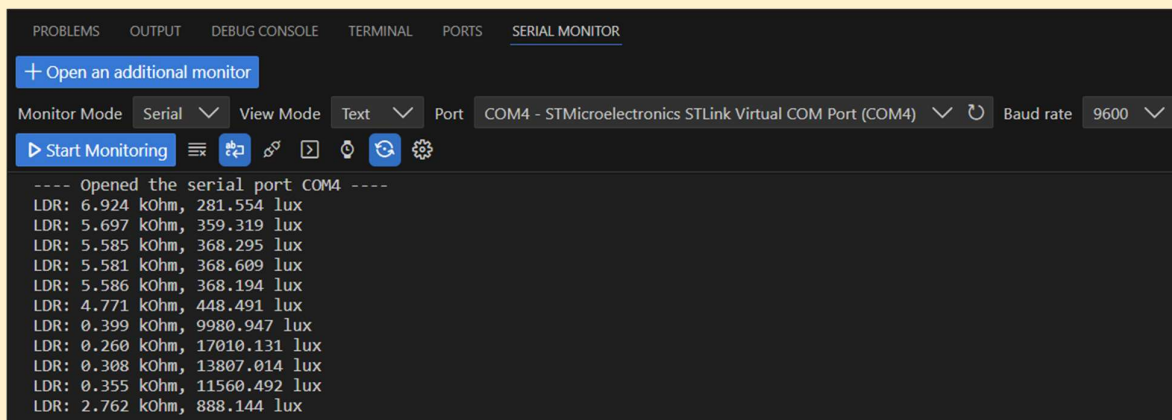
Using this voltage, the resistance of the LDR is calculated. The computed resistance values are then accumulated in the variable average, while the counter count is incremented each time a new value is acquired.

```
68  /* Private user code -----*/
69  /* USER CODE BEGIN 0 */
70  void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc) {
71      for (int i = 0; i < SAMPLES / 2; i++) {
72          float voltage = LDR_buffer[i] * (3.3 / 4095);
73          float R_LDR = (voltage * 100) / (3.3 - voltage);
74          average += R_LDR;
75          count++;
76      }
77  }
78
79  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
80      for (int i = SAMPLES / 2; i < SAMPLES; i++) {
81          float voltage = LDR_buffer[i] * (3.3 / 4095);
82          float R_LDR = (voltage * 100) / (3.3 - voltage);
83          average += R_LDR;
84          count++;
85      }
86
87      if (count >= SAMPLES) {
88          average = average / SAMPLES;
89          float lux = 10 * pow((100/average), 1.25);
90          static char string[50];
91          snprintf(string, sizeof(string), "LDR: %.3f kOhm, %.3f lux \r\n", average, lux);
92          HAL_UART_Transmit_DMA(&huart2, string, sizeof(string));
93          average = 0;
94          count = 0;
95      }
96  }
97  /* USER CODE END 0 */
```

Once the counter reaches 1000, meaning approximately one second has passed (since we are sampling at 1 kHz), we compute the average resistance by dividing the accumulated sum by the number of samples. This average resistance is then used to calculate the corresponding light intensity in lux, using the formulas provided in the lecture slides. Finally, the computed values (such as average resistance and lux) are formatted into a string and transmitted to the serial terminal via USART using DMA.

Before exiting the conditional block, both the average sum and the count are reset to begin the next one-second measurement cycle.

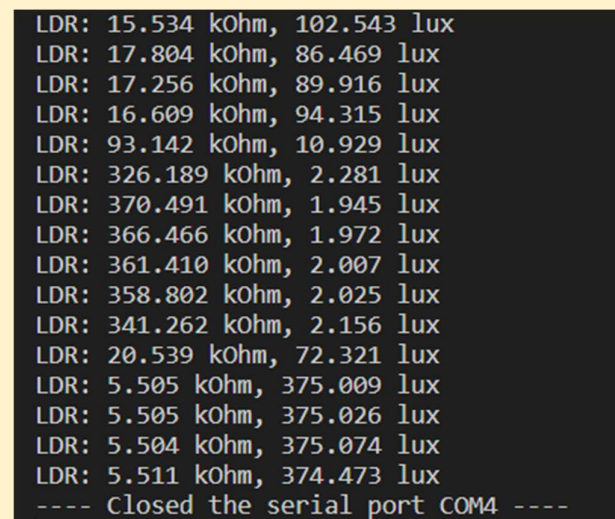
The code was then tested on the board and confirmed to work as expected. From the remote terminal, plausible values were observed under different lighting conditions: in a normally lit room (around 5 kOhm/ 300 lux), when the sensor was illuminated with a flashlight (less than 1 kOhm / over 10 klux):



The screenshot shows the Serial Monitor window with the following settings: Monitor Mode: Serial, View Mode: Text, Port: COM4 - STMicroelectronics STLink Virtual COM Port (COM4), Baud rate: 9600. The output text is as follows:

```
---- Opened the serial port COM4 ----  
LDR: 6.924 kOhm, 281.554 lux  
LDR: 5.697 kOhm, 359.319 lux  
LDR: 5.585 kOhm, 368.295 lux  
LDR: 5.581 kOhm, 368.609 lux  
LDR: 5.586 kOhm, 368.194 lux  
LDR: 4.771 kOhm, 448.491 lux  
LDR: 0.399 kOhm, 9980.947 lux  
LDR: 0.260 kOhm, 17010.131 lux  
LDR: 0.308 kOhm, 13807.014 lux  
LDR: 0.355 kOhm, 11560.492 lux  
LDR: 2.762 kOhm, 888.144 lux
```

and when the sensor was covered to simulate darkness (around 350 kOhm / only a few lux):



The screenshot shows the Serial Monitor window displaying the following output text:

```
LDR: 15.534 kOhm, 102.543 lux  
LDR: 17.804 kOhm, 86.469 lux  
LDR: 17.256 kOhm, 89.916 lux  
LDR: 16.609 kOhm, 94.315 lux  
LDR: 93.142 kOhm, 10.929 lux  
LDR: 326.189 kOhm, 2.281 lux  
LDR: 370.491 kOhm, 1.945 lux  
LDR: 366.466 kOhm, 1.972 lux  
LDR: 361.410 kOhm, 2.007 lux  
LDR: 358.802 kOhm, 2.025 lux  
LDR: 341.262 kOhm, 2.156 lux  
LDR: 20.539 kOhm, 72.321 lux  
LDR: 5.505 kOhm, 375.009 lux  
LDR: 5.505 kOhm, 375.026 lux  
LDR: 5.504 kOhm, 375.074 lux  
LDR: 5.511 kOhm, 374.473 lux  
---- Closed the serial port COM4 ----
```

Professor comments: