Activity No. 3	
Digital I/O Operations	
Course Code: CPE006	**Program:** CPE
Course Title: Microprocessor Systems	**Date Performed:** 10/2/23
Section: 41S1	**Date Submitted:**
Name/s: Bencito, Sonny Jay Bote, Briant Jerome Fetalino, Mico Sevilla, Mylen Valenzuela, Robin	**Instructor:** Engr Cris Paulo Hate

1. Objective:

This activity aims to demonstrate the different digital I/O operations in a microcontroller based system. Included in the activity is the programming of a source code for processing digital data.

2. Intended Learning Outcomes (ILOs):

After completion of this activity the students should be able to:
2.1 Write a code for a microprocessor based system that implements digital input devices.
2.2 Design a circuit that accepts digital transducer data, processes and then displays digital output.

3. Discussion :

Guide Questions:

1. What is digital I/O, and how does it work in the context of Arduino?

    - Arduino's digital I/O is vital for connecting sensors, actuators, and ICs. It enables tasks like reading switches, lighting indicators, and controlling relays. Digital signals only come in HIGH (1) or LOW (0) values. The important Arduino methods "pinMode()", "digitalRead()", and "digitalWrite()" read input pins and return HIGH or LOW, respectively, and write values to pins as HIGH or LOW, respectively. To use digital I/O, connect devices to Arduino's digital pins and employ these functions for signal control.

2. Explain the fundamental difference between digital input and digital output pins on an Arduino board.

    - An Arduino's digital input pins may read digital signals (HIGH or LOW), which are frequently used for switches and buttons. By setting them to HIGH or LOW, digital output pins write digital signals that are used to operate devices like LEDs.

3. How does digital I/O in Arduino compare to analog I/O, and when is it more suitable for a project?

    - Digital I/O manages binary signals (HIGH or LOW), serving purposes like button state detection and LED manipulation through functions such as pinMode(), digitalRead(), and digitalWrite(). It might operate at a slower pace but proves cost-efficient. In contrast,

analog I/O deals with continuous signals, utilizing analogRead() and analogWrite(), aided by an embedded analog-to-digital converter (ADC) for signal conversion. This option provides superior precision and granularity, making it suitable for tasks like monitoring temperature or light sensors, albeit potentially incurring higher costs and complexity. Digital I/O suits straightforward on/off control or projects with budget constraints, while analog I/O excels in scenarios requiring meticulous signal processing without cost limitations.

4. How are digital signals represented in Arduino, and what are the commonly used voltage levels for HIGH and LOW states?

    - It determines the voltage levels of the HIGH and LOW digital signals used by Arduino. When a pin's voltage is between 3V and 5.5V on a 5V board, it is considered HIGH, and when it is below 1.5V, it is considered LOW. Both the LOW and HIGH ranges for input pins are from 0V to 0.99Vcc (0.3V) respectively. Unpredictable input voltages occur in the range of 1V and 1.8V. A datasheet should be consulted for exact information because voltage levels can fluctuate depending on the board's supply voltage and other variables.

5. Discuss the advantages of using digital I/O for tasks like controlling LEDs, reading switches or sensors, and interfacing with other digital devices.

    - There are various benefits to using digital I/O for activities including LED control, switch and sensor reading, and interacting with digital devices. With routines like pinMode(), digitalRead(), and digitalWrite(), digital pins on an Arduino are simple to control and may be used as inputs or outputs. For simple jobs like controlling LEDs, it proves to be cost-effective. It also offers reliable current source capabilities for driving devices and dependability due to its lower sensitivity to noise and interference compared to analog signals. While analog I/O is better suitable for complex operations demanding accuracy and detail, digital I/O is a simple and reliable option for controlling devices like LEDs and switches and interacting with digital devices.

4. Directions:

Part 1: Toggle Button

1. Configure your circuit to accept one(1) input from a single switch and one(1) LED bulb.
2. Use the button to toggle the state of the LED bulb.
3. Test your system and document your work.

Part 2: Push Counter

1. Configure your circuit to accept one(1) input from a single switch and eight(8) LED bulbs.
2. The device must detect and count button presses.
3. The number of button presses must be reflected in the LED array as an 8-bit binary number.
4. Test your system and document your work.

Part 3: Push controller

1. Retain your original circuit but add a second button.
2. Configure your code to have a single LED to be ON at position 4 of the LED array.
3. Configure the two buttons to behave as LEFT and RIGHT controls.
4. When the LEFT button is pushed, the Active LED bulb 'moves' to the left. This should also be the condition for the RIGHT button.
5. When the Active LED reaches the edge of the LED array, it will loop to the other edge.0
6. Test your system and document your work.

5. Resources:

The activity will require the following software, tools and equipment:

Software:

- Arduino 1.8.19

Tools:

- Arduino Uno (1)
- Breadboard (1)
- Set wires (1)
- Push button (2)
- LED (8)
- Resistor 100 Ω (8)
- Resistor 1kΩ (2)

6. Methodology

**Document EVERYTHING you did to accomplish this(think of a user manual).*
**A step by step procedure would be helpful.*
**Include schematic diagrams and source code for this activity.*
*****Flowcharts*
For Part 1: Toggle Button:
Step 1: *Get a borrowed slip and fill it up.*
Step 2: *Get the materials needed.*
Step 3: *Arrange the arduino based on tasks that need to be done.*
Step 4: *Write a program that is applicable to the task.*

- *It should be able to configure the circuit to accept one(1) input from a single switch and one(1) LED bulb.*
- *Use the button to toggle the state of the LED bulb.*

```
int x = 0;
int buttonState = LOW;
int lastButtonState = LOW;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
 pinMode(8, INPUT_PULLUP); // Use INPUT_PULLUP to enable the internal pull-up resistor
 Serial.begin(9600);
}

void loop() {
 int reading = digitalRead(8);

 if (reading != lastButtonState) {
 lastDebounceTime = millis();
 }

 if ((millis() - lastDebounceTime) > debounceDelay) {
 if (reading != buttonState) {
 buttonState = reading;

 if (buttonState == HIGH) {
 x = 1 - x; // Toggle the value between 0 and 1
 digitalWrite(8, x);
 }
 }
 }

 lastButtonState = reading;
 Serial.println(x);
 delay(150);
}
```

Figure 1: Toggle Button Source Code

Step 5: *Run the program*
Step 6: *Fix the program if there is any error or check the arduino if the arrangement is correct.*
Step 7: *Document the task.*

For Part 2: Push Counter

Step 1: *Rearrange the connection in the arduino that is applicable to Push counter.*
Step 2: *Write a program that for Push counter*

- *The circuit should be able to accept one(1) input from a single switch and eight(8) LED bulbs.*
- *It should be able to detect and count button presses.*
- *The number of button presses must be reflected in the LED array as an 8-bit binary number.*

```
int buttonPin = 5;
int leds[] = {13, 12, 11, 10, 9, 8, 7, 6};
int count = 0;
int button1 = LOW;
int button2 = LOW;
void setup() {
 pinMode(buttonPin, INPUT_PULLUP);
 for (int i = 0; i < 8; i++) {
 pinMode(leds[i], OUTPUT);
 digitalWrite(leds[i], LOW);
 }
 Serial.begin(9600);
}
void loop() {
 button1 = digitalRead(buttonPin);
 if (buttonState == LOW && button2 == HIGH) {
 count++;
 if (count > 255) {
 count = 0;
 }
 updateLEDs(count);
 }
 lastButtonState = buttonState;
}
void updateLEDs(int value) {
 for (int i = 0; i < 8; i++) {
 int bit = (value >> i) & 1;
 digitalWrite(leds[i], bit);
 }
}
```

Figure 2: **Push Counter** ***Source Code***

Step 3: *Run the created program*

Step 4: *Fix the program if there is any error or check the arduino if the arrangement is correct.*

Step 5: *Document the task.*

For Part 3: Push controller:

Step 1: *Used again the previous Arduino Uno circuit that the student arranged and added another push button.*

Step 2: *Write a program that for Push counter*

- The code should be able to have a single LED to be ON at position 4 of the LED array.
- The two buttons should be able to behave as LEFT and RIGHT controls.
- When the LEFT button is pushed, the Active LED bulb 'moves' to the left. This should also be the condition for the RIGHT button.
- When the Active LED reaches the edge of the LED array, it will loop to the other edge.0

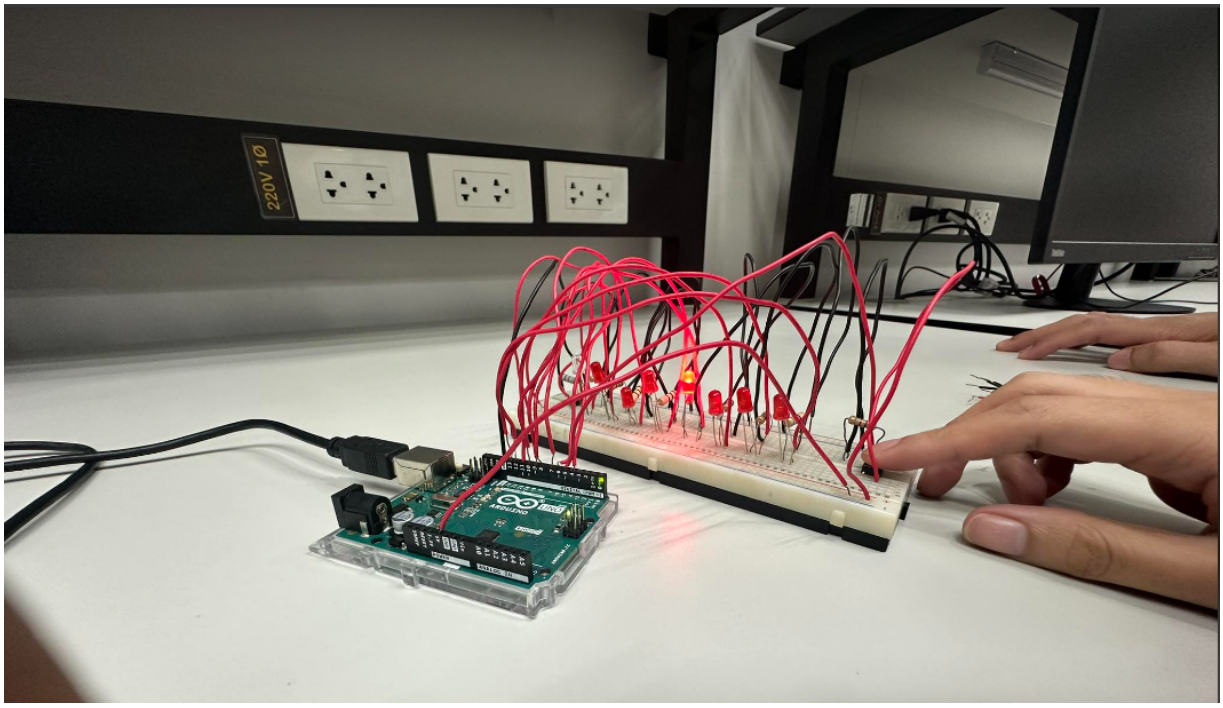


Figure 3: **Push Controller *Source Code***

Step 3: *Run the created program*

Step 4: *Fix the program if there is any error or check the arduino if the arrangement is correct.*

Step 5: *Document the task.*

7. Results

**Pictures of the circuit, (link of the) video of the running system, graphs, plots, and other elements are useful here.*

**Table format and graph format if you have data results.*

**Do not forget to add captions if you are using images.*

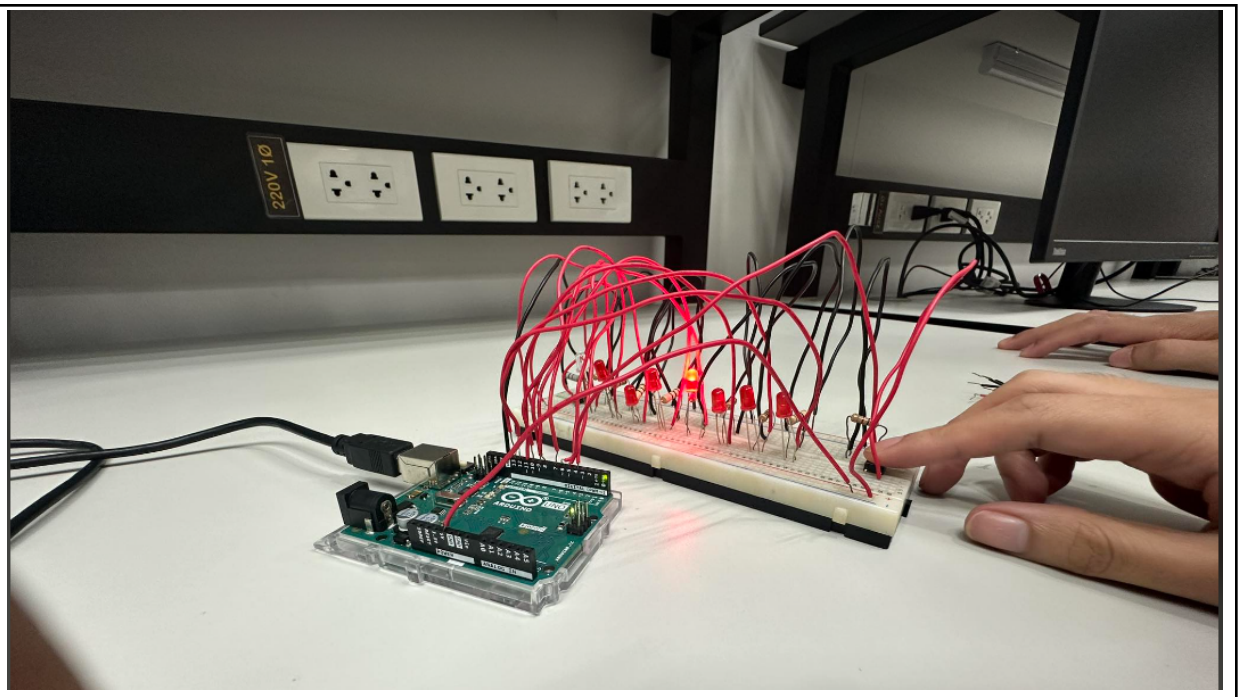


Figure 1.1: Toggle Button Output Analysis

Button	Output
Toggle	1
	0

Figure 1.2: Toggle Button Output Table

<Insert image here>

Figure 2.1: Push Counter Output Analysis

Button	Binary	LED							
		128	64	32	16	8	4	2	1
1x	0000001	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
1x	11111111	ON	ON	ON	ON	ON	ON	ON	ON

Figure 2.2: Push Counter Output Table

<Insert image here>

Figure 3.1: Push Controller Output Analysis

Button	LED							
	128	64	32	16	8	4	2	1
Default	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
Right	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
Left	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Figure 3.2: Push Controller Output Table

Video Link:
https://drive.google.com/drive/folders/1NkjjF8h5n1lXDHihy223VGVE2-MGKHaP?usp=sharing

8. Results Analysis

**Cite your analysis based on the observations of the results.*

Toggle Button Output Analysis

The students observed that clicking and releasing the button allowed the code to appropriately swap the value of 'x' between 0 and 1, when they looked at the program's output. The program's serial monitor displays output one (1) when you press the button; if you press it again, the output changes to zero (0). This behavior demonstrates the basic functionality of a toggle switch using digital input and output in an Arduino project.

Push Counter Output Analysis

A Push Counter system with eight LEDs and one button is controlled by the provided code. Every time the button is pressed, a counter is advanced and the matching binary representation is shown on the LED array. When the count hits 255, it smoothly resets to 00000001, creating a loop in which binary numbers are continuously counted from 00000001 to 11111111 and then back to 00000001. This Push Counter is used to demonstrate binary counting concepts in a visual and interactive way.

Push Controller Output Analysis

When the student examines the output in the table, they see that it manages a Push Controller with 8 LEDs and 2 buttons. To move left and right, the buttons are used, and logic is included into the code to detect button pushes. A similar indication that the left button cannot move any further to the left is when it is fully depressed. The right button indicates that it cannot go any further to the right when it is fully pressed.

9. Summary and Conclusions

To sum it up, this activity was created to give students a hands-on understanding of digital I/O operations in microcontroller-based systems, with a focus on subjects like toggle switches, binary counting, and button-controlled LED movement. Students learned how to use Arduino to construct useful circuits, write code for microprocessor-based devices, and understand the foundations of digital input and output through this task.

Students demonstrated the concepts of digital input and output in the Toggle Button segment by simulating a toggle switch situation where pressing the button switched the output between 1 and 0. They set up a system with eight LEDs and a button to count in binary as part of the Push Counter segment, providing a convincing illustration of how digital input and output may be used to build a binary counting display. Last but not least, students added a second button for left and right control to the Push Controller component, enlarging the circuit and demonstrating the usefulness of digital input devices for interactive LED movement, including looping when the LEDs reached the boundaries of the array. Overall, this exercise gave the students practical skills and information that would be useful for their upcoming projects using microcontroller-based systems.

10. Learnings and Contributions of each member

**What did you do to contribute to this activity?*
**Write your personal reflection, what did you learn?*
**What did you lack?*
**What do you need to improve?*

Bencito, Sonny Jay:

- In this activity, I contribute by doing the figure 2 which has a push counter output and based on our code. I learned how to call it like in the last activity. I am also doing the debugging on the arduino.

Bote, Briant Jerome:

- I learned how to connect a push button into the arduino that is used for the binary counter when being pressed, I helped with the creation of the hardware and also for the analyzation of the codes even though i lack the ability to debug the codes properly, but I still helped in checking for the errors and correcting it.

Fetalino, Mico:

- In this laboratory activity, I learned the use of toggles in arduino, binary counter for push counter and also push controller. Also I learned how to put wire, resistor, push button, leds etc that needs for specific parts. creating code and debugging for microprocessor based systems that applied digital input devices.

Sevilla, Mylen:

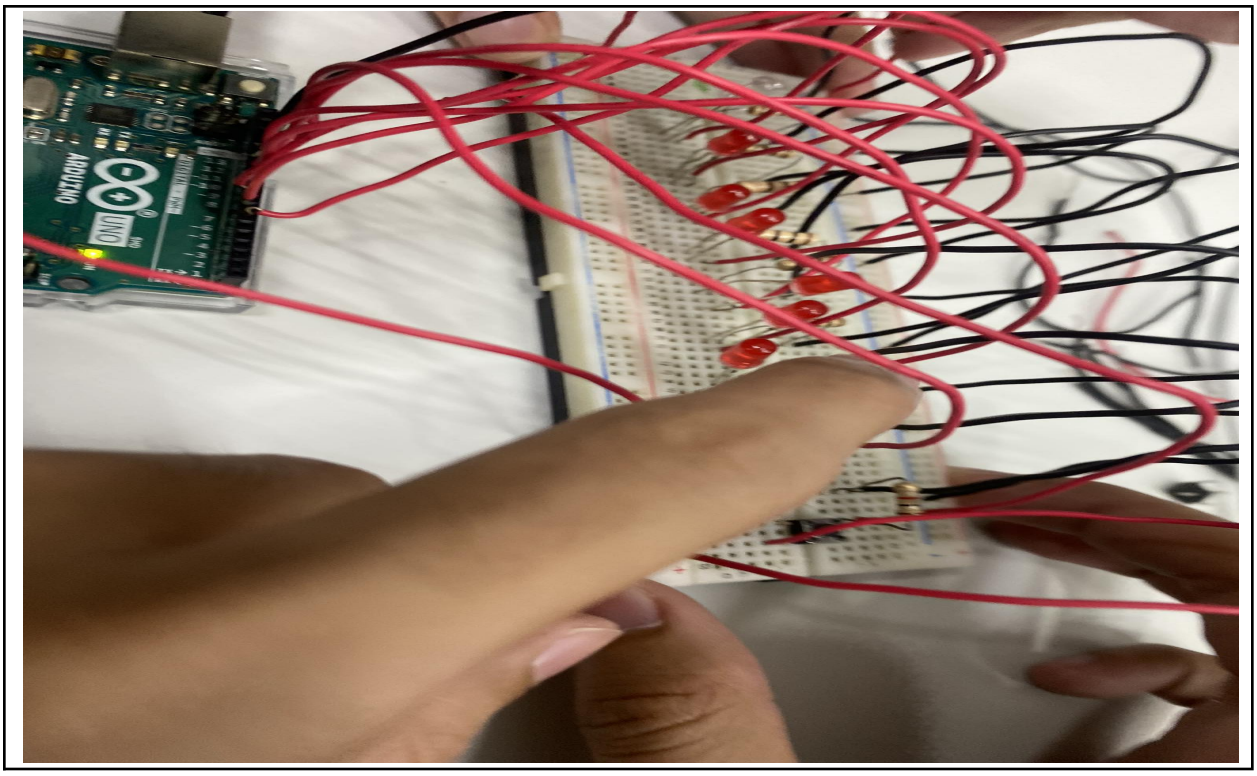
- To sum it up, this activity is all about digital I/O operation wherein the students were assigned to do the three tasks which are toggle button, push counter, and push controller. With the help of this activity, the student was able to understand and gain experience in integrating buttons with

Arduino, using them in conjunction with a binary counter to create interactive functionality. Lastly, the student was able to contribute to this activity by doing the documentation and assisting with the hardware setup.

Valenzuela, Robin:

- In this activity I learned how to use a button and incorporate it to the arduino. The binary counter that we used was used in tandem with the push button to create interactive code. I did some of the code and debugging in the activity, I also helped with the hardware. .

11. Group Picture with Activity


ARDUINO
UNO