Practical Sheet Nº 2 (Basic Input & Output)

Objectives	Duration
At the end of this practical sheet, you will be able to: • Use the Arduino IDE to compile and run a program written in ArduinoC. • Use the ESP32s DEVKIT to execute source code and perform debugging via UART. • Program in Arduino C using decision and repetition structures.	30 min.

1. Using the circuit implemented in the previous training sheet, add a pushbutton to the circuit using pin G25 to read the state of the button to look like Figure 1.

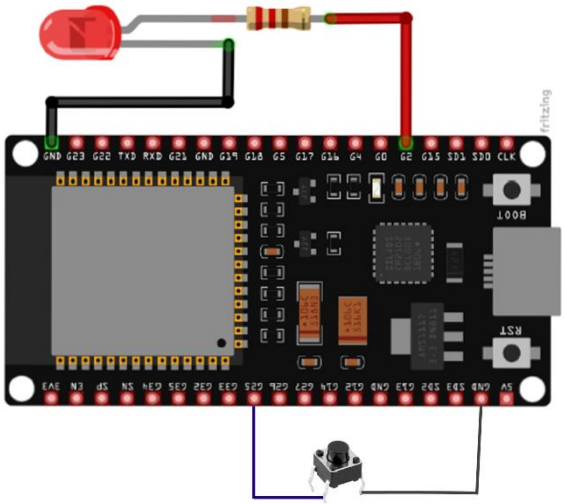

Figure 1: Button and LED circuit

2. Create a new sketch in the Arduino IDE and paste the following code:

```
//////////////////////////////////////////////////
// Practical -
// MECHAUZC2023
//////////////////////////////////////////////////

#define BUTTON_PIN 25
#define LED_PIN 2

// variables
int led_state = LOW;    // the current state of LED
int button_state;       // the current state of button
```

```
int last_button_state;  // the previous state of button

void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT_PULLUP);

button_state = digitalRead(BUTTON_PIN);
}

void loop() {
last_button_state = button_state;      // save the last state
  button_state = digitalRead(BUTTON_PIN); // read new state

if (last_button_state == HIGH && button_state == LOW) {
    Serial.println("The button is pressed");

    // toggle state of LED
    led_state = !led_state;

    // control LED arccoding to the toggled state
    digitalWrite(LED_PIN, led_state);
  }
}
```

The state of the button is performed by the function **digitalRead()** and every time the button is pressed the variable of the state of the LED is toggle before writing the state to the LED´s GPIO.

3. Implement a program that counts the number of clicks on the button. The program should the accumulated number of clicks as soon as a new click is after a new click is counted.
   Note that in a real circuit, it will be necessary to use a strategy to eliminate the multiple bounces that occur in the electrical signal after the button is clicked, as shown in the Figure 2. The "debounce" can be done by two different methods:
   - **by hardware** using a Low Pass Filter (LPF)
   - **by software**, by including a waiting time T, right after the first transition, to ensure that the multiple transitions that occur, for mechanical reasons inherent to the button, are not accounted for.

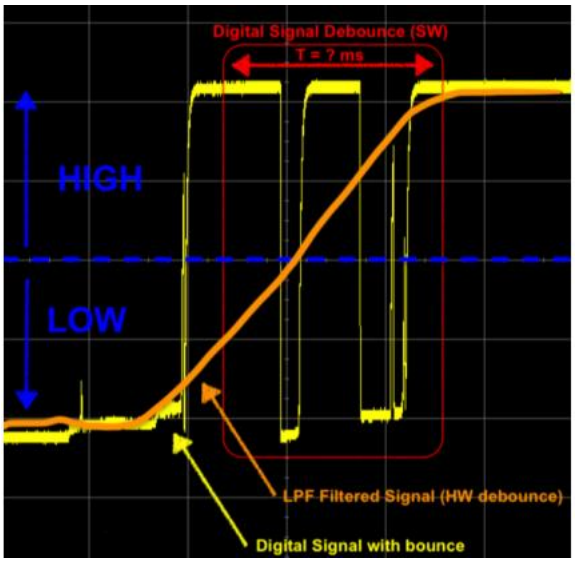

Figure 2: Signal debouncing

4. Implement software debouncing.
   Above is a solution to implement software debouncing for the button.

```
//////////////////////////////////////////////////
// Practical -
// MECHAUZC2023
//////////////////////////////////////////////////
#define BUTTON_PIN 25
#define LED_PIN 2
#define DEBOUNCE_TIME  50

// variables
int led_state ;    // the current state of LED.
int button_state=LOW;       // the current state of button
int last_button_state;
int last_Flickerable_State = LOW;  // the previous state of button
int counter=0;

unsigned long lastDebounceTime = 0;
```

```
void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT_PULLUP);
button_state = digitalRead(BUTTON_PIN);
}
void loop() {
button_state = digitalRead(BUTTON_PIN); // read new state

if (button_state != last_Flickerable_State){
  lastDebounceTime = millis();
  last_Flickerable_State = button_state;
}
if((millis()-lastDebounceTime)>DEBOUNCE_TIME ){
  if (last_button_state == HIGH && button_state == LOW) {
    Serial.println((String)"The button is pressed | counter= "+counter++);
    // toggle state of LED
    led_state = !led_state;
    // control LED arccoding to the toggled state
    digitalWrite(LED_PIN, led_state);
  }
  last_button_state=button_state;
}
}
```