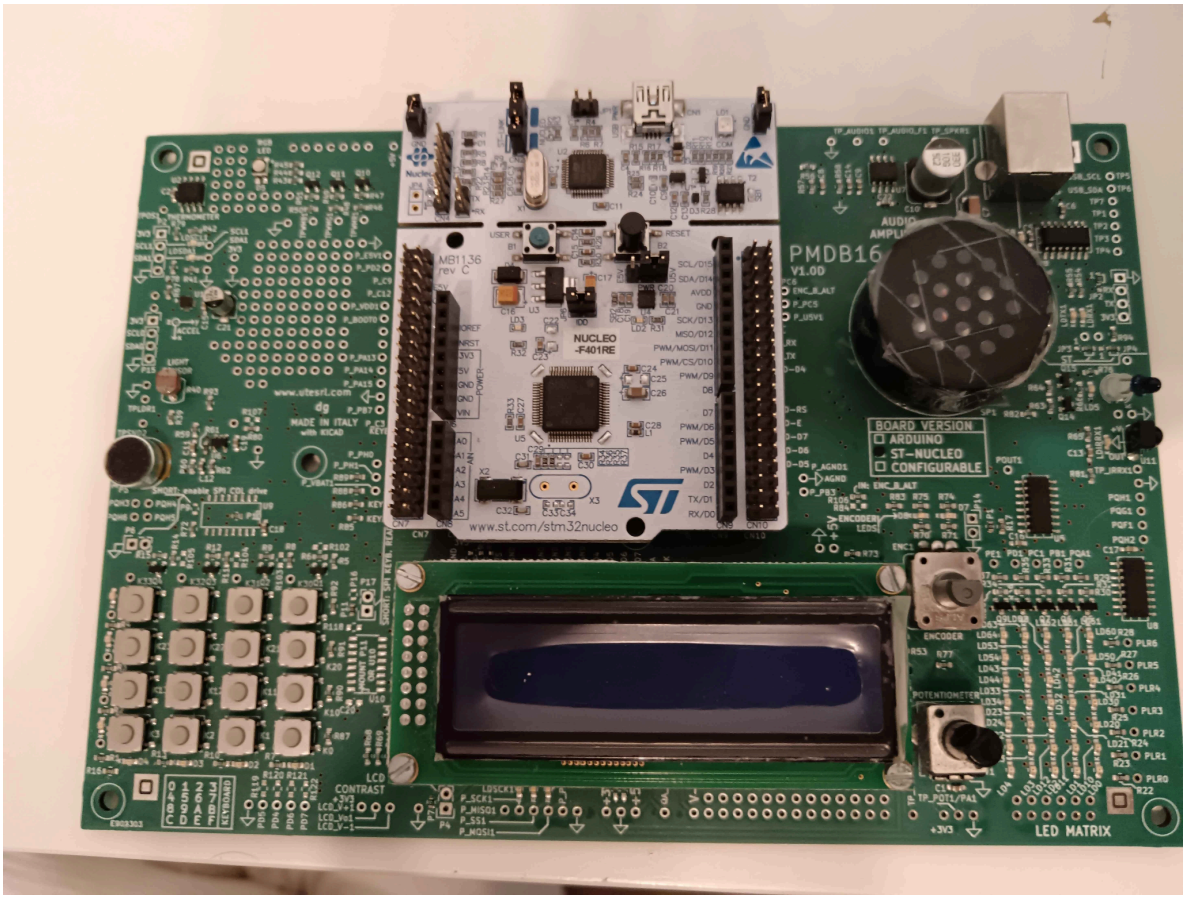
Sensor system

Homework Report



HOMEWORK NUMBER 2

2024 - 2025

POLITECNICO
MILANO 1863

Summary

I. Members work

Members	Exercise 1	Exercise 2	General
Cazin Némo	✓	✓	Written report and did the videos
Adrien Paliferro	✓	✓	Edited the report
Heikki Leveelahti	✓	✓	
Osmo Kieksi	✓	✓	
Constantijn Coppers	✓	✓	Reviewed and edited the report

In general :

We worked on Exercise 2 on Tuesday, and Exercise 1 on Wednesday. We gave the others time to do the exercises on their own.

Link to the github : *https://github.com/nemocazin/sensor_system_labs*

II. Exercise 1

Research through the datasheet and technical schematics of the NUCLEO board and the STM32 microcontroller revealed the following pins:

- Green LED (LD2) on D13: **PA5**
- Microphone (SND_IN): **PA8**

In the STM32CubeIDE, we connected the microphone pin (**PA8**) to the external interrupt line **GPIO_EXTI** (`.ioc` file). Then, **EXTI line[9:5] interrupts** was enabled in the NVIC tab. Finally, in the GPIO tab, we change the GPIO mode of **PA8** to **External Interrupt Mode with Rising edge trigger detection**.

After saving these changes, we could find the function **EXTI9_5_IRQHandler** in the `stm32f4xx_it.c` file. This proved that the pin was correctly configured. The callback function `HAL_GPIO_EXTI_Callback` was composed in the `main.c` file.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  switch(GPIO_Pin)
  {
    case GPIO_PIN_8: // if pin == 8
      static uint32_t lastInterruptTime = 0;
      uint32_t currentTime = HAL_GetTick();

      // only write to LED when time between interrupts is > 200 ms
      // to cancel the noise
      if((currentTime - lastInterruptTime) > 200)
      {
        HAL_GPIO_WritePin(GPIOA, GREEN_LED, led_state); // write to LED
        led_state = !led_state;      // invert state of LED
        lastInterruptTime = currentTime; // get last intrr time
        break;
      }

    default:  // pin != 8…
      break;  // …do nothing
  }
}
```

We notice that the LED turned OFF/ON on the sound but if it's too noisy, the LED can't stop turning ON and OFF
So, we decided to add a timer to cancel the noise.
Now, we get the actual time and we do a check every 200ms if there is one sound that happens.
Link to the demonstration : https://youtu.be/4WrYZGMUwQw

III. Exercise 2

Using the STM32CubeIDE we configured timer TIM1 as a PWM generator on Channel 1.
The desired frequency can be set by changing the period and prescaler, and is given by the formula

$$F_{PWM} = \frac{F_{CLK}}{(1 + period) * (1 + prescaler)}$$

Where:

- F_{PWM} = frequency of the PWM signal (1Hz)
- F_{CLK} = Clock Frequency (84 MHz)
- period = Counter of the timer
- prescaler = Prescaler of the timer

Using the formula above, $F_{PWM} = 1$ is obtained by making the denominator equal to the numerator. However, we must keep in mind that the maximum available value for the prescaler and period is 65535 (16 bit). Therefore the prescaler is set to 8399 and the period to 9999. Furthermore, a duty cycle of 50% is obtained by configuring the pulse at 4999.

Now we must make a function to know when the LED should change its state. For that, we use the condition: "__HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1) > __HAL_TIM_GET_COUNTER(&htim1)".

```
while (1)
  {
          if      (__HAL_TIM_GET_COMPARE(&htim1,      TIM_CHANNEL_1)      >
__HAL_TIM_GET_COUNTER(&htim1))
  {
     HAL_GPIO_WritePin(GPIOA, GREEN_LED, led_state); // Change state of
green led
    led_state = !led_state;
  }
```

It checks whether the current counter value is less than the comparison value.
However, we have tried several types of variables to change the state of the LED (uint8_t, uint16_t, int, GPIO_PinState). In all cases except uint8_t, the LED only lights up partially or not at all. We therefore chose the uint8_t type.

Link to the demonstration : https://youtube.com/shorts/ggwv8n-rMyo

IV. Professors comments

EXERCISE 1 :
EXERCISE 2 :