



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

ACTIVITY REPORT

Week 2: Digital Logic Design

Interfacing a 7 segment display with Arduino. Numbering sequence from 0 to 9.

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

Table of content.....	2
Abstract.....	3
Material and equipment.....	4
Experimental setup and Methodology.....	6
Results.....	7
Discussions.....	8
Question.....	15
Recommendation.....	17
Conclusion.....	17
Student Declaration.....	18

Abstract

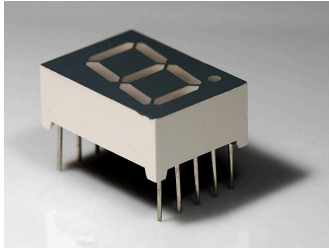
This project's main goal is to utilize an Arduino microcontroller in order to manage a one digit 7-segment display that presents numbers ranging from 0 to 9. The primary objective is to provide a simple method for users to adjust the shown number by utilizing a push-button switch. Whenever the button is pushed, the number goes up by one and goes back to 0 after reaching 9.

The project also includes methods to ensure the button presses are registered accurately, preventing any mistakes. This project merges fundamental programming such as C++, with digital logic design and electrical applied knowledge, offering a straightforward introduction to how user input can influence displays in the electronic circuits constructed .

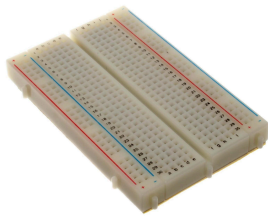
Most importantly, in this project, we succeed in applying basic principles in digital electronics, digital logic concepts while we believe that it has the potential for further development in more complex uses.

Material and equipment

1) 7-Segment Display



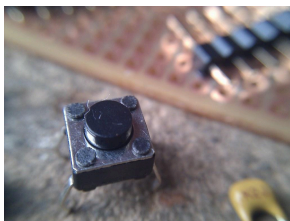
2) Breadboard



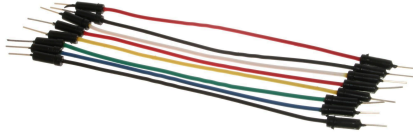
3) Arduino Board



4) Push Button



5) Wire



Experimental setup and Methodology

Firstly, the materials and components are being placed onto the breadboard, the common cathode 7-segment display is placed on the breadboard. Then the wire is connected to all of the legs of the display. One of the wires is connected to the ground on an Arduino UNO board. The remaining leg is connected to the digital pins at the Arduino board.

Next the push button is placed on the breadboard and the wire is connected to the digital pin and the other to the common ground.

Lastly, the programming coding was made on the Arduino IDE on the computer. The Arduino board is connected to the computer via USB cable to upload the coding into the Arduino board.

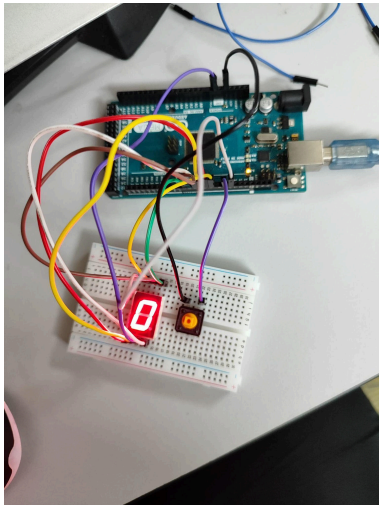


Figure 1: Full setup of the experiment.

Results

The 7-segment display correctly counted up in response to button presses, as demonstrated by the results of the experiment. This feature highlights the effective integration of the hardware configuration with the software. The experiment successfully implemented a mechanism for increasing the count using a single push button. The numerical digits, ranging from 0 to 9, were displayed sequentially on the 7-segment display. Once the count reached 9, it reset to 0 upon the next button press, allowing for continuous counting.

The system functioned reliably, registering each button press accurately, and ensuring no unintended skips or repetitions in the count. This demonstrates that such a system can be adapted for various applications that require counting or tracking, providing a simple yet effective solution for real-world uses.

Discussions

Hardware Discussion

a) 7-Segment Display

A 7-segment display is assembled with seven LEDs, with some segment displays including an additional LED for a decimal point. Each LED has its leg to be inserted into a breadboard. Using microcontrollers, the LED can be controlled to display numbers and some alphabet letters that can be used to show hexadecimal numbers.

The 7-segment display can be configured into two types: common anode and common cathode. The common cathode can be achieved by connecting all cathode legs to the ground, thus lighting the LEDs requires the value of HIGH. On the other hand, the common anode is achieved by connecting all anode legs to Vcc and lighting the LEDs by giving the value LOW.

b) Breadboard

Breadboards are used to connect electronic components without the need of soldering.

c) Arduino board

Arduino, which comes in many forms, is a microcontroller that can be integrated into many electronics projects by utilising its pin and be controlled by programming.

d) Push button

A push button is a mechanical device that acts as a switch. They can be used as a normally open(NO) or normally closed(NC) switch. Depending on the connections.

e) Wire/ Jumper cable male-to-male

Wires are used to connect electrical components.

Software Discussion

The attached Arduino code controls a 7-segment display to show numbers from 0 to 9, incrementing each time a button is pressed. It features a debounce mechanism to ensure accurate detection of button presses, preventing multiple increments from a single press. Some key functions of this code include:

a) Pin Definitions

The code begins by defining the pins connected to each segment of the 7-segment display and the button. Each segment (A to G) is assigned a specific digital pin on the Arduino, while the button pin uses an internal pull-up resistor for reliable input.

b) Setup Function

In the `setup()` function, all segment pins are configured as outputs, and the button pin is set as an input. The function initializes the display to show the starting number, 0, by calling the `displayNumber()` function.

c) Loop Function

The `loop()` function continuously checks the button's state. It detects a button press, waits for a debounce delay to filter out noise, and increments `currentNumber` if the button remains pressed. The display is updated accordingly, resetting to 0 after reaching 9.

d) Debounce Function

The debounce functionality is integrated within the loop to manage button presses. It uses a timestamp to ensure that only one button press is registered within a specified delay period, effectively filtering out any rapid, unintended toggles when the button is pressed.

e) Display Function

The `displayNumber(int number)` function controls which segments are lit based on the current number. It first turns off all segments, then uses if-else statements to activate the appropriate segments for each digit from 0 to 9, ensuring the correct number is displayed.

ARDUINO CODE

```
// Define the pins for each segment (D0 to D6)
const int segmentA = 2; // D0
const int segmentB = 3; // D1
const int segmentC = 4; // D2
const int segmentD = 5; // D3
const int segmentE = 6; // D4
const int segmentF = 7; // D5
const int segmentG = 10; // D6
const int buttonPin = 9; // Button pin

// Variable to hold the current number

int currentNumber = 0; // Start counting from 0

// Debounce variables
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;
bool lastButtonState = HIGH;

void setup() {
    // Initialize the digital pins as OUTPUTs
    pinMode(segmentA, OUTPUT);
    pinMode(segmentB, OUTPUT);
```

```

pinMode(segmentC, OUTPUT);

pinMode(segmentD, OUTPUT);           // Check for button press (with debounce)

pinMode(segmentE, OUTPUT);           if (buttonState == LOW &&
lastButtonState == HIGH) {

pinMode(segmentF, OUTPUT);           // Wait for debounce delay

pinMode(segmentG, OUTPUT);           delay(debounceDelay);

// Initialize button pin

// Read the button state again
pinMode(buttonPin, INPUT_PULLUP);

buttonState = digitalRead(buttonPin);

// Use internal pull-up resistor

// Display the initial number (0)           // Only increment if the button is still
pressed

displayNumber(currentNumber); // Show
zero initially

if (buttonState == LOW) {

// Button was pressed, increase the
number

currentNumber++;

if (currentNumber > 9) currentNumber
= 0; // Reset to 0 after 9

}

void loop() {

// Read the button state

int buttonState = digitalRead(buttonPin);

```

```
displayNumber(currentNumber); //
Update display with new number
```

```
}
```

```
}
```

```
lastButtonState = buttonState; // Store the
last button state
```

```
}
```

```
// Function to display the current number
on the 7-segment display
```

```
void displayNumber(int number) {
```

```
    // Clear all segments first
```

```
    digitalWrite(segmentA, LOW);
```

```
    digitalWrite(segmentB, LOW);
```

```
    digitalWrite(segmentC, LOW);
```

```
    digitalWrite(segmentD, LOW);
```

```
    digitalWrite(segmentE, LOW);
```

```
    digitalWrite(segmentF, LOW);
```

```
digitalWrite(segmentG, LOW);
```

```
    // Set segments based on the number
```

```
using if-else statements
```

```
if (number == 1) {
```

```
    digitalWrite(segmentB, HIGH); // b
```

```
    digitalWrite(segmentC, HIGH); // c
```

```
} else if (number == 2) {
```

```
    digitalWrite(segmentA, HIGH); // a
```

```
    digitalWrite(segmentB, HIGH); // b
```

```
    digitalWrite(segmentD, HIGH); // d
```

```
    digitalWrite(segmentE, HIGH); // e
```

```
    digitalWrite(segmentG, HIGH); // g
```

```
} else if (number == 3) {
```

```
    digitalWrite(segmentA, HIGH); // a
```

```
    digitalWrite(segmentB, HIGH); // b
```

```
    digitalWrite(segmentC, HIGH); // c
```

```
    digitalWrite(segmentD, HIGH); // d
```

```

digitalWrite(segmentG, HIGH); // g

} else if (number == 4) {

digitalWrite(segmentB, HIGH); // b

digitalWrite(segmentC, HIGH); // c

digitalWrite(segmentF, HIGH); // f

digitalWrite(segmentG, HIGH); // g

} else if (number == 5) {

digitalWrite(segmentA, HIGH); // a

digitalWrite(segmentC, HIGH); // c

digitalWrite(segmentD, HIGH); // d

digitalWrite(segmentF, HIGH); // f

digitalWrite(segmentG, HIGH); // g

} else if (number == 6) {

digitalWrite(segmentA, HIGH); // a

digitalWrite(segmentB, HIGH); // b

digitalWrite(segmentC, HIGH); // c

digitalWrite(segmentD, HIGH); // d

digitalWrite(segmentE, HIGH); // e

```

```

digitalWrite(segmentF, HIGH); // f

digitalWrite(segmentG, HIGH); // g

} else if (number == 7) {

digitalWrite(segmentA, HIGH); // a

digitalWrite(segmentB, HIGH); // b

digitalWrite(segmentC, HIGH); // c

} else if (number == 8) {

digitalWrite(segmentA, HIGH); // a

digitalWrite(segmentB, HIGH); // b

digitalWrite(segmentC, HIGH); // c

digitalWrite(segmentD, HIGH); // d

digitalWrite(segmentE, HIGH); // e

digitalWrite(segmentF, HIGH); // f

digitalWrite(segmentG, HIGH); // g

} else if (number == 9) {

digitalWrite(segmentA, HIGH); // a

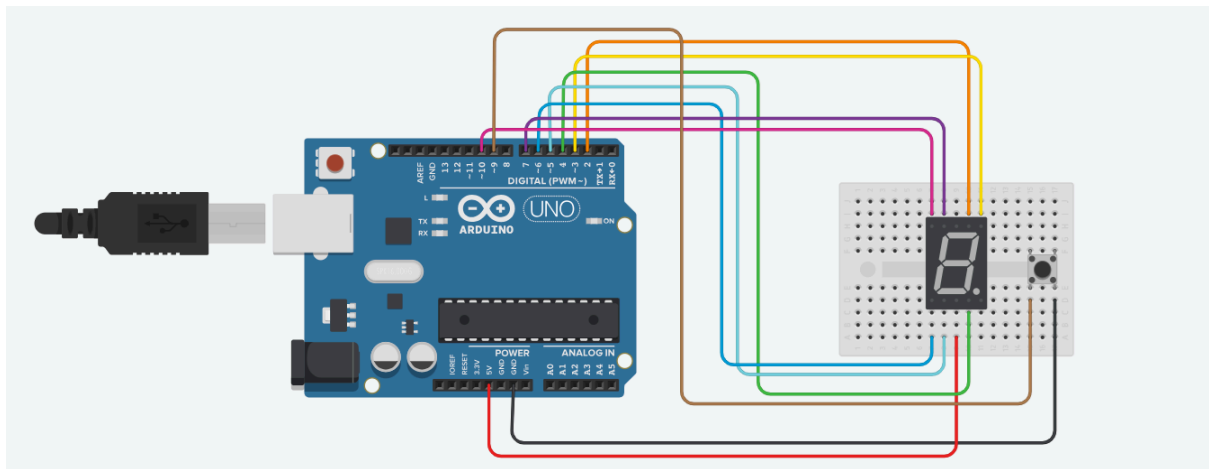
digitalWrite(segmentB, HIGH); // b

digitalWrite(segmentC, HIGH); // c

```

digitalWrite(segmentD, HIGH); // d	digitalWrite(segmentD, HIGH); // d
digitalWrite(segmentF, HIGH); // f	digitalWrite(segmentE, HIGH); // e
digitalWrite(segmentG, HIGH); // g	digitalWrite(segmentF, HIGH); // f
} else if (number == 0) {	}
digitalWrite(segmentA, HIGH); // a	}
digitalWrite(segmentB, HIGH); // b	
digitalWrite(segmentC, HIGH); // c	

Electrical discussion



Question

1. How to interface an I2C LCD with Arduino? Explain the coding principle behind it compared with 7 segments display and matrix LED.

Interfacing I2C LCD with Arduino

The I2C LCD can be connected to the Arduino by connecting the VCC pin to 5V, the GND pin to the ground, the SDA pin and the SCL pin connected to the designated analogue pins. This setup allows for easy communication between the Arduino and the LCD.

I2C LCD Coding Principles and comparison 7 segment display and matrix LED

The I2C LCD uses libraries like LiquidCrystal_I2C to simplify communication. This abstraction allows users to send commands easily without needing to manage the low-level I2C

protocol. The LCD is initialised with its I2C address and dimensions using functions like `lcd.begin()` and `lcd.backlight()`. For displaying text, functions like `lcd.setCursor(column, row)` and `lcd.print("text")` are employed, making it straightforward to output messages. This ease of use is one of the key advantages of I2C LCDs for textual displays.

However, 7-segment displays are controlled by individual pins for each segment (A-G). Users must specify which segments to light up for each digit, leading to more complex logic. Coding often involves if-else statements or arrays to determine segment activation.

Moreover, Matrix LEDs consist of an array of LEDs that can be individually controlled. Users manage which rows and columns are activated, allowing for complex patterns and graphics. This requires advanced logic, often involving bit manipulation or character maps. Coding for matrix LEDs can be intricate, and while libraries like `Adafruit_GFX` can simplify some aspects, a solid understanding of the grid layout is necessary. They excel at displaying dynamic content, such as animations and scrolling text.

In conclusion, I2C LCDs offer the simplest interface for text display, while 7-segment displays require more intricate segment control. Matrix LEDs are the most complex due to their grid structure. When it comes to display capabilities, I2C LCDs are ideal for text, 7-segment displays for numbers, and matrix LEDs for graphics and animations. Each display type has unique strengths and coding challenges, with I2C LCDs standing out for their simplicity and effectiveness.

Recommendation

For the recommendation, we suggest adding a second push button for decreasing the count. This would allow users to both increment and decrement the number displayed, making the system more versatile and improving the functionality of user input. Next, we recommend expanding the project by incorporating a multi-digit 7-segment display. By using multiple displays, you can show larger numbers or more complex outputs, such as a four-digit system to create a basic counter or clock. Lastly, consider introducing an external debounce component like a capacitor to improve input reliability. This would ensure that button presses are registered more consistently, reducing the chances of missed or unintended counts.

Conclusion

To conclude, this project shows how to control a 7-segment display using an Arduino and a push-button switch. We faced many challenges and setbacks during the development, even though the procedures seemed simple. By carefully going through the code line by line and trying different solutions, we were able to fix the problems and make the system work.


This experience taught us the importance of being persistent and paying attention to detail in programming. We also learned that understanding how the 7-segment display works and how its connection in the circuit responded with the code, was essential for writing the functional code.


Overall, this project gives a solid foundation in digital control systems and opens the door for future improvements and more complex projects.


Student Declaration


This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been taken or done by unspecified sources or person. We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report. We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report have been verified by us.

NAME: Muhammad Basil bin Abdul Hakim	READ	✓
MATRIC NO: 2215315	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: Muhammad Syafiq Bin Nor Azman	READ	✓
MATRIC NO: 2213187	UNDERSTAND	✓
SIGNATURE 	AGREE	✓

NAME: Muhammad Hafiz Hamzah Bin Fansuri	READ	✓
MATRIC NO: 2212803	UNDERSTAND	✓
SIGNATURE : 	AGREE	✓

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✓
MATRIC NO: 2110259	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✓
MATRIC NO: 2120225	UNDERSTAND	✓
SIGNATURE	AGREE	✓