THE ASSEMBLY OF A RESPONSIVE PUSH BUTTON PROGRAM USING ARDUINO

UNIVERSITY OF THE PHILIPPINES VISAYAS

COLLEGE OF ARTS AND SCIENCES

DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS

CMSC 130 - LOGIC DESIGN AND DIGITAL COMPUTER CIRCUITS

2ND SEMESTER AY 2021-2022

ASSIGNMENT 4. PUSH BUTTONS IN ARDUINO

PREPARED BY:

DELA CRUZ , LLOYD WALLYS

LESCANO , RENMAR

ORAÑO , MYRTLLE GEM

MAY 9, 2022

Abstract

The purpose of this laboratory assignment is to build an Arduino circuit that receives input from a pushbutton and produces output through an LED. The system was implemented with an Arduino UNO together with components from *Makerlab Electronics' Arduino Upgraded Learning Ki*t and programmed in the proprietary IDE with the *ezButton* library imported. The system uses the pushbutton in two ways: (1) to turn the LED pin on and off with short-presses, and (2) to set the LED pin to a blinking state with a long-press lasting more than 5 seconds. Print flags were added to the code, which aided in testing and debugging the software, and were checked with the Arduino IDE's serial monitor. The developed program proved functional in correctly manipulating the static LED state— although, a brute-force workaround was developed to reliably toggle the blinking state.

Table of Contents

Parts **Pages**

List of Figures

Parts **Pages**

I. Design

A. Problem Statement:

The main objective of this project is to construct a system using Arduino hardware and write a corresponding software to receive input from a push button to produce output through an LED (Light Emitting Diode). When the button is pressed, the LED pin must switch to on (HIGH) and switch to off (LOW) when the button is pressed again. Each time the button is pressed, the LED cycles between being switched on and off. Additionally, the Arduino circuit must detect a prolonged push of the button. If the button is pressed for longer than 5 seconds, the LED pin should blink automatically, cycling on and off with a one-second interval. Furthermore, debouncing the input from the pushbutton must also be implemented.

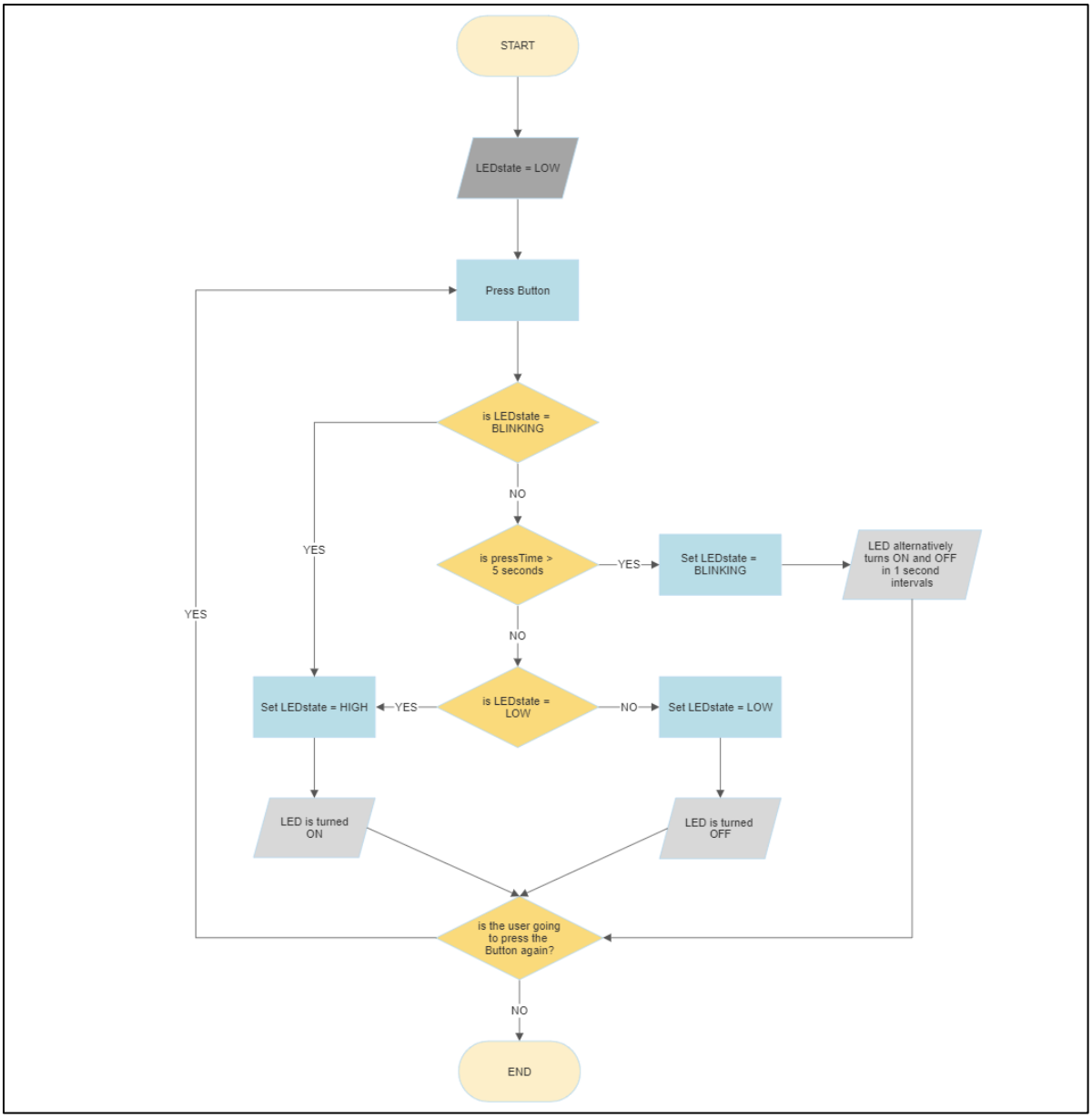

Fig. 1. The flowchart of the push-button process

B. Overview:

For a systematic procedure, the following objectives were pursued:

- Understand Arduino basics (setup, language, terminal)
- Understand Arduino buttons
- Understand how LEDs and resistors work
- Understand the breadboard connections
- Learn how to troubleshoot Arduino hardware and software

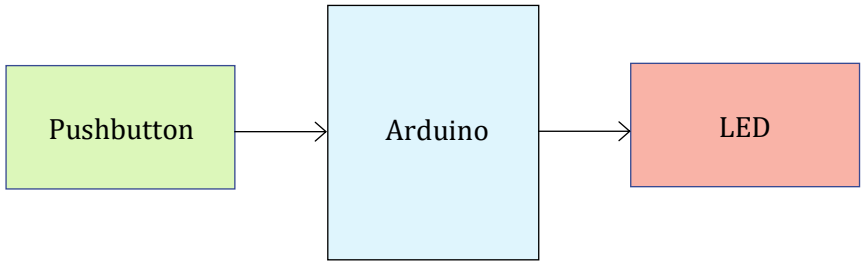

Fig. 2. The block diagram of the push button system

In summary, a full circuit must be created by pressing a button initiating the interaction between the breadboard circuit and the Arduino system. The initial short button press, making contact with the circuit for a fraction of a second, switches the LED to static on. Conversely, the next short press closes the connection, cutting off the current source and turning off the LED.

> *When the switch is closed, the two contacts detach and reattach 10 to 100 times in less than one millisecond (Horowitz & Hill, 1989).*

However, it is important to understand that pressing the button without considering *debouncing* may result in unpredictable results.

Due to mechanical and physical malfunctions, pushbuttons frequently generate false transitions when pressed; these transitions might be perceived as multiple presses in a brief period, misleading the execution. *Debouncing*, facilitates a verification stage the pushbutton briefly to ensure a realistic and accurate input. Specifically for this experiment, the implementation of debouncing pertains to *setting up an interval* after the initial button contact where the circuit ignores (bounces back) any succeeding input.

Using the template provided in the Arduino samples, a 50 msec interval is set for the debounce. As Ganssle (2004) argues, a 20-50 msec debounce interval is practical and realistic for an average human-triggered press. Thus, the debounce was set accordingly to detect only one press every 50 milliseconds.

C. Hardware:

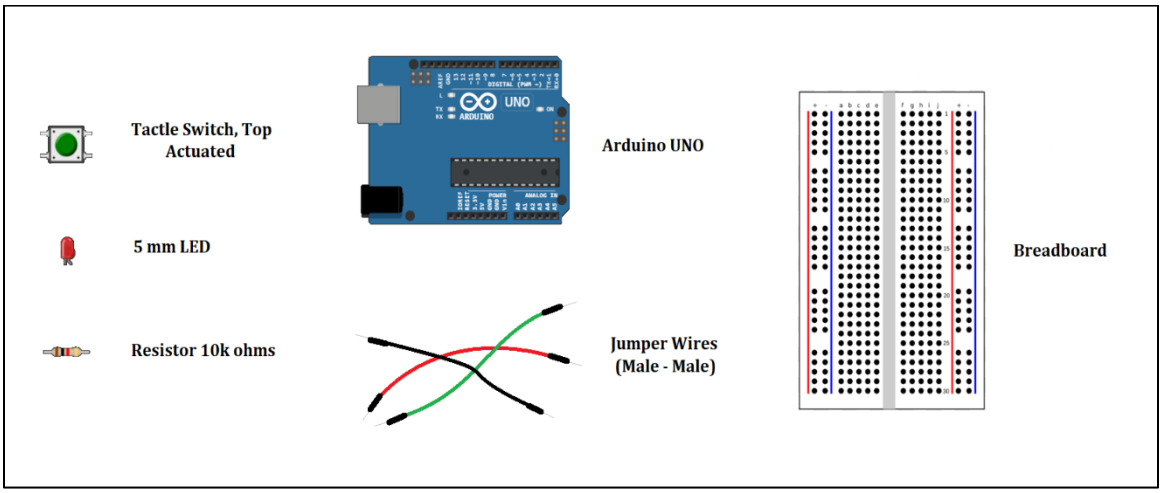

Fig. 3. The hardware needed in the push-button program

The following hardware components were used for the push button program:

- Switch – to obtain the program input.
- LED – to produce the program output.
- 10K Ω resistor – to regulate the flow of electrical current and to supply voltage to the active device acting like a transistor.
- Arduino Uno – the platform to facilitate the interaction between software and hardware components
- Jumper Wires – to connect the pins of each component
- Breadboard – to facilitate the flow of electricity between the hardware components.

D. Circuit:

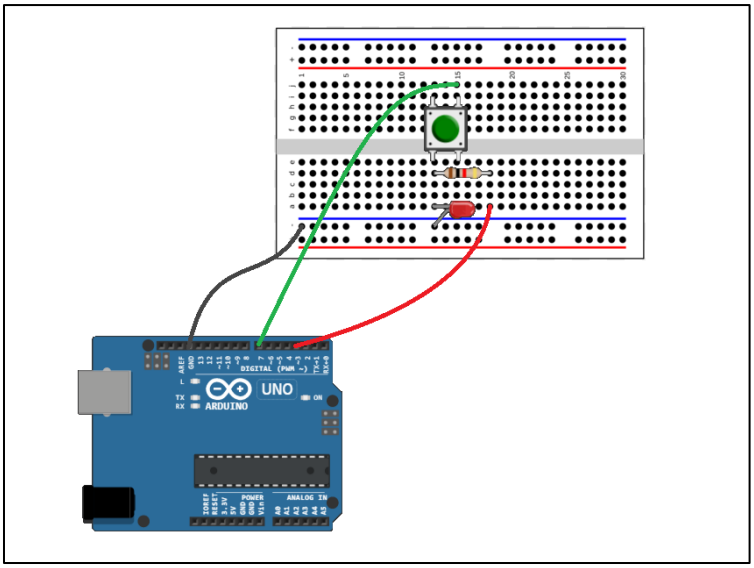

Fig. 4. The setup model of the Arduino push button system

To build the circuit for this experiment, we begin by connecting a black wire from the Arduino's ground port GND (3) to the negative breadboard rail. Correspondingly, the LED's cathode (shorter leg) is then attached to the negative rail. Next, the anode of the LED is linked to one leg of the 10k Ω resistor going to one leg of the button. Meanwhile, the other leg of the resistor is connected to pin 3 of the Arduino through a jump wire (red). This setup allows the LED to be turned off and on by means of restricting or allowing the flow of electricity. Finally, another leg of the button, diagonally oriented from the first leg, is connected to pin 7.

E. Schematic:

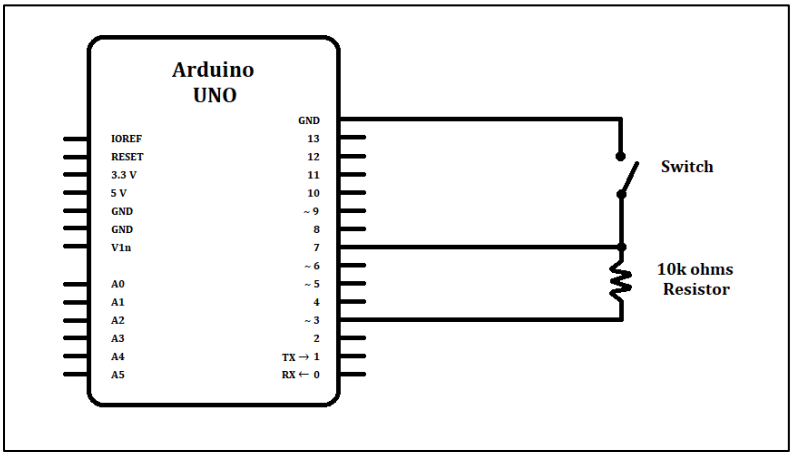

Fig. 5. The schematic for the Arduino push button system

F. Required Libraries:

The *ezButton* library was included into the Arduino file as part of the program. This library introduces convenient functionalities for pushbuttons and debouncing.

G. Code:

```
#include <ezButton.h>

const int BUTTON_PIN = 7;
const int LED_PIN = 3;
const int LONG_PRESS_TIME  = 5000;

ezButton button(BUTTON_PIN);        // library for debounce

int ledState = LOW;                 // default LED OFF

unsigned long pressedTime  = 0;     // start
unsigned long releasedTime = 0;     // end

long pressDuration;                 // duration (end-start)
bool isPressing = false;            // allow detect while pressing
bool startBlink = false;            // switches to toggle blink
bool stopBlink = false;

void setup() {
  Serial.begin(9600);               // initiate terminal
  button.setDebounceTime(50);       // debounce of 50ms
}
```

Fig. 6. Declarations, Assignments, and Setup

```
void loop() {
  button.loop();                    // prerequisite call

  if(button.isPressed()){
    stopBlink = true;               // stop blinking of applicable
    isPressing = true;
    pressedTime = millis();
  }

  if(button.isReleased()) {
    isPressing = false;
    releasedTime = millis();

    pressDuration = releasedTime - pressedTime;
    if(pressDuration < LONG_PRESS_TIME){
      Serial.print("A SHORT press is detected: \t");
      if (startBlink == true && stopBlink == true){       // started blinking; needs to stop
        ledState = HIGH;
        Serial.print("<blink off> \t");
        digitalWrite(LED_PIN,ledState);                   // LED state: STATIC HIGH
        startBlink = false;
    } else {
      ledState = !ledState;                               // toggle LED state (invert): STATIC
      digitalWrite(LED_PIN, ledState);
      }
    Serial.print(ledState);
    Serial.print("\n");
    }
  }
  if(isPressing == true) {
    pressDuration = millis() - pressedTime;
    if(pressDuration > LONG_PRESS_TIME) {
      Serial.println("A LONG press is detected: \t <blink on>");
      startBlink = true;                                                  // toggle LED state: BLINKING
      stopBlink = false;
    }
  }

  if (startBlink == true && stopBlink == false){                  // start blinking; not need to stop
    digitalWrite(LED_PIN, HIGH);                                  // LED state: BLINKING (1 second delay)
    Serial.println("HIGH \t 1");
    delay(1000);
    digitalWrite(LED_PIN, LOW);
    Serial.println("LOW \t 0");
    delay(1000);
  }
}
```

Fig. 7. Main loop

II. Testing and Debugging

Before the code and circuit were finalized, multiple challenges, both minor and critical, were encountered. Specifically, relative to the complexity of this assignment, the minor challenges were:

1. *Mismatch in port assignments between code and circuits.*
2. *Errors in setting up the connections for the Arduino button.*

However, these were immediately addressed (a) by changing the port parameters, and (b) by referring to the Arduino button documentation (see Fig. 8-9).

Fig. 8. The pinout layout for the Arduino Uno kit button

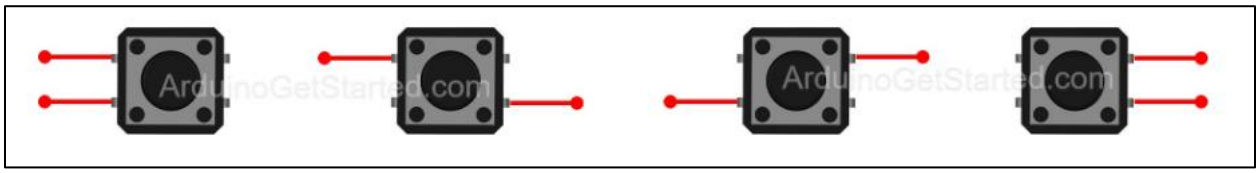

Fig. 9. The four possible connection setups
for the Arduino Uno kit button

Meanwhile, the following critical challenges with their corresponding workarounds were observed:

1. *The unreliable detection of clicks while in the blinking loop*.

To review, in this assignment, the circuit is required to set the LED state to HIGH after the button click that supposedly halts the blinking loop. However, during debugging, it was traced that short button clicks fail to reliably register while the loop is ongoing. To be specific, while there are rare instances where the button press register, the success rate is highly unreliable.

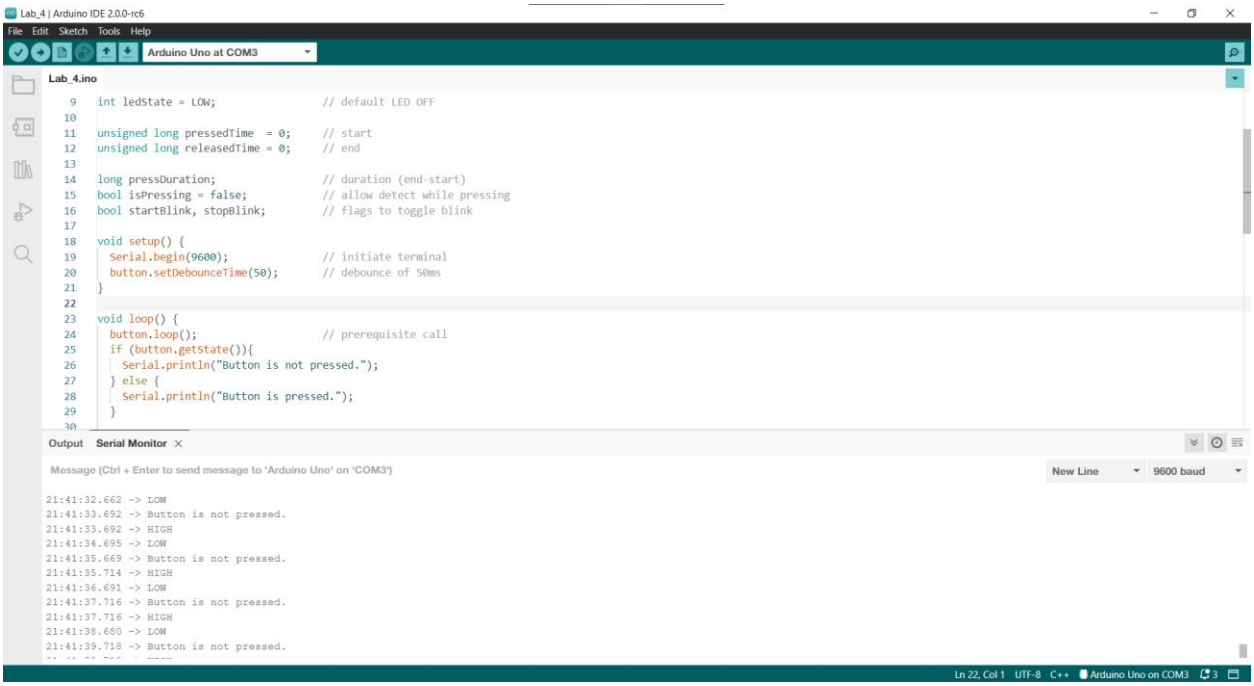

Fig. 10. The illustration of how short button presses are not detected during the main loop

Accordingly, it was assumed that the press needs to be timed exactly 1000 milliseconds after the LED state is set to blinking LOW. Since this reaction time would be impractical, the solution was to perform a long press instead such that the timing would be set in a *brute-force manner*. Purposely, the duration necessary for the press to register is observed to average at around 4000 milliseconds. This coincides with the duration of one blink loop of 2(1000 msec) with some tolerance to cover the interval of the exact timing.

Fortunately, this workaround proved to be successful in addressing the critical issue. Precisely, by performing a long press, the circuit was able to reliably toggle the state between blinking and static HIGH.

2. *The challenge is lies in identifying the optimal number of flag variables.*

Based on the specifications, it is observed that there may be overlaps with the LED states for this circuit. Specifically, it may be set to HIGH during short presses as well as when toggling the blinking state to stop. From this, we can infer that it is necessary to have deterministic switch (or trigger) variables within the respective code blocks.

After a series of tests, it was found that at least two switch variables must be set to trigger the correct behavior, especially for the long-press instances. Particularly, there must be a specific switch variable during a long press, and another for a general (short or long) press. Then, all together, these switches must be considered to trigger the blinking state. Otherwise, if only one switch is used to toggle long-press behaviors, it will overlap with the expected behaviors in the short press.

In this solution, switch variables *startBlink* and *stopBlink* were used. Tables 1-2, summarize how the states of these switches affect the LED state.

TABLE 1
THE INFLUENCE OF THE SHORT AND LONG BUTTON PRESSES
TO THE *STARTBLINK*, *STOPBLINK*, AND THE LED STATE VARIABLES

PREVIOUS LED STATE	**PRESS DURATION**	**BOOLEAN VALUE**		**LED STATE**
		startBlink	*stopBlink*	
LOW	Short	false (default)	true	HIGH
HIGH		false (default)	true	LOW
BLINK OFF	Long	true	false	BLINK
BLINK ON		true	true	HIGH

TABLE 2
THE TRUTH TABLE DESCRIBING
THE BLINKING BEHAVIOR OF THE LED

startBlink	***stopBlink***	**BLINK LED**
false	false	NO
false	true	NO
true	false	YES
true	true	NO

After all the issues were addressed; extensive tests were conducted. Multiple adjustments were made to the code specifically in terms of restructuring the conditional statements. Moreover, print flags were identified to help in troubleshooting.

Ultimately, the code was finalized with the following serial monitor output that corresponds to the behavior of the hardware (circuit).

```
11:10:21.698 -> A SHORT press is detected:      1
11:10:22.401 -> A SHORT press is detected:      0
11:10:22.969 -> A SHORT press is detected:      1
11:10:23.634 -> A SHORT press is detected:      0
11:10:29.008 -> A LONG press is detected:        <blink on>
11:10:29.008 -> HIGH     1
11:10:29.995 -> LOW      0
11:10:30.979 -> A LONG press is detected:        <blink on>
11:10:31.026 -> HIGH     1
11:10:31.977 -> LOW      0
11:10:32.972 -> HIGH     1
11:10:33.971 -> LOW      0
11:10:35.009 -> HIGH     1
11:10:35.994 -> LOW      0
11:10:37.836 -> A SHORT press is detected:      <blink off>     1
11:10:39.008 -> A SHORT press is detected:      0
11:10:40.227 -> A SHORT press is detected:      1
11:10:41.212 -> A SHORT press is detected:      0
11:10:42.572 -> A SHORT press is detected:      1
11:10:43.557 -> A SHORT press is detected:      0
11:10:44.401 -> A SHORT press is detected:      1
11:10:48.337 -> A SHORT press is detected:      0
11:10:53.774 -> A LONG press is detected:        <blink on>
11:10:53.822 -> HIGH     1
11:10:54.759 -> LOW      0
11:10:55.791 -> A LONG press is detected:        <blink on>
11:10:55.791 -> HIGH     1
11:10:56.776 -> LOW      0
11:10:57.751 -> HIGH     1
11:10:58.797 -> LOW      0
11:10:59.774 -> HIGH     1
11:11:00.758 -> LOW      0
11:11:03.050 -> A SHORT press is detected:      <blink off>     1
11:11:04.097 -> A SHORT press is detected:      0
11:11:04.790 -> A SHORT press is detected:      1
```

Fig. 11. The serial monitor output of the programmed solution

Notably, two long presses are detected in succession despite a single press instance. Based on the timestamps, this behavior has a negligible effect on the solution. Nevertheless, strenuous troubleshooting was performed. Unfortunately, the issue could not be traced and currently, is dismissed as a hardware limitation.

III. Conclusion

To conclude, the pushbutton system was successfully implemented using Arduino. The circuit was able to identify the short presses on the button with precision. The LED pin turns on and off with each successive brief button press. Although, significant difficulties were discovered while testing and debugging, particularly when the LED was in blinking mode. Accordingly, workarounds were determined to address these problems. Specifically, (a) by performing a long press instead of a short press, the circuit was able to correctly detect the input and toggle between blinking mode and static HIGH; and (b) by identifying the optimal set of switch variables, the problematic overlapping of press functions was avoided. To note, it was also observed that during a single long press of the button, the Arduino serial monitor abnormally detects two long-presses in succession. Fortunately, this issue poses a negligible impact on the function of the system. It was determined to be a hardware limitation because the cause could not be determined despite intensive troubleshooting.

For the benefit of the project, it is recommended that additional study and research be undertaken on Arduino's hardware components—particularly on their specifications and limitations. Consequently, future developers should then be able to accurately assess their system designs.

V. References

ArduinoGetStarted.com, “Arduino - button - long-press short press: Arduino tutorial,” Arduino Getting Started, 2021. [Online]. Available: https://arduinogetstarted.com/tutorials/arduino-button-long-press-short-press. [Accessed: 29-Apr-2022].

ArduinoGetStarted.com, “Arduino - button: Arduino tutorial,” *Arduino Getting Started*, 2022. [Online]. Available: https://arduinogetstarted.com/tutorials/arduino-button. [Accessed: 05-May-2022].

Colson2, “(solved) Push Button to blink led,” Arduino Forum, 28-May-2020. [Online]. Available: https://forum.arduino.cc/t/solved-push-button-to-blink-led/657628. [Accessed: 29-Apr-2022].

J. Ganssle, “A guide to debouncing,” *The Ganssle Group*, 2004. [Online]. Available: https://my.eng.utah.edu/~cs5780/debouncing.pdf. [Accessed: 08-May-2022].

P. Horowitz and W. Hill, “The Arts of Electronics - 2nd Edition,” *Press Syndicate of the University of Cambridge*, 1989. [Online]. Available: https://www.pearl-hifi.com/06_Lit_Archive/02_PEARL_Arch/Vol_16/Sec_51/4420_The_Art_of_Electronics.pdf. [Accessed: 2022].

Reverendfuzzy, “Simple on/off pushbutton,” Arduino Project Hub, 2019. [Online]. Available: https://create.arduino.cc/projecthub/reverendfuzzy/simple-on-off-pushbutton-f637a7. [Accessed: 29-Apr-2022].

Roy B, “Blinking led with Arduino Uno,” Arduino Project Hub, 2020. [Online]. Available: https://create.arduino.cc/projecthub/RoyB/blinking-led-with-arduino-uno-01e098. [Accessed: 29-Apr-2022].

T. A. Team, “How to wire and program a button: Arduino documentation,” Arduino Documentation | Arduino Documentation, 2022. [Online]. Available: https://docs.arduino.cc/built-in-examples/digital/Button. [Accessed: 29-Apr-2022].

Appendices