Mark	A

Team name:	*A1*		
Homework number:	*HOMEWORK 5*		
Due date:	29/10/2022		
Contribution	NO	Partial	Full
Monti Pietro			*x*
Moretto Alessia			*x*
Pallotto Francesco			*x*
Perna Alessandro			*x*
Ventura Ludovico			*x*
Notes:			

Project name	ADC scan using DMA and Light Dependent Resistor		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Explanation:
We successfully completed the homework.

Part 1a:

Firstly, we initiated the configuration of the ADC on the STM32 platform by activating the following channels from the Input/Output Configuration (IOC): IN1 (for potentiometer reading), the Temperature Sensor Channel, and Vrefint Channel. Within the ADC parameter settings, we specified a prescaler value of PCLK divided by 4 and a resolution of 12 bits. Furthermore, we enabled Scan Conversion Mode, established the number of conversions as 3 (corresponding to the number of channels in use), and assigned rank 1 to IN1, rank 2 to the Temperature Sensor Channel, and rank 3 to Vrefint Channel, all with a sampling time of 480 cycles. Additionally, we configured the DMA connection from peripheral to memory in circular mode and enabled DMA continuous requests in the ADC settings.
Since the method with which data was to be sent to the terminal was unspecified, we chose to transmit via DMA. To this aim, we configured the USART in Asynchronous mode with a Baud Rate of 115200 Bits/s and added its DMA in Memory to Peripheral mode in normal operation.
Lastly, we employed Timer TIM2 to generate periodic output events. To achieve a timer frequency of 1Hz, we configured the Prescaler to 8400-1 and the Auto Reload Register to 10000-1. After configuring the graphical user interface, we set USART2 and TIM2 as global interrupts in the NVIC table.
With the timer initiated, we used the function HAL_ADC_Start_DMA to initiate the ADC in DMA mode. In the last parameter of this function, we specified the number of channels to be used, which in this case is 3. Data collection and transmission to the terminal were managed through interrupt callbacks.
We defined an interrupt callback function for the ADC, which stores the values of the three channels in an array of three elements called "to_buffer" after the sampling phase is completed. The first position holds the potentiometer voltage, the second contains the thermometer temperature, and the last stores the Vrefint value. Subsequently, we formatted the message to be transmitted to the terminal and wrote it to "UART_buf".

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
    if (hadc == &hadc1) {
       to_buffer[0] = ADC_results[0]*3.3/4095.0;
       to_buffer[1] = ((ADC_results[1] * 3.3 / 4096.0) - 0.76) / 0.0025 + 25;
       to_buffer[2] = ADC_results[2]*3.3/4095;
       UART_buf_len = snprintf(UART_buf,sizeof(UART_buf),"Vpot: %.3f V   Temperature: %.3f %cC "
                       "  Vref: %.3f V\r\n", to_buffer[0], to_buffer[1], degree, to_buffer[2]);
    }
}
```

In the timer interrupt callback, we transmitted the message to the terminal via UART and then resumed the ADC sampling phase, as done previously.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim==&htim2){
        HAL_UART_Transmit_DMA(&huart2, (uint8_t *) UART_buf, UART_buf_len);
        HAL_ADC_Start_DMA(&hadc1, (uint32_t *) ADC_results, num_channels);
    }
}
```

Part 1b:

We initiated the configuration of the ADC on the STM32 platform by enabling channel IN0 from the IOC, as it is the one connected to the Light Dependent Resistor. In the ADC parameter settings, we specified a prescaler value of PCLK divided by 4 and a resolution of 12 bits. We then selected the external Trigger Conversion Source as "Timer 2 Trigger Out Event" to enable TIM2 to initiate a hardware based sampling phase every 1ms. Additionally, we configured the DMA connection from peripheral to memory in circular mode and enabled continuous DMA requests in the ADC settings.
Subsequently, we configured the Universal Synchronous Asynchronous Receiver Transmitter (USART) in Asynchronous mode with a Baud Rate of 115200 Bits/s.
For Timer TIM2, set as the internal clock, we established a frequency of 1kHz by configuring the Prescaler to 8400-1 and the Auto-Reload Register to 10-1, and we selected the Update Event Option as the Trigger Event Selection. We also activated Timer TIM3 as the internal clock, configuring the Prescaler to 8400-1 and the Auto-Reload Register to 10000-1, resulting in a clock frequency of 1Hz. Finally, we enabled it as a global interrupt in the NVIC table.
Following the initialization in interrupt mode for TIM2 and TIM3, and in DMA mode for the ADC, as part of this homework assignment, we managed the input and output data stream through interrupt callbacks. We defined an interrupt callback function for the ADC, which stores the voltage of the ADC in the "v_adc" variable. We then converted this voltage to the corresponding resistance value and added it to the "resistance_sum" variable. This variable is used to calculate the average resistance value. Additionally, we incremented the "resistance_counter" to keep track of the number of samples acquired, ensuring accuracy despite potential software delays or different clocks inconsistencies.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc) {
        v_adc = adc_result[0] * 3.3 / 4095.0;
        resistance_sum = resistance_sum + (v_adc * 100000) / (3.3 - v_adc);
        resistance_counter++;
    }
```

In the TIM3 callback, which executes once per second, we calculated the average resistance value from the nearly 1000 samples acquired and stored in the "avg" variable. We then formatted a message for transmission to the terminal, which included the average resistance value and the corresponding lux level calculated according to the formula provided in the homework assignment. Afterward, we reset the

"resistance_counter" and "resistance_sum" variables in preparation for a new sampling sequence of approximately 1 second. Finally, we transmitted the message to the terminal via UART.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim == &htim3) {
        avg=(resistance_sum/resistance_counter);
        length = snprintf(text, sizeof(text),"Resistance = %.3f ohm; Light = %.3f lux \r\n",
                          avg, 10 * pow(100000 / (avg), 1.25));
        resistance_counter = 0;
        resistance_sum = 0;
        HAL_UART_Transmit(&huart2, (uint8_t *)text, length, 20);
        }
    }
```

Professor comments:
PART A: It is not clear if TIM2 also triggers the ADC or if the ADC is continuously converting at its maximum speed and you use the TIM only to send data every 1 s. Obviously this second option is highly inefficient.

PART B: You should use the DMA to directly save all the 1000 values to be averaged. Gabriele will explain exactly how to do it during the class