American International University- Bangladesh (AIUB)
Faculty of Engineering (EEE)

Course Name :	MICROPROCESSOR AND EMBEDDED SYSTEMS	**Course Code :**	4103
Semester :	Summer 2023	**Sec :**	F
Lab Instructor :	Md Sajid Hossain		

Experiment No :	04
Experiment Name :	Part-3:Software Debouncing Implementation for Pushbutton Switches: Eliminating Noise and Achieving Reliable Input Detection

Submitted by (NAME):	**Rifat Hossain**	**Student ID:**	**20-42461-1**

Group Members	**ID**	**Name**
1.	20-42461-1	Rifat Hossain
2.	20-42831-1	Ishrat Jahan
3.	21-45019-2	Ahnaf Abdullah Zayad
4.	21-45038-2	Srabone Raxit
5.	21-45206-2	Kazi Ramisa Samiha
6.	21-45263-2	Shakibul Hasan

Performance Date :	**20.06.2023**	**Due Date :**	**04.07.2023**

Marking Rubrics (to be filled by Lab Instructor)

Category	Proficient [6]	Good [4]	Acceptable [2]	Unacceptable [1]	Secured Marks
Theoretical Background, Methods & procedures sections	All information, measures and variables are provided and explained.	All Information provided that is sufficient, but more explanation is needed.	Most information correct, but some information may be missing or inaccurate.	Much information missing and/or inaccurate.	
Results	All of the criteria are met; results are described clearly and accurately;	Most criteria are met, but there may be some lack of clarity and/or incorrect information.	Experimental results don't match exactly with the theoretical values and/or analysis is unclear.	Experimental results are missing or incorrect;	
Discussion	Demonstrates thorough and sophisticated understanding. Conclusions drawn are appropriate for analyses;	Hypotheses are clearly stated, but some concluding statements not supported by data or data not well integrated.	Some hypotheses missing or misstated; conclusions not supported by data.	Conclusions don't match hypotheses, not supported by data; no integration of data from different sources.	
General formatting	Title page, placement of figures and figure captions, and other formatting issues all correct.	Minor errors in formatting.	Major errors and/or missing information.	Not proper style in text.	
Writing & organization	Writing is strong and easy to understand; ideas are fully elaborated and connected; effective transitions between sentences; no typographic, spelling, or grammatical errors.	Writing is clear and easy to understand; ideas are connected; effective transitions between sentences; minor typographic, spelling, or grammatical errors.	Most of the required criteria are met, but some lack of clarity, typographic, spelling, or grammatical errors are present.	Very unclear, many errors.	
Comments:				Total Marks (Out of):	

Software Debouncing Implementation for Pushbutton Switches: Eliminating Noise and Achieving Reliable Input Detection

I. Abstract: This lab report describes the software debouncing techniques implemented for pushbutton switches. By implementing a debouncing algorithm in software, the objective of this experiment is to eliminate noise and accomplish dependable input detection. Arduino is being used to demonstrate the debouncing procedure. Connecting a pushbutton switch and an LED to the Arduino board and using the provided code to debounce the switch input constitutes the experiment. The LED's state is then determined by the debounced trigger input. The results demonstrate that software debouncing effectively eliminates erratic measurements and provides precise button state detection. This experiment demonstrates the significance of debouncing techniques for improving the reliability as well as the effectiveness of input devices in electronic systems.

II. Introduction: This experiment employs an Arduino board to implement software debouncing techniques for pushbutton switches. Contact bounce, a prevalent issue with mechanical switches, causes inconsistent and unreliable button presses. This issue is resolved by software debouncing, which filters out noise and fluctuations to provide accurate and stable input detection.

Connecting a pushbutton switch and an LED to the Arduino board initiates the experiment. The code provided implements a debounce algorithm that reads the switch state and applies a delay to ensure stable readings. If the switch state changes and remains stable during the debounce delay, the button press is regarded valid. The state of the LED is then determined by the debounced switch input. The theory section describes contact bounce and the significance of debouncing techniques for achieving reliable input detection. The methodology section describes the steps necessary to connect the components and programme the Arduino board in order to implement software debouncing. The experiment demonstrates the efficacy of software debouncing in removing contact deflection and providing precise switch input detection.

This experiment demonstrates the importance of debouncing techniques in enhancing the dependability and efficacy of input devices. Software debouncing, when implemented on the Arduino platform, provides a practical and accessible means of reducing contact rebound. The results of this experiment can be implemented in numerous electronic systems requiring precise and dependable switch inputs.

The implementation of software debouncing on the Arduino board provides insight into the practical application of debouncing techniques. This experiment contributes to a greater comprehension of reliable input detection and lays the groundwork for future exploration and optimisation of switch debouncing in electronic projects and systems.

III. Theory and Methodology: Contact bounce is a typical occurrence in mechanical switches, such as pushbutton switches. When a switch is depressed or released, the mechanical movement of the contacts causes rapid on-off transitions, resulting in unreliable and unstable readings. In applications requiring accurate and dependable input detection, this contact vibration can pose a problem.

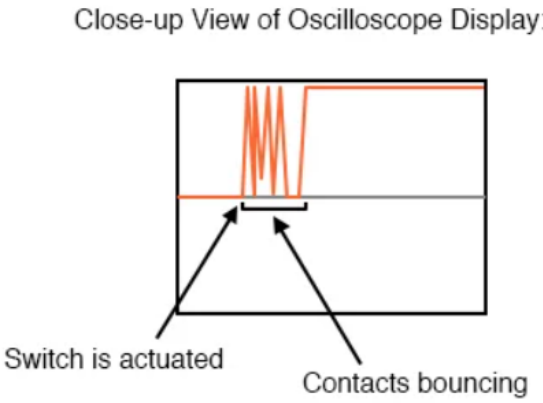

Figure 3.1: Close up view of Oscilloscope.

Debouncing is the process of filtering out contact bounce-induced noise and fluctuations to ensure stable and accurate input detection. Both hardware and software approaches can be used to implement debouncing techniques. This experiment focuses on software debouncing, which employs software algorithms to eradicate contact bounce and provide reliable switch input detection.

Typically, the software debouncing algorithm involves introducing a delay and analysing the input's stability over time. Before deeming a switch state change to be a valid button press, the algorithm waits for a specified debounce delay after a switch state change to ensure that the state has stabilised. By incorporating this delay and stability test, the software debouncing algorithm filters out unwanted noise and provides a trustworthy representation of the button's state.

IV. Apparatus:

1. Arduino Uno R3 Board (ATmega328P)
2. Arduino IDE
3. Resistors (200 ohms, 1) and (10 k ohms, 1)
4. LED Lights (Red)
5. Connecting Wires.
6. Breadboard
7. USB Cable
8. Tinkercad Platform for simulation.

V. Precautions:

1. The computers' connection with the Arduino board was made sure.
2. The ports' connection on the board that matched with the code implementation were checked carefully.
3. Ensure that the Arduino board and its components are not connected to any power source while making connections or modifications to the circuit. This prevents accidental short circuits and component damage.
4. Connect the Arduino board's ground (GND) pin to the ground of the external power supply or the ground reference of the circuit.
5. Handle the components with care and refrain from using excessive force or roughness, particularly when inserting or removing jumper wires. Mishandling can cause injury to the components or loose connections, affecting the experiment's reliability.
6. Before uploading the code to the Arduino board, it should be inspected for errors and mistakes. Incorrect code can lead to unanticipated results or behaviour. If practicable, consult a mentor or coworker to confirm the code's accuracy.

VI. Experimental Procedure:

1. Gather all the required components listed in the Apparatus section.

2. Set up the Arduino board and connect it to the computer using a USB cable. Ensure that the Arduino IDE software is installed on the computer.

3. Open the Arduino IDE and create a new sketch.

4. Connect the pushbutton switch to the Arduino board:

    - Connect one terminal of the switch to a digital input pin (buttonPin) on the Arduino board.
    - Connect the other terminal of the switch to the ground (GND) pin on the Arduino board.
5. Connect the LED to the Arduino board:

    - Connect the longer leg (anode) of the LED to a current-limiting resistor, and then connect the other end of the resistor to a digital output pin (ledPin) on the Arduino board.
    - Connect the shorter leg (cathode) of the LED directly to the ground (GND) pin on the Arduino board.
6. Double-check all the wiring connections to ensure they are secure and correctly made according to the circuit diagram and pin assignments.

7. Upload the provided software debouncing code to the Arduino board:

    - Verify that the code does not contain any errors or typos.
    - Select the correct Arduino board model and port from the Tools menu in the Arduino IDE.
    - Click on the "Upload" button to compile and upload the code to the Arduino board.
8. Observe the behaviour of the LED as you press and release the pushbutton switch. Take note of any changes in the LED state.

9. Repeat the button press and release multiple times to test the effectiveness of the software debouncing in eliminating contact bounce and providing stable input detection.

10. Record the observations and note any notable behaviour or patterns observed during the experiment.

11. Repeat the experiment with variations in the debounce delay to observe the effects on the switch input detection.

12. Once the experiment is complete, power off the Arduino board and disconnect it from the power source.

13. Analyse and interpret the results, comparing the behaviour of the LED with and without the software debouncing technique. Discuss the effectiveness of the software debouncing algorithm in mitigating contact bounce and providing stable input detection.

14. Document your findings, observations, and conclusions in the lab report.

Code for Traffic System Control:

```
//part 3

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin = 13; // the number of the LED pin
// Variables will change:
int ledState = HIGH; // the current state of the output pin
int buttonState; // the current reading from the input pin
int lastButtonState = LOW; // the previous reading from the input pin
// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
unsigned long lastDebounceTime = 0; // the last time the output pin was toggled
unsigned long debounceDelay = 1; // the debounce time; increase if the o/p flickers
void setup() {
pinMode(buttonPin, INPUT);
pinMode(ledPin, OUTPUT);
// set initial LED state
digitalWrite(ledPin, ledState);
}
void loop() {
// read the state of the switch into a local variable:
int reading = digitalRead(buttonPin);
// check to see if you just pressed the button
// (i.e. the input went from LOW to HIGH), and you've waited long enough
// since the last press to ignore any noise:
// If the switch changed, due to noise or pressing:
if (reading != lastButtonState) {
// reset the debouncing timer
lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
// whatever the reading is at, it's been there for longer than the debounce
// delay, so take it as the actual current state:
// if the button state has changed:
if (reading != buttonState) {
buttonState = reading;
// only toggle the LED if the new button state is HIGH
if (buttonState == HIGH) {
ledState = !ledState;
}
}
}
// set the LED:
digitalWrite(ledPin, lastButtonState); //button state low
// save the reading. Next time through the loop, it'll be the lastButtonState:
lastButtonState = reading;
}


```

VII. Simulation and Measurement:

Simulation was done after implementation of the hardware part to match with the result.

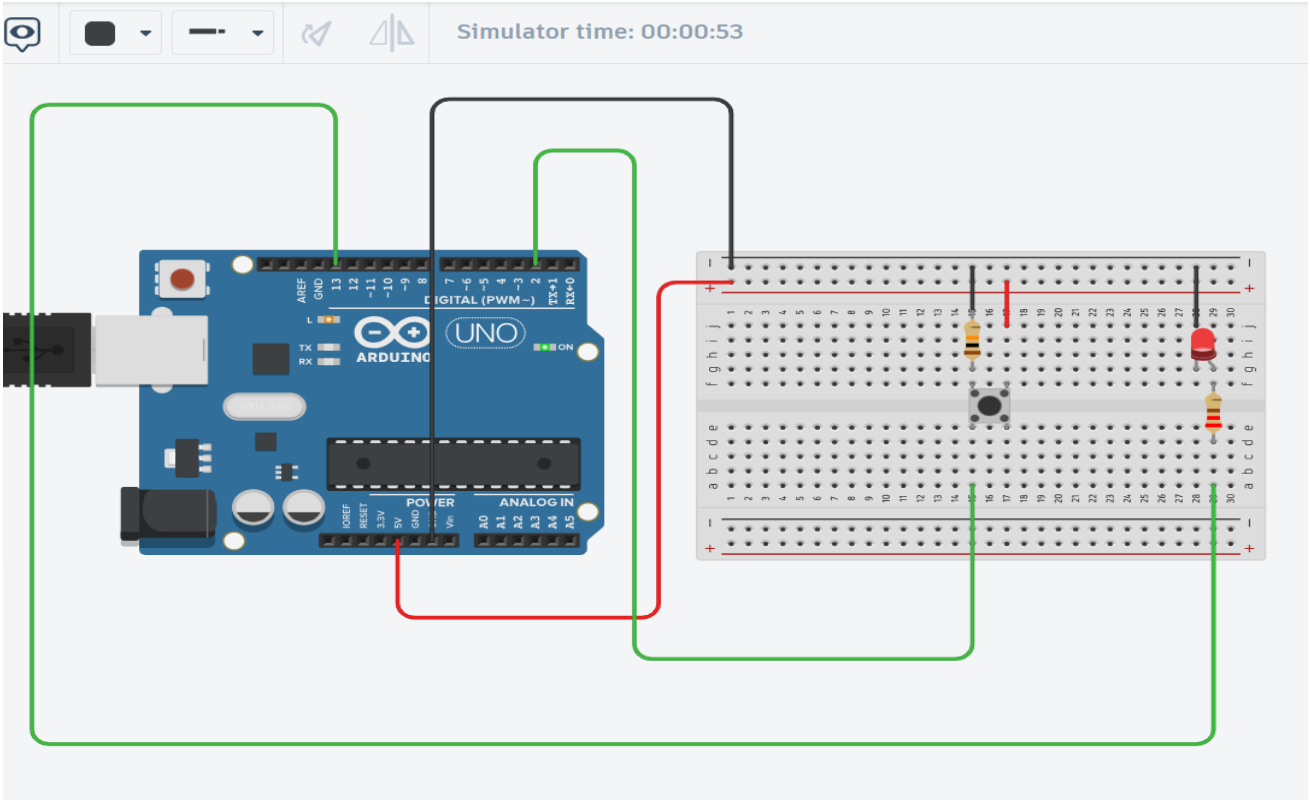

Figure 7.1: Button pressed (LED On).

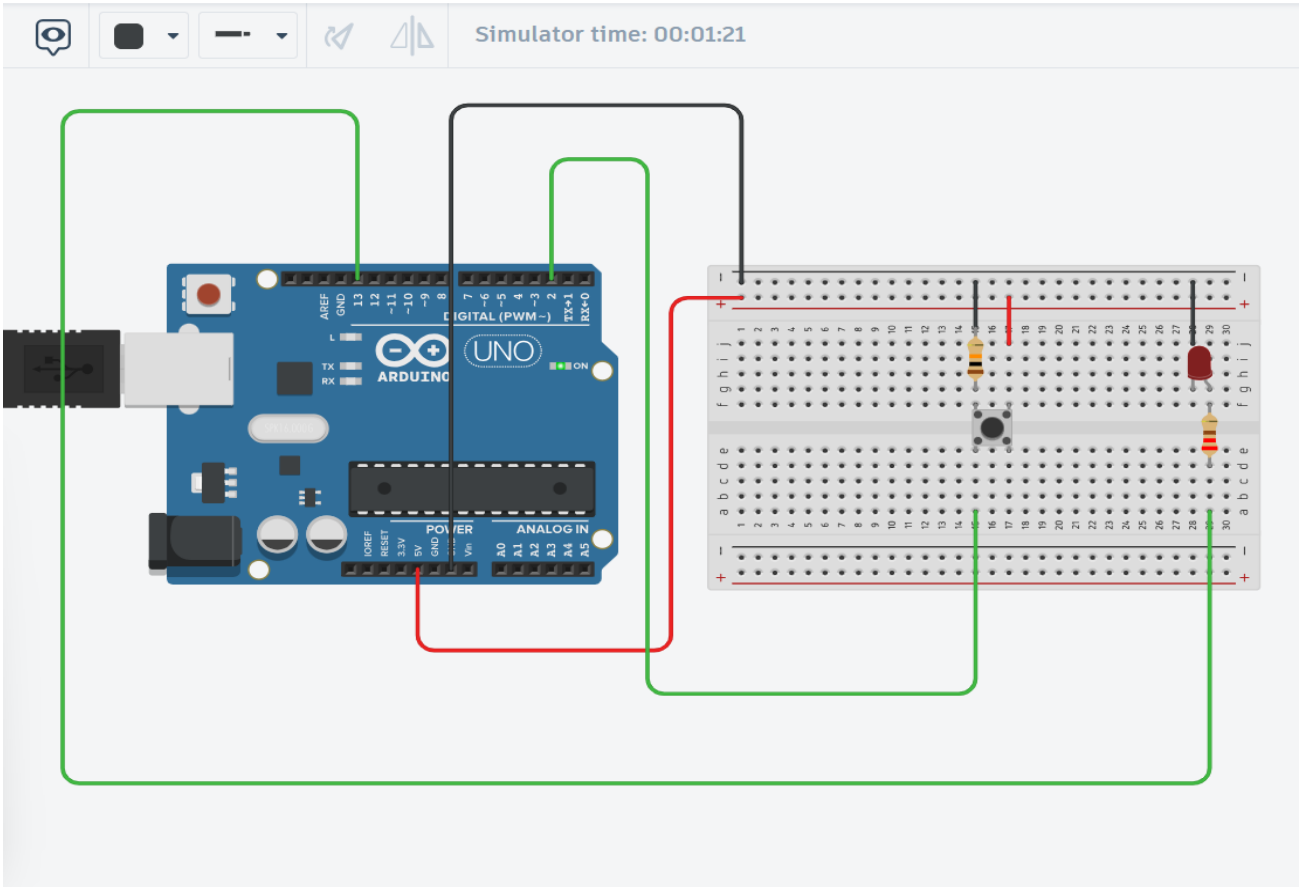

Figure 7.1: Button is not being pressed (LED Off).

VIII. Results:

The Algorithm was uploaded to the Arduino.

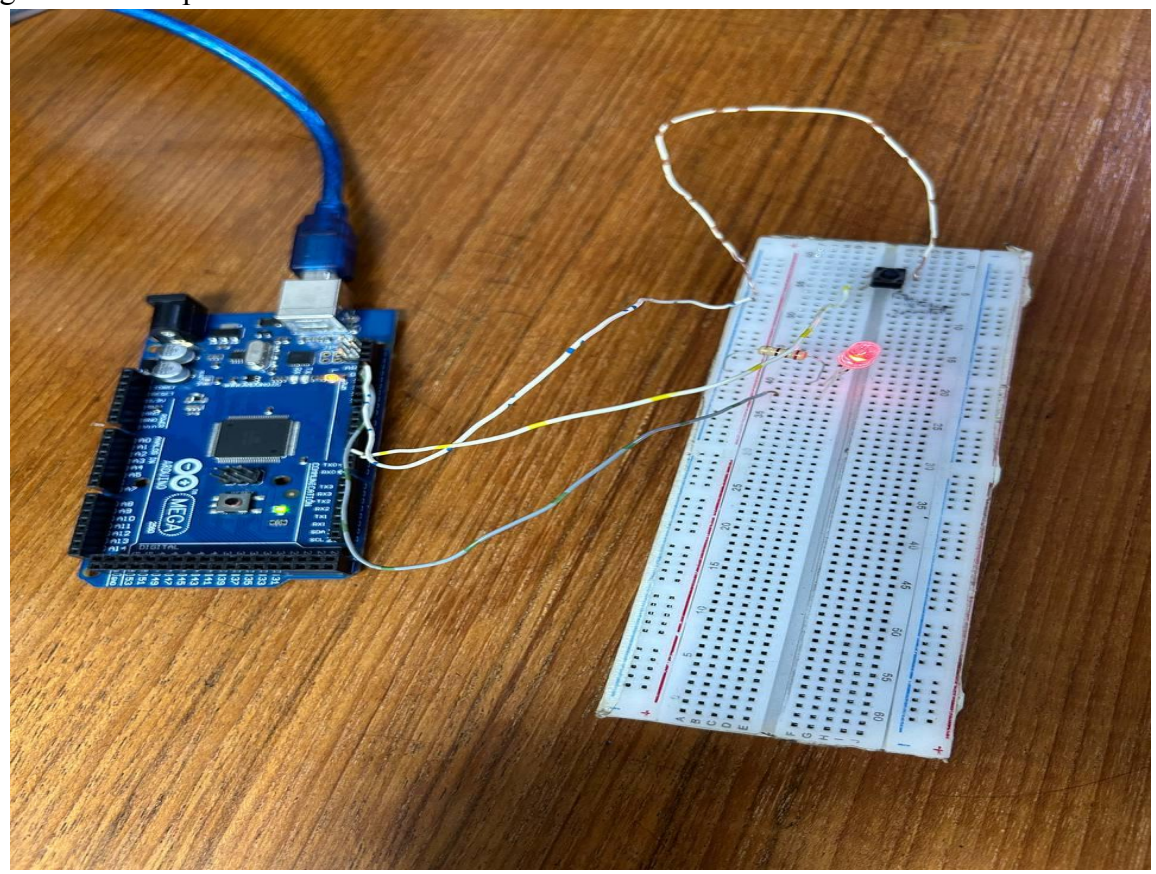

Figure 8.1: LED On (no debouncing effect)

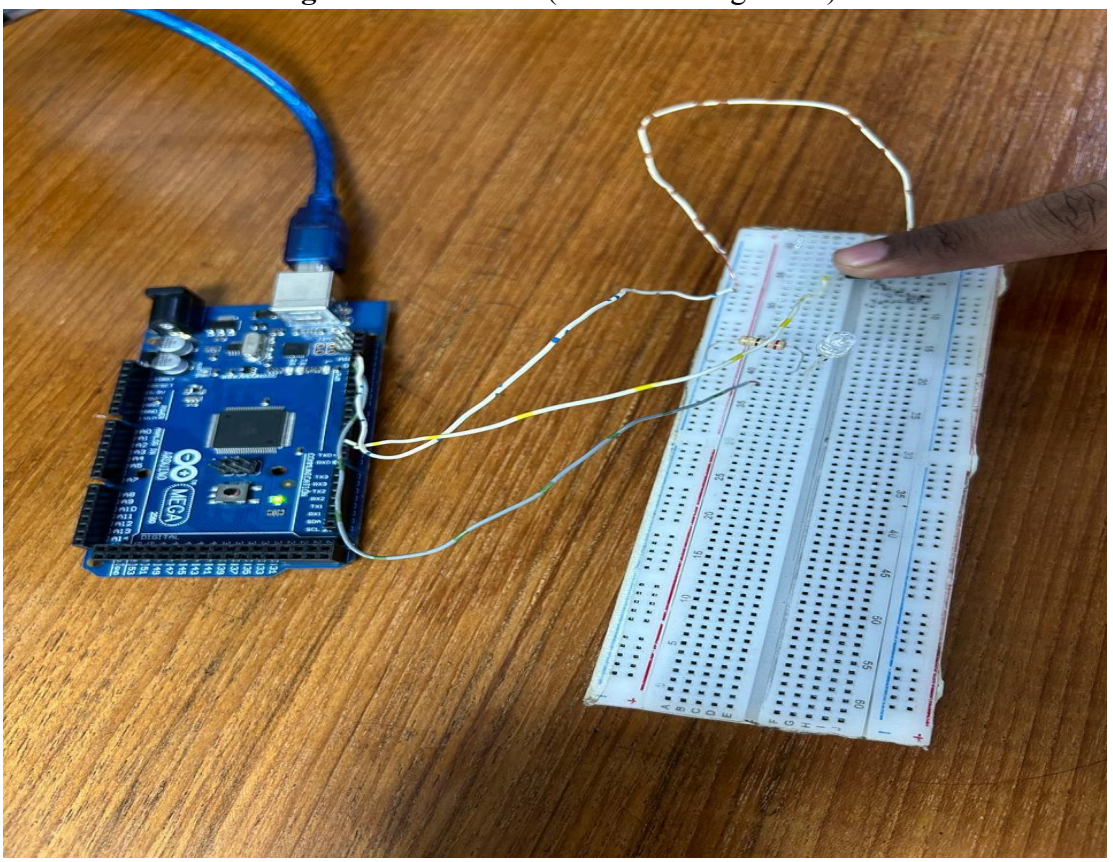

Figure 8.2: LED Off

IX. Discussion:

The purpose of the experiment was to implement software debouncing techniques using an Arduino board and evaluate their efficacy in mitigating contact bounce and providing reliable input detection. The results demonstrated that the software debouncing algorithm eradicated contact bounce, as evidenced by the stable state of an LED that reflected the switch input. Variations in the debounce delay illustrated the technique's adaptability. Software debounce offers a cost-effective solution that is adaptable to various switch characteristics and debounce needs. Although it causes a minor delay in detecting button presses, the benefits of accurate and consistent input detection outweigh this drawback. Future experiments could investigate various switch types, the efficacy in noisy environments, alternative algorithms, and optimised debounce delays. Overall, the experiment aided in the comprehension of debouncing techniques and their application to improving the dependability of input devices in electronic systems.

X. Conclusion:

In conclusion, the experiment utilised an Arduino board to effectively implement software debouncing techniques. The results demonstrated the efficacy of the debouncing algorithm in removing contact noise and providing dependable input detection. Software debouncing has proven to be a cost-effective and adaptable means of reducing transition noise. Despite introducing a slight delay in button detection, the advantages of accurate and stable input detection outweighed this limitation. Overall, the experiment confirmed that software debouncing is a practical method for enhancing the reliability and performance of input devices in electronic systems.

XI. Reference (s):

[1] "Contact 'Bounce' | Switches | Electronics Textbook," All About Circuits, Available: https://www.allaboutcircuits.com/textbook/digital/chpt-4/contact-bounce/. [Accessed: Jul. 3, 2023].

[2] ATmega328 Arduino Uno Board Working and Its Applications-Elprocus. Available at: https://www.elprocus.com/atmega328-arduino-uno-board-working-and-itsapplications/?fbclid=IwAR31Al9NcljIcVp0TRvwr7XA7Pu6Vq9yaaJXyTs7kpuPHvqNjOXkS2dFUAk [Accessed: Jul. 3, 2023].