WRIGHT STATE UNIVERSITY

LAB-2

GPIO: General-Purpose Input/Output

CEG 7360 Embedded Systems

Name : - Keerthi Patil
UID : - U01110229
Email : - patil.115@wright.edu

Name : - Dhamini Vootkuri
UID : - U01111434
Email : - vootkuri.3@wright.edu

Lab 2

GPIO: General-Purpose Input/Output

A. Introduction:

GPIO (General purpose input output) is used to connect a microcontroller to many external devices for indicating whether the signal is 0 or 1. These pins basically acts as inputs or outputs. Inputs reads the signal from sensors or buttons, while outputs can control devices such as LED. ADC converts analog signals to digital values. In Nucleo board internally ADC is present which are used for many applications. ADC pin on the board is used for performing the experiment.

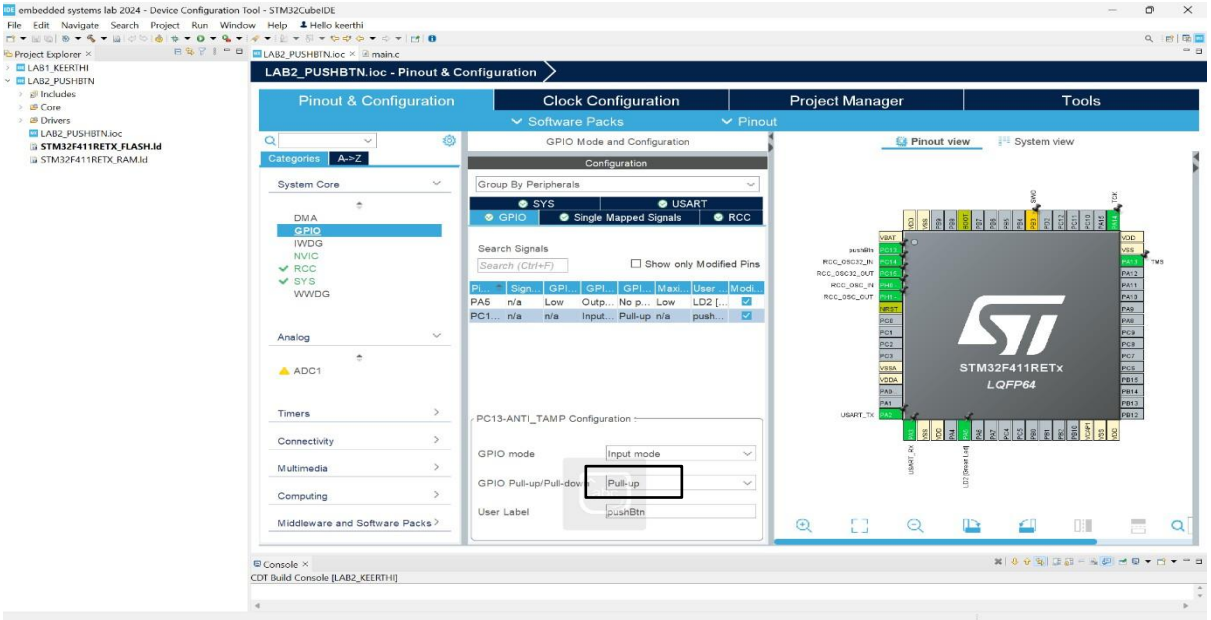
Push buttons are like mechanical switches where these act as binary inputs to the board. To perform the experiment, I have used a push button as GPIO input. LDR is a photoresistor sensor that changes resistance according to the intensity of light. So, if there is bright light the resistance is low and for low light resistance is high. So, in the experiment the board senses the light conditions using LDR and act accordingly based on ADC value readings.

B. Experiment Setup:

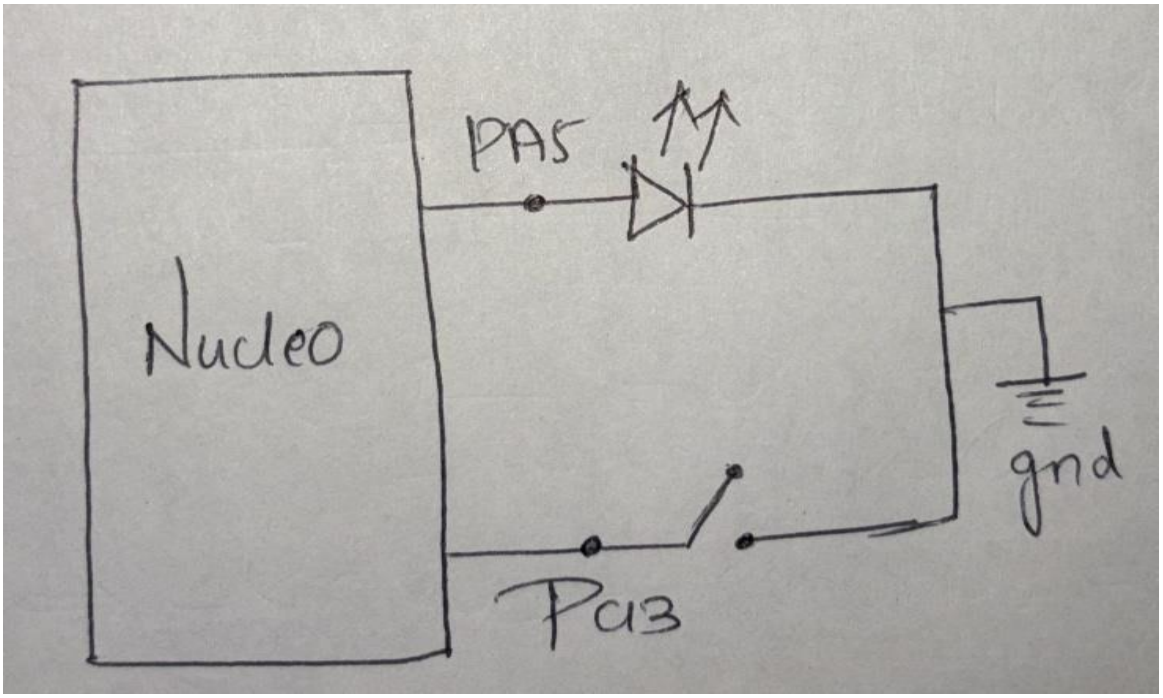
1. LED Blinking using Push button:

Components: To perform this experiment the components required are microcontroller board, USB Cable and software.

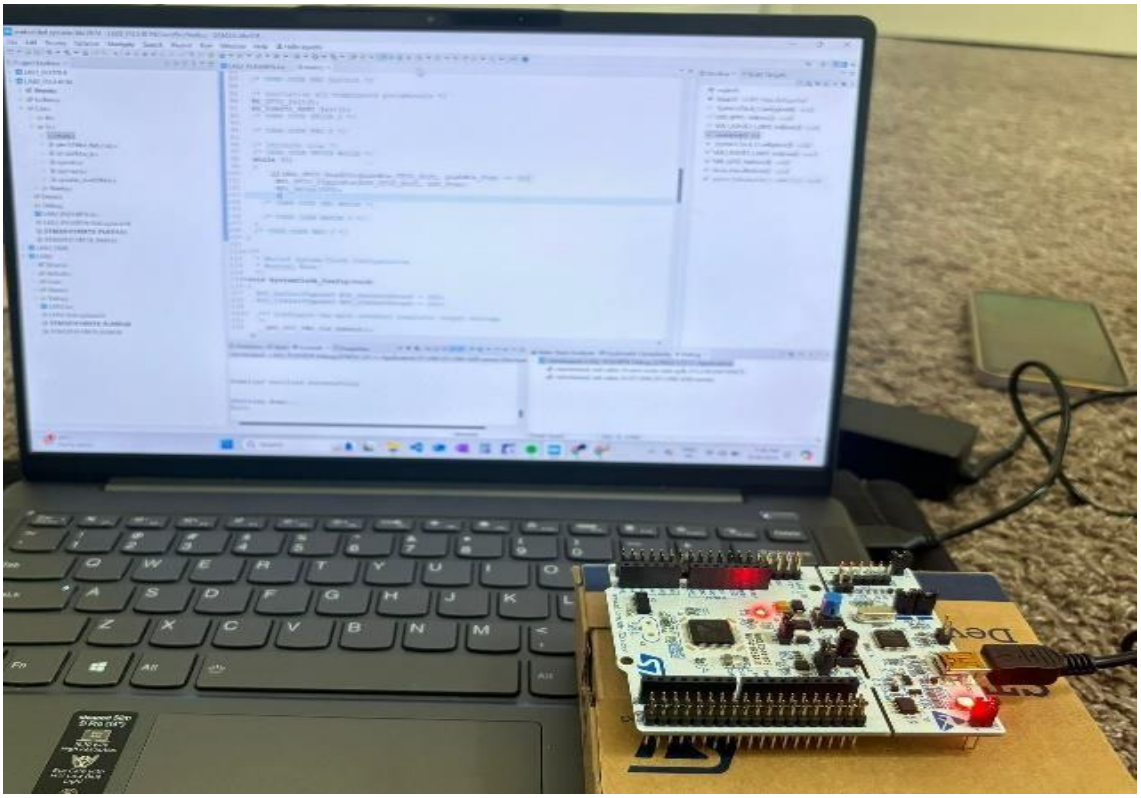
- Here I have set the Pin **PC13** as **GIPO input mode** and under System core I have set the GPIO to be **pull-up** and named it as **pushBtn**.



- The below is the circuit diagram and experimental setup.



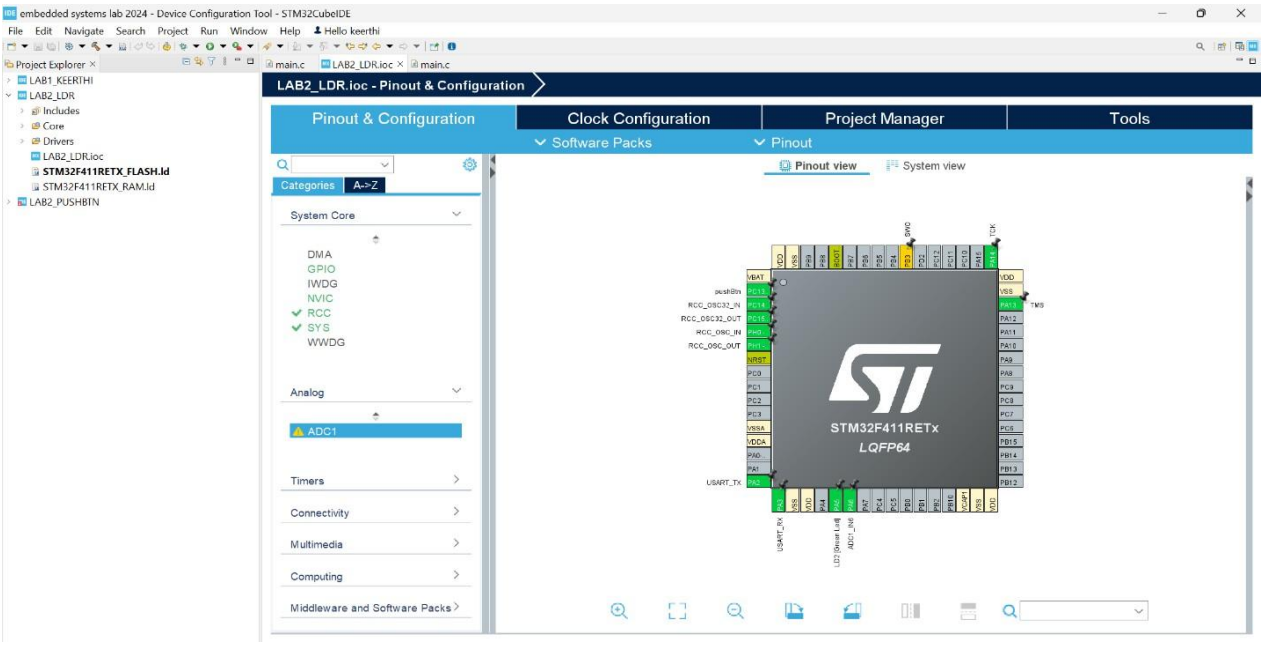
- Here the board is directly connected to the system using a USB cable and the blue switch on the board acts as push button.



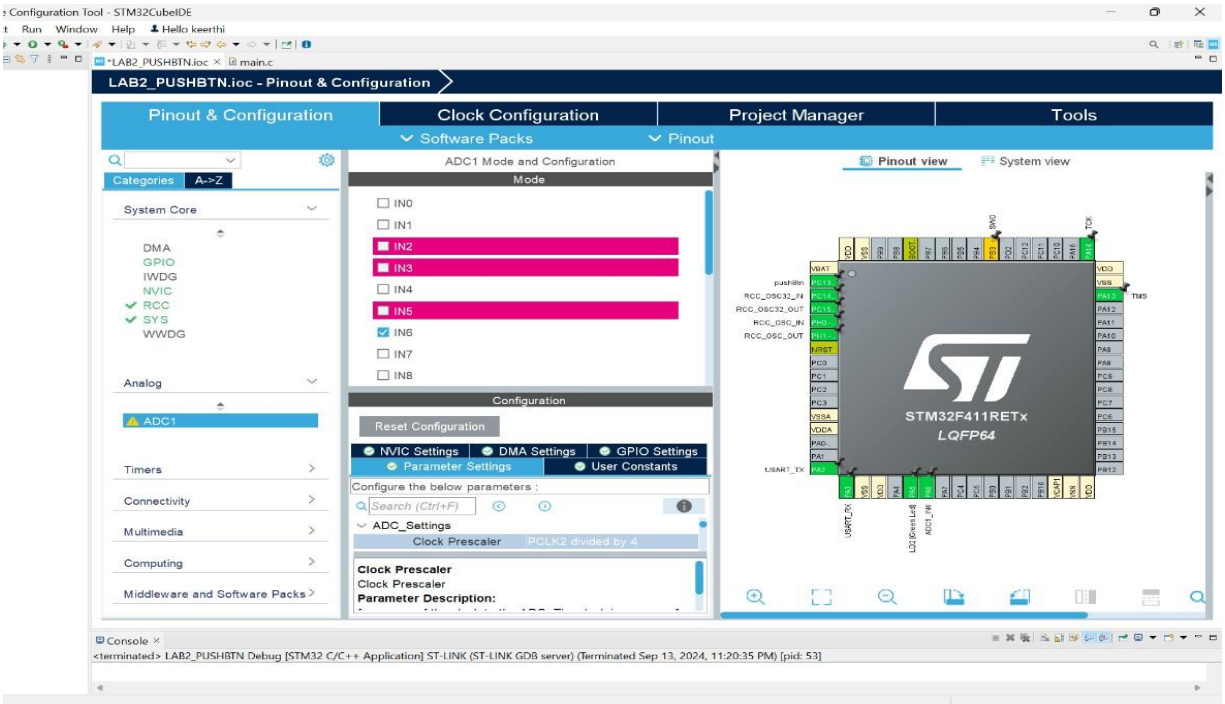
2. The ADC with LDR – Single Conversion Mode

Components: To perform this experiment the components required are microcontroller board, USB Cable, software, Bread board, wires, Resistor and LDR sensor.

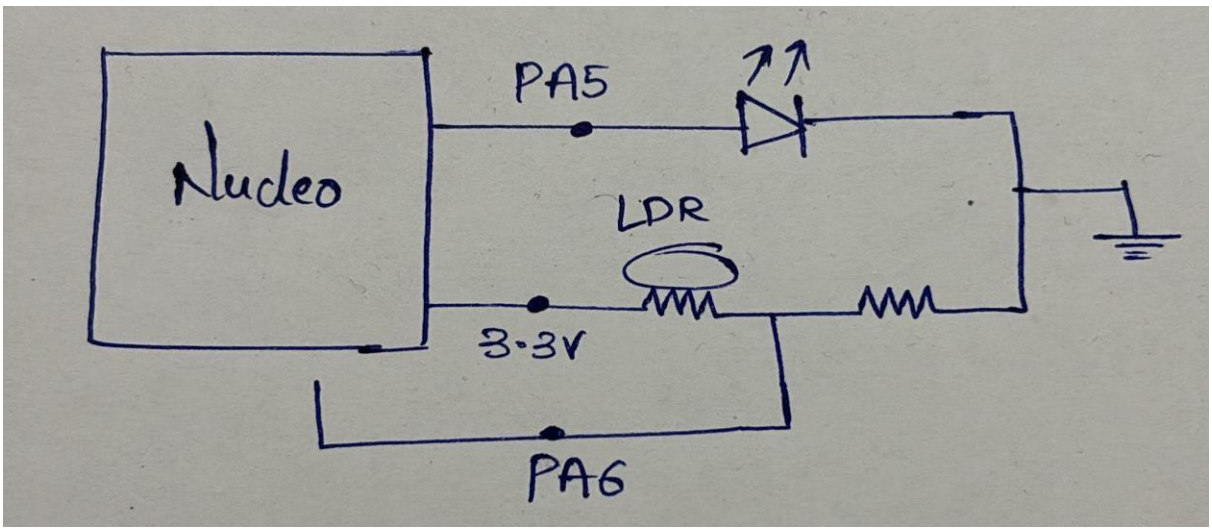
- I have selected the pin **PA6** as **ADC1_IN6.**



- Under Analog is ADC1 , I have checked for the Clock Prescaler and it is PCLK2 divided by 4 (which exactly meets the requirement for the experiment to perform).



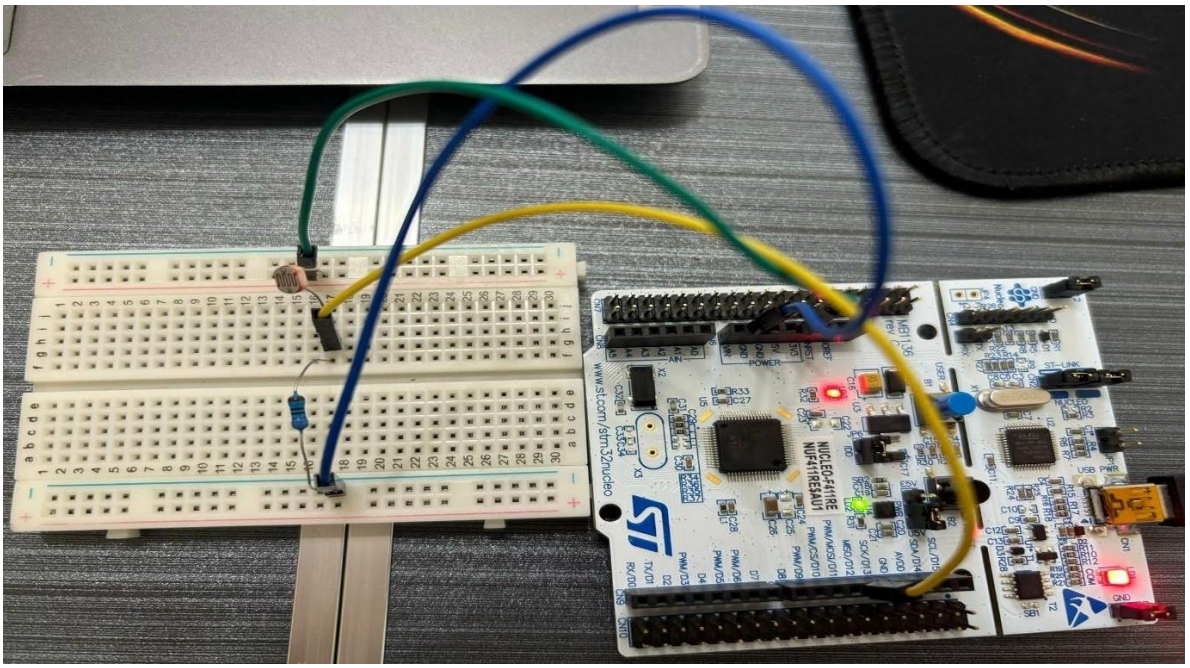
- The below is the circuit diagram for the experiment. The hardware connections are made according to the circuit diagram.



- The below image depicts the hardware setup for the experiment to perform. The connections are made as follows

1.LDR: One of the terminal is connected to 3.3v on the board through a wire. The other terminal of LDR is connected to PA6 (because we configured the PA6 pin to ADC1_IN6).

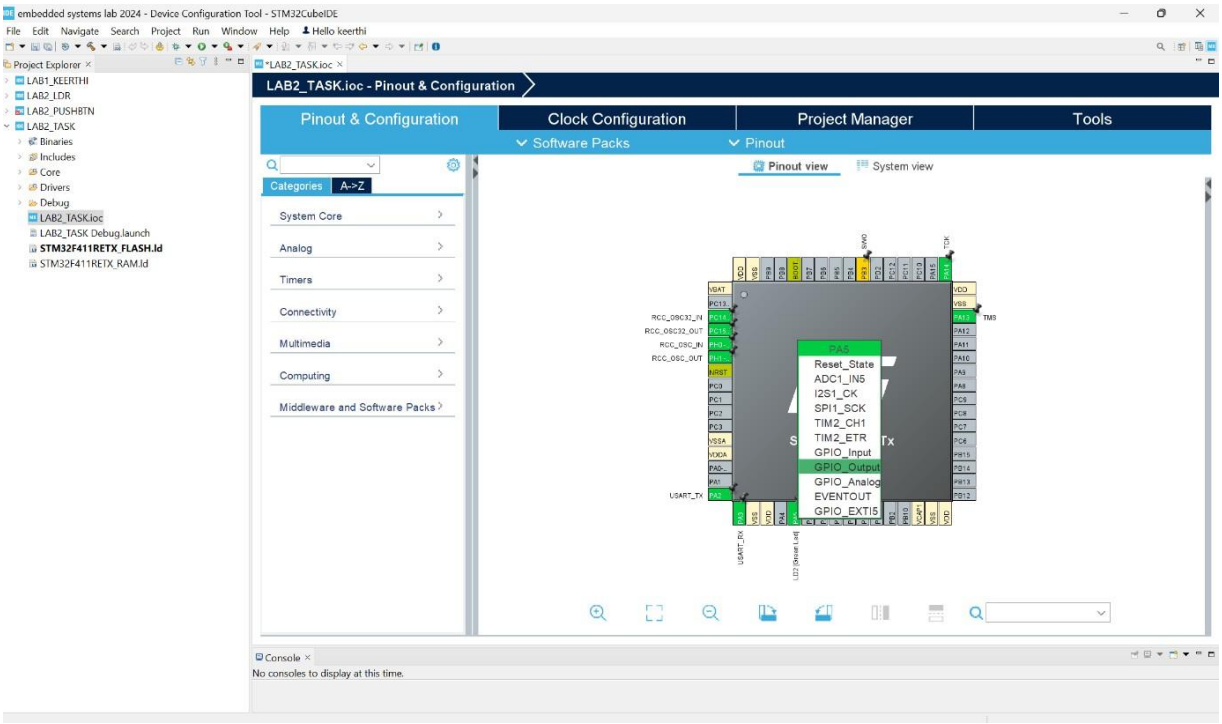
2. Resistor: One of the resistor terminal to connected in parallel to the LDR terminal that is connected to PA6 pin. Other terminal of resistor is connected to the ground using a wire to the board. Pin PA5(LD2 PIN) is internally connected to LED on board.



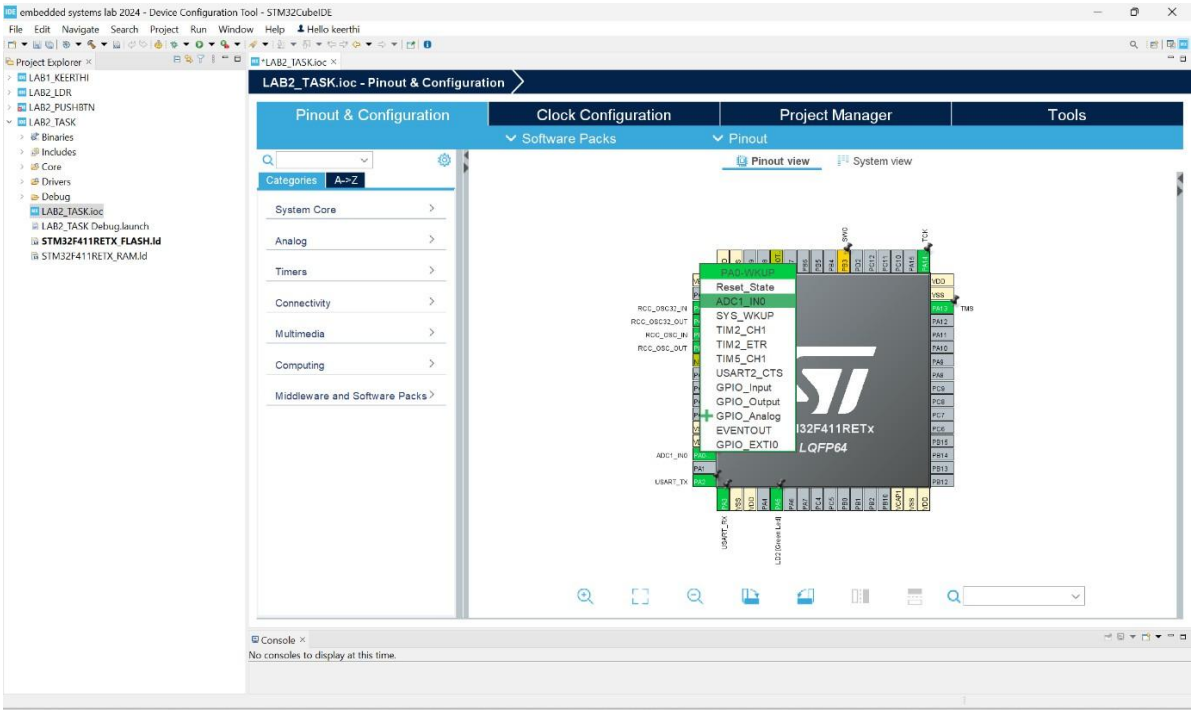
3. Both the LDR is covered, and the push button is pressed simultaneously.

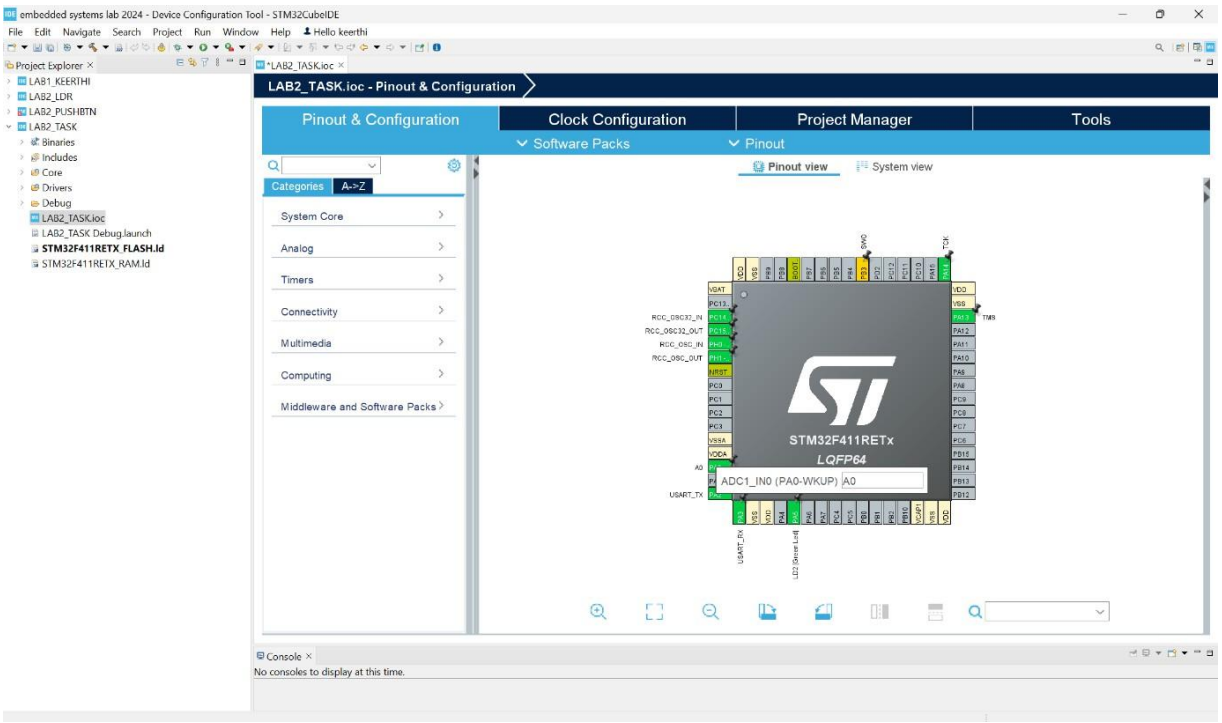
Components: To perform this experiment the components required are microcontroller board, USB Cable, software, Bread board, wires, Resistor and LDR sensor.

- I have set the Pin **PA5** to **GPIO_Output**.

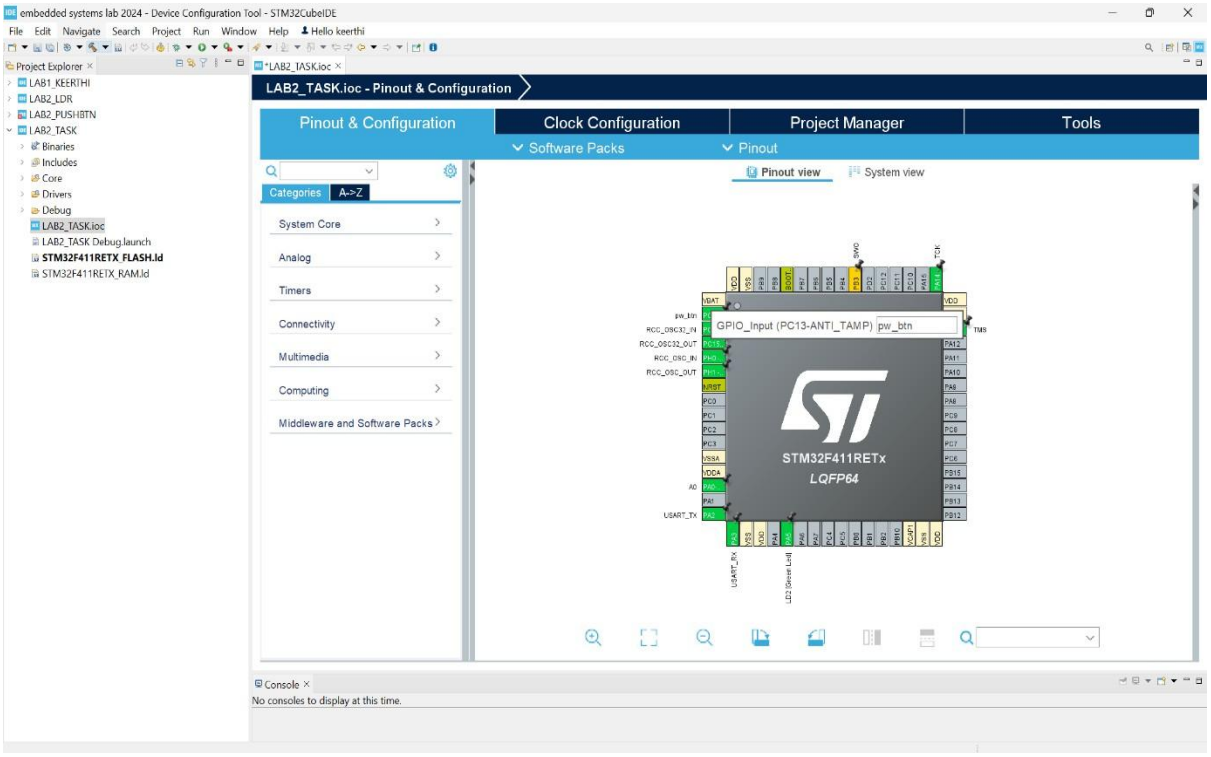


- Then, I have chosen the Pin PA0 and set the pin **PA0** to **ADC1_IN0** and named it as **A0**.

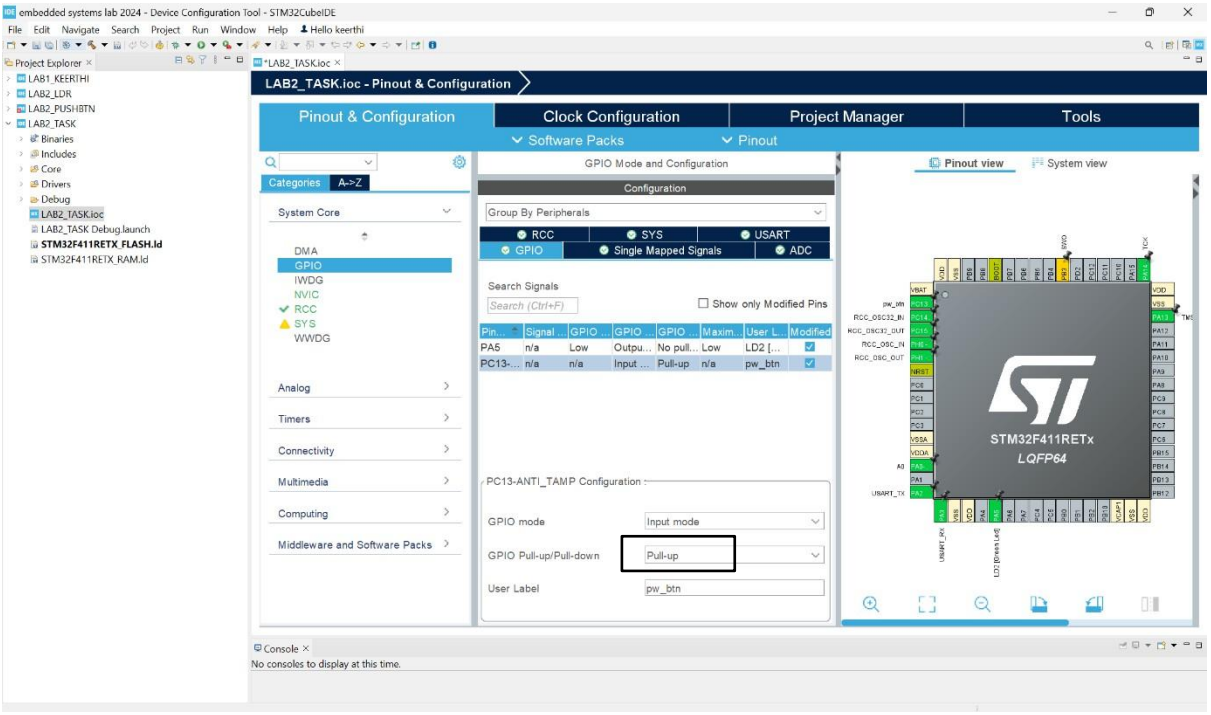




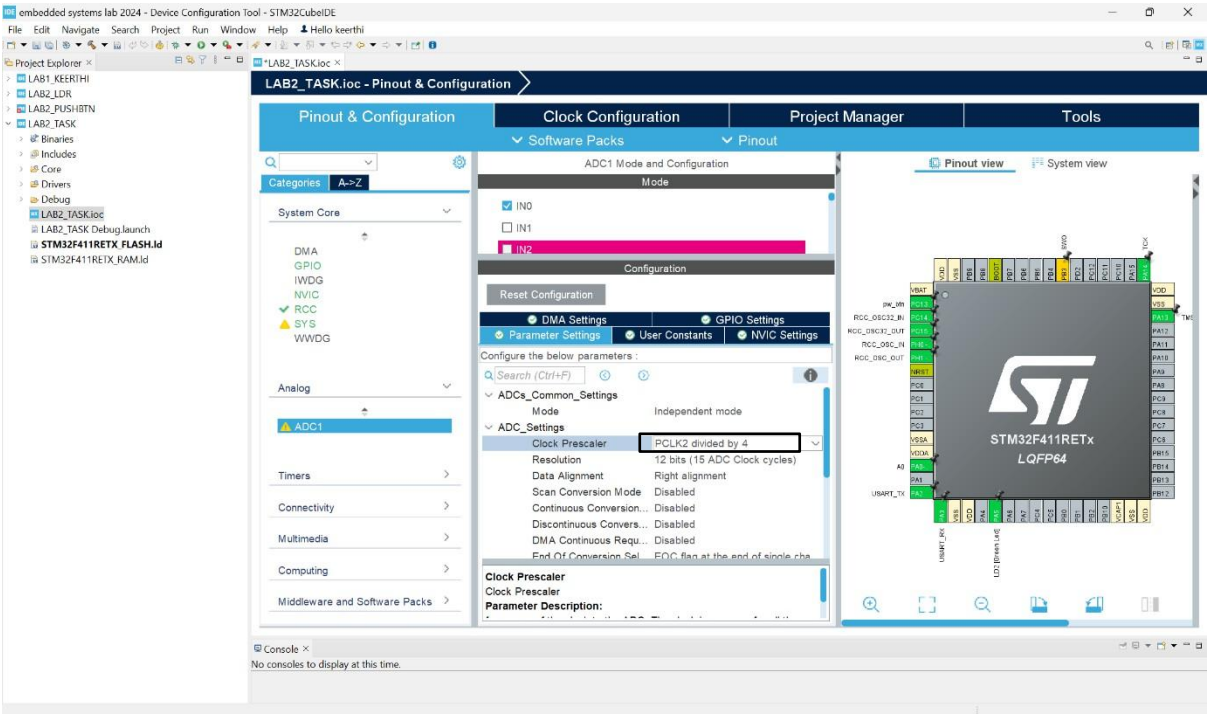
- I have made the Pin **PC13** as **GPIO_Input** and named it **pw_btn**.



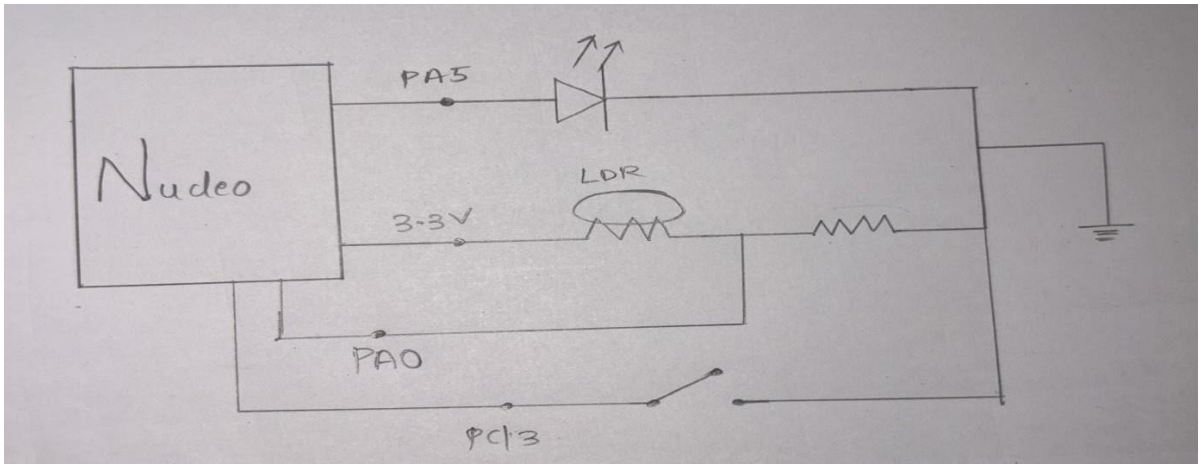
- Under System Core in GPIO, I have selected **PC13** and made **GPIO Pull-up/Pull-down** to be **Pull-up.**



- Under Analog is ADC1 , I have checked for the Clock Prescaler and it is PCLK2 divided by 4 (which exactly meets the requirement for the experiment to perform).



- The below is the circuit diagram for the experiment. The hardware connections are made according to the circuit diagram.

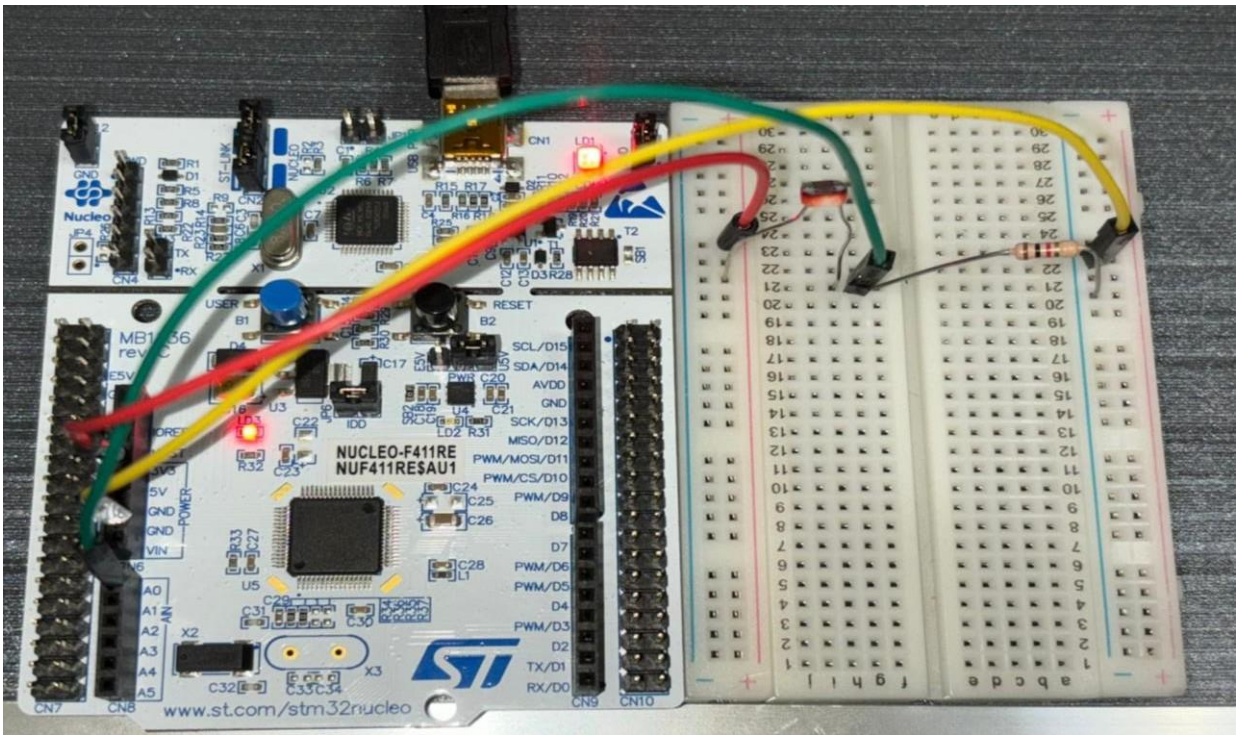


- The below image depicts the hardware setup for the experiment to perform. The connections are made as follows

1.LDR: One of the terminal is connected to 3.3v on the board through a wire. The other terminal of LDR is connected to A0 (because we configured the PA0 pin to ADC1_IN0).

2. Resistor: One of the resistor terminal to connected in parallel to the LDR terminal that is connected to A0 pin. Other terminal of resistor is connected to the ground using a wire to the board.

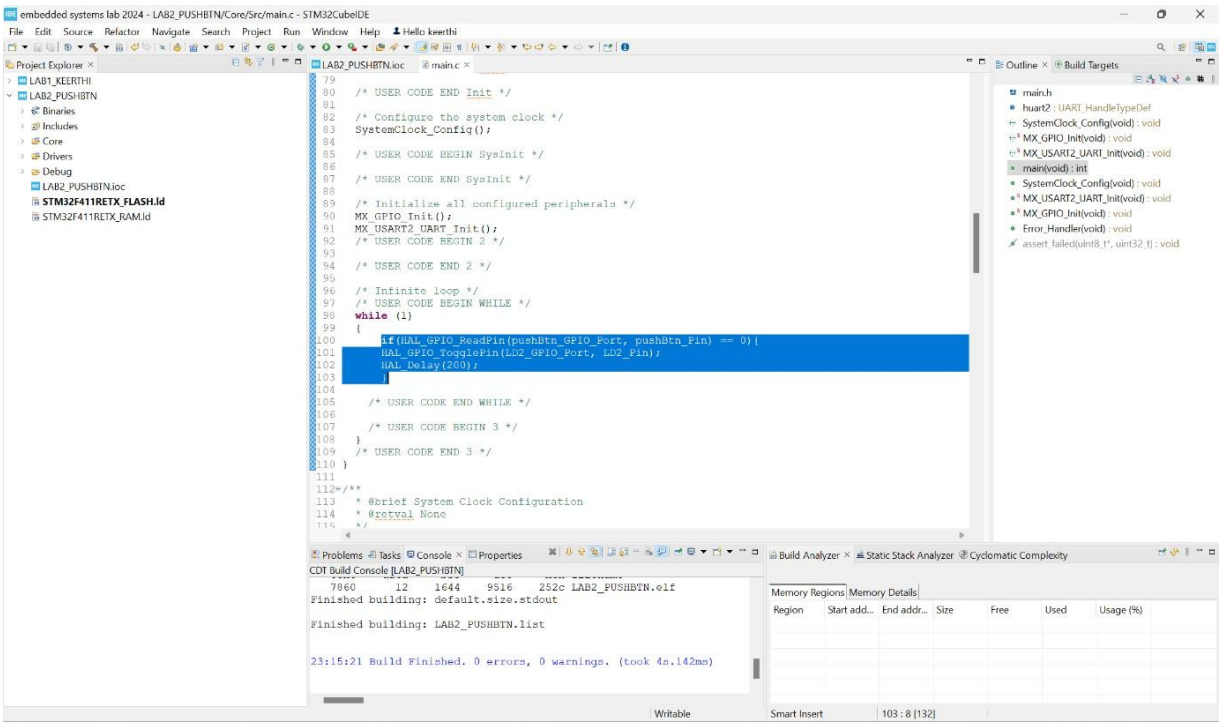
3. PC13: It's connected to the input of the switch or signal source. The other side of switch or signal source is connected to the ground. Pin PA5(LD2 PIN) is internally connected to LED on board.



C. Code Explanation:

1. LED Blinking using Push button:

This code is sort of reading the status of the push button. When the button is pushed its reads LOW then the status of LED will be toggled and a short delay i.e 200ms is added to avoid fast toggling.



Code:

```
if (HAL_GPIO_ReadPin(pushBtn_GPIO_Port, pushBtn_Pin) == 0)
{
HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
 HAL_Delay(200);
}
```

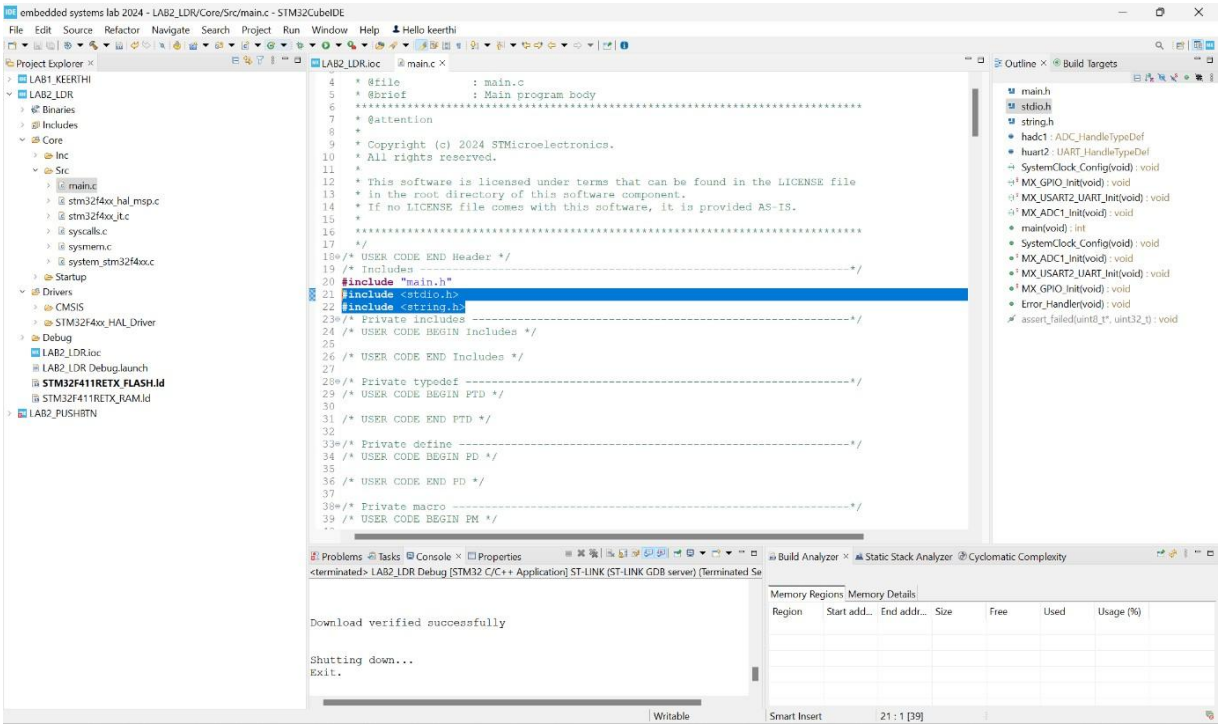
It basically reads the present state of the Pin PC13 and returns if the pin is HIGH[1] or LOW[0]. This is the 'if condition' where it says if the button is pushed that indicates the pin is LOW[0] and if its true the code inside the loop gets executed.

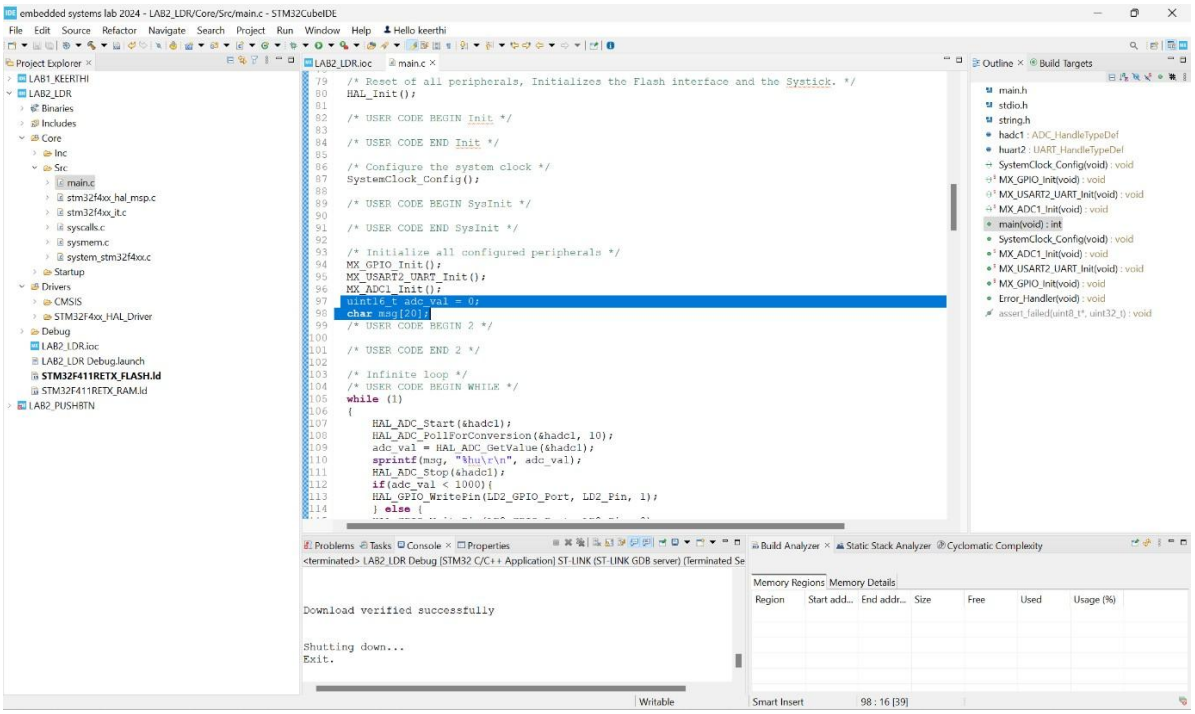
It introduces delay with 200ms once after LED gets toggled.

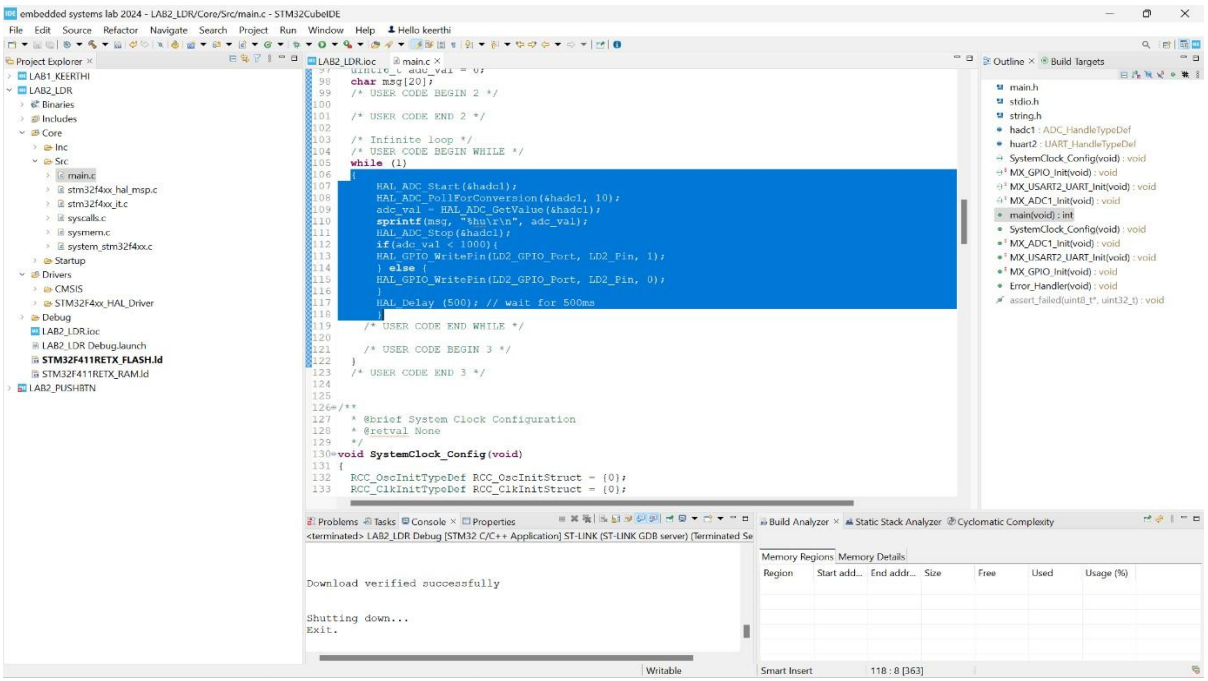
Once the button is pushed, this lines basically toggles the state of LD2_pin i.e, PA5 which is connected to LED and changes the state of pin. If the present state is HIGH[1] it switches to LOW[0] and viceversa.

2. The ADC with LDR – Single Conversion Mode

In this code basically when LDR is covered the LED is ON and when its not covered LED is OFF. So in the code it reads the ADC value and if the value of ADC is above 1000 LED turns OFF and turns ON when ADC Value is less than 1000 (threshold).







Code:

```
#include <stdio.h>
#include <string.h>
```

Header files and <string.h> is used to format the ADC value to string.

```
uint16_t adc_val = 0;
char msg[20];
```

Holds the ADC value that is read from LDR sensor i.e., 16-bit

The string i.e., represented for the ADC value is stored in char.

```
while (1)
  {
```

```
HAL_ADC_Start(&hadc1);
```

This starts the ADC Conversion.

```
HAL_ADC_PollForConversion(&hadc1, 10);
```

This actually waits for the conversion of ADC to complete within 10ms of time.

```
adc_val = HAL_ADC_GetValue(&hadc1);
```

The converted ADC Value is stored in adc_val.

```
sprintf(msg, "%hu\r\n", adc_val);
HAL_UART_Transmit(&huart2, msg, strlen(msg), HAL_MAX_DELAY);

HAL_ADC_Stop(&hadc1);
```

Once the ADC value is collected the ADC is stopped

```
if(adc_val < 1000){
    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
}
```

If condition satisfies then LED turns ON

```
else
{
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 0);
}
```

Turns OFF the LED if adc_value is greater than 1000.

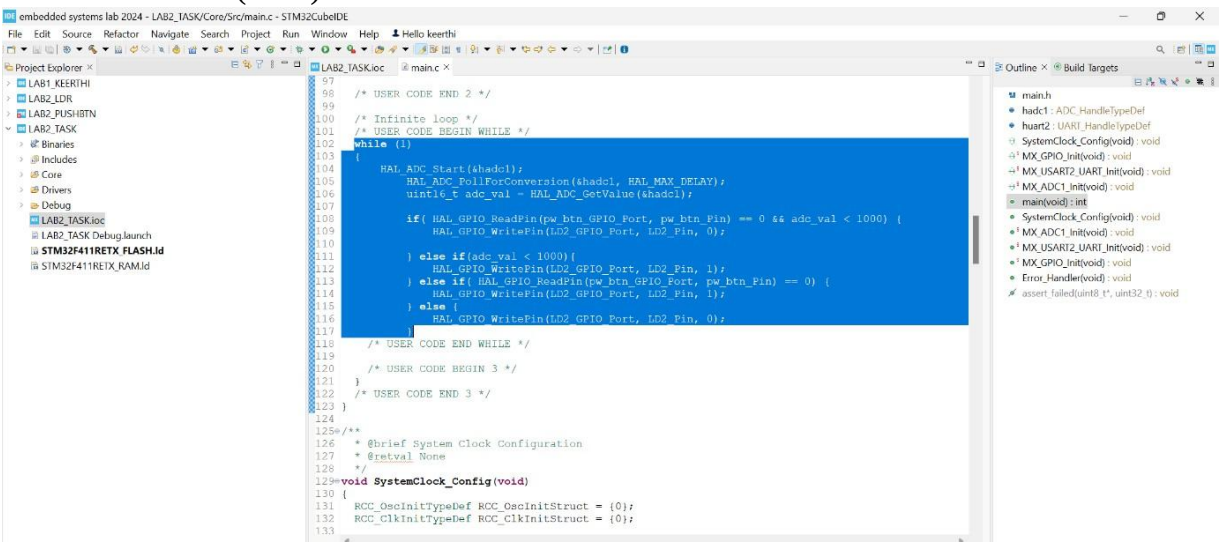
This is if condition its checks if the value of ADC IS less than 1000.

```
HAL_Delay (500);
}
```

Delay

3. Both the LDR is covered, and the push button is pressed simultaneously.

This code states tells that state of LED is a function of readings from an ADC and also push button state. LED gets activated or turned on/off based on:
Whether I pressed the push button or not, whether ADC value is below a threshold (1000) or above a threshold (1000) or both.



Code:

```
while (1)
 {
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
uint16_t adc_val = HAL_ADC_GetValue(&hadc1);
if( HAL_GPIO_ReadPin(pw_btn_GPIO_Port, pw_btn_Pin) == 0 && adc_val < 1000)
 {
 HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 0);
 }
 else if(adc_val < 1000)
 {
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
 }
 else if( HAL_GPIO_ReadPin(pw_btn_GPIO_Port, pw_btn_Pin) == 0)
 {
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
 }
 else
       {
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 0);}
```

while (1) → It creates an infinite loop, where the code runs continuously in the loop.

HAL_ADC_Start(&hadc1); → This starts the ADC conversion

HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY); → it waits until the ADC conversion is Completed.

uint16_t adc_val = HAL_ADC_GetValue(&hadc1); → Reads the value of ADC and obtains the result of ADC conversion and adc_val stores the digital value of the corresponding analog input.

This basically checks the condition if push button (pw_btn) is pressed it returns 0 and if ADC value is below 1000, if both of the conditions satisfy then the LED turns OFF.

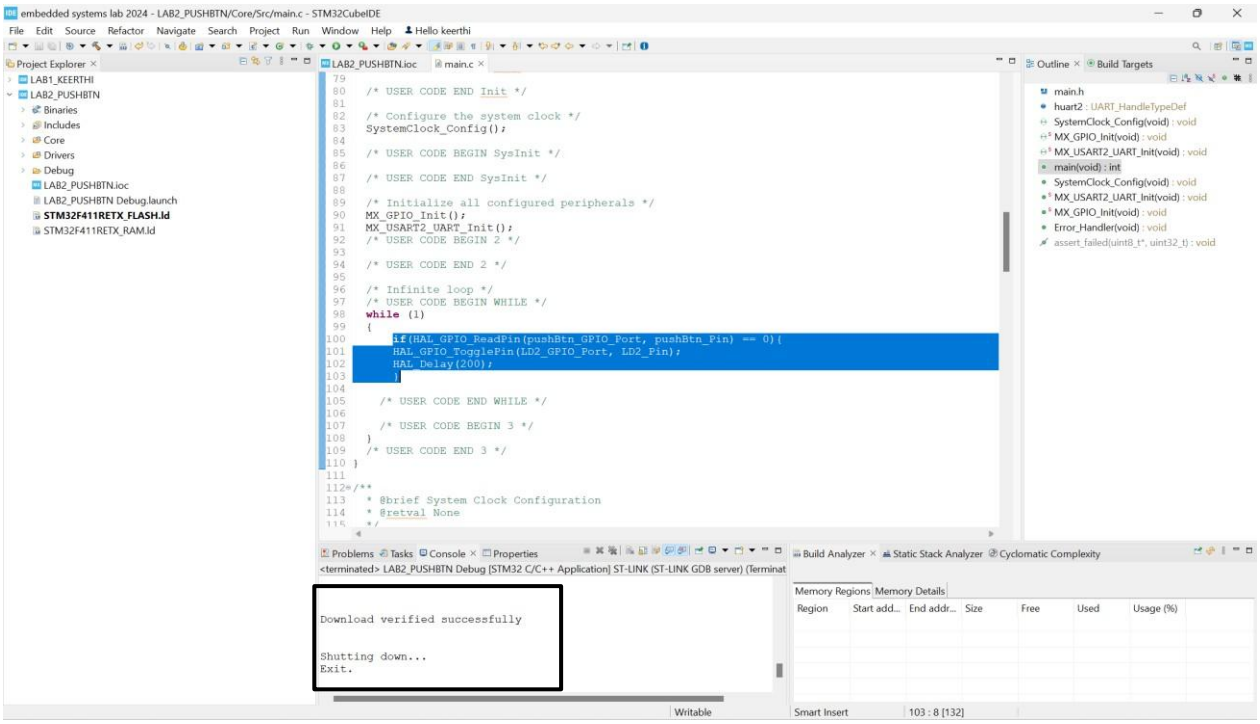
This is the second condition if ADC Value is below 1000. The LED turns ON.

This condition says if button is pushed then LED turns ON.

If none of the above conditions are met then LED turns OFF.

D. Observations:

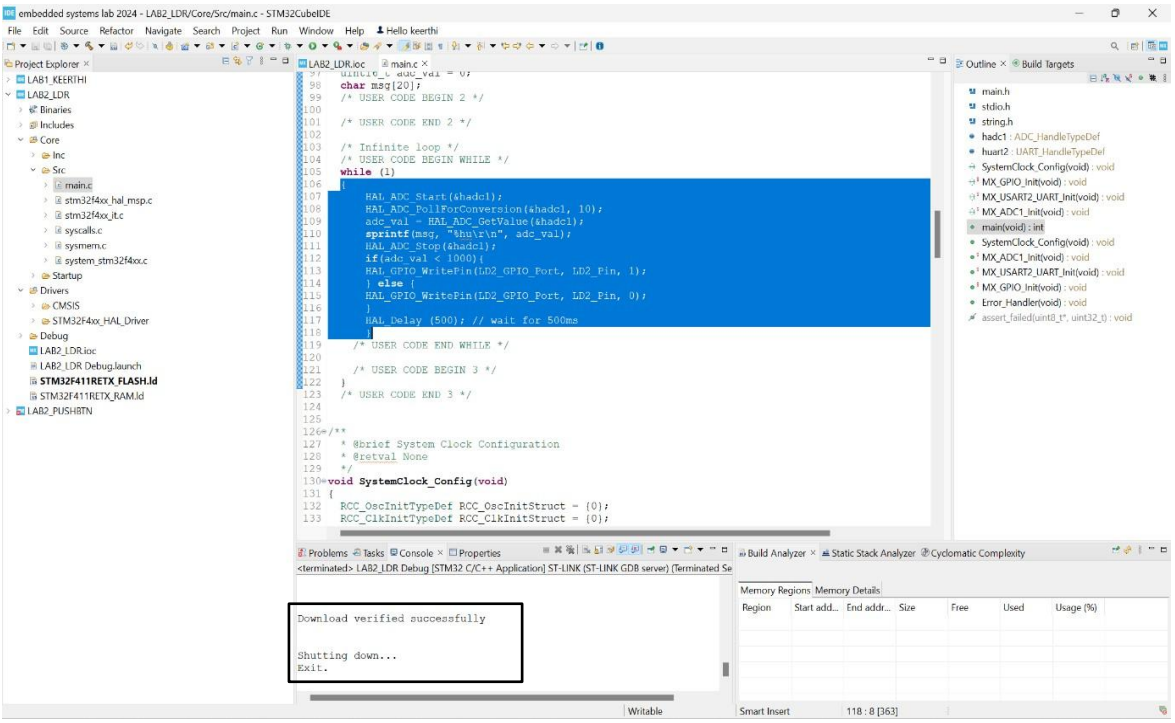
- **The push button is pressed (GPIO Input experiment).**



- After successfully, running the code I have observed that when I pushed the button on the board the LED glows and when I again push the button the LED goes OFF it continuously happens vice versa as many times I push the button on the board.
- From my observation, I basically understood the functionality of GPIO on the board, where it allowed me to push the button externally and the output is visualized as LED toggling.
- Below is the video demonstration of the experiment that I have performed.

  https://drive.google.com/file/d/12R5IY2mSoD5VBxnZmex_IVIIfkliyR8z/view?usp=drive_link

- **The LDR is covered (ADC experiment).**



- While implementing, according to the code the LED is ON when the LDR is covered. Here, I have observed that the LED control can be done by using LDR sensor.

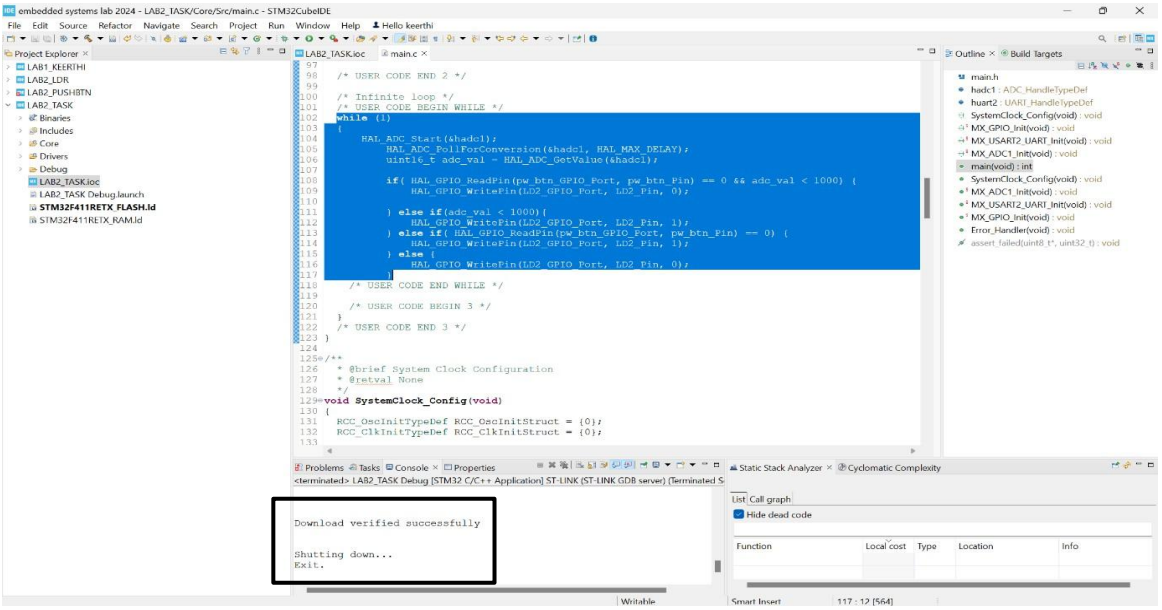
- The LDR is a photoresistor sensor that changes resistance according to the intensity of light. Here in this circuit as intensity of light increases the resistance of the LDR decreases and the voltage at A6 increases and viceversa.

- The LDR is connected in parallel to the resistor on the bread board. A6 is a common pin between the LDR and resistor which acts as voltage divider. since we connected 3.3v to the LDR and the light intensity is the other voltage where the ADC is used to convert Analog input to digital output and A6 reads the ADC value this leads the microcontroller to turn ON or OFF the LED.

- Here in this experiment, I have observed that the covering the LDR produces a low ADC value which causes to light the LED ON.

- Below is the video demonstration of the experiment that I have performed.
  https://drive.google.com/file/d/1QZHkEgMT8vX_1qcRT1eyX83fTohUPQqO/view?usp=drive_link

- **Both the LDR is covered, and the push button is pressed simultaneously.**



- While implementing, according to the code I have observed that the LED is OFF when the button is pressed and the LDR sensor is covered simultaneously. But the LED is ON when the button is pressed, or if the LDR sensor is covered. Here, I have observed that the LED control can be done by using LDR sensor and push button.

- The LDR is a photoresistor sensor that changes resistance according to the intensity of light. Here in this circuit as intensity of light increases the resistance of the LDR decreases and the voltage at A0 increases and viceversa.

- The LDR is connected in parallel to the resistor on the bread board. A0 is a common pin between the LDR and resistor which acts as voltage divider. since we connected 3.3v to the LDR and the light intensity is the other voltage where the ADC is used to convert Analog input to digital output and A0 reads the ADC value this leads the microcontroller to turn ON or OFF the LED.

- PC13 is basically GPIO pin connected to the push button where once pushed it turns on the LED and turns OFF viceversa.

- Here in this experiment I have made the LED OFF, where simultaneously button is pushed and LDR is covered (when LDR is covered light intensity decreases and resistance of LDR increases).

- Below is the video demonstration of the experiment that I have performed. https://drive.google.com/file/d/1eVlO2yeX9VGipGnWHWaoXhebKFC66spM/view?usp=drive_link

E. Conclusion:

In these experiments, I have successfully demonstrated and gained both knowledge and hands-on experience with the GPIO input and ADC functionality on the Nucleo board. The push button and LDR demonstrated the function of digital and analog inputs, which took decisions for outputs. I have also added in delays and threshold that allowed me to check how the system behaves in different conditions, which gave me a little more insight as to how GPIO and ADC can be used practically.

Critical Thinking:

1: How does the Pull-up resistor influence the button's operation in GPIO?
A: while doing the GPIO configuration the pin PA13 is made to be pull-up. So, basically pull-up register keeps the GPIO pin always high when Button is not pushed, but when pushed it changes to LOW state.

2 : Why is it necessary to include a delay (HAL_Delay(200)) in the button press code?
A: The delay (HAL_Delay(200)) in the button press code is necessary for Debouncing and to ensure a single push.

3: What could be the impact of changing the ADC threshold value (1000) on the behavior of the system?
A: Threshold < 1000 : If the ADC threshold value is set lower than 1000 then it makes the LED to turn ON in low light conditions.so, its more sensitive to light.
Threshold > 1000 : If the ADC threshold value is set greater than 1000 then to turn the LED ON it requires more light.so, its less sensitive to light.

4: What other sensors could you use with the ADC besides the LDR?
A: The choice of sensor depends on the specific application. We can use Thermistors sensor, potentiometers, pH Sensor and IR sensor for motion detection.