Mechatronics System Integration (MCTA3203) Semester 2, 2023/2024

Report: experiment; week 2

Lecturers:

Dr/ WAHJU SEDIONO

Dr/ ZULKIFLI BIN ZAINAL ABIDIN

Dr/ ALI SOPHIAN

Section 1 Group C

Names	**Matric**
Abdallah Mahamat Abacar	2117777
Almasri Suhail Jihad	2128771
Ibrahim Bin Nasrum	2116467
Khan Toqi Tahmid	2025209

Table of Contents

Abstract

This experiment explores the integration of a 7-segment display with an Arduino microcontroller and pushbutton input. The 7-segment display is a common output device used for visualizing numerical data. By interfacing it with an Arduino board and utilizing pushbuttons as input, a simple interactive system is created. The Arduino reads the state of the pushbuttons and displays corresponding numbers on the 7-segment display. This project demonstrates basic digital input and output concepts, along with the practical application of multiplexing techniques to drive the 7-segment display efficiently. Additionally, it serves as a foundational exercise for beginners in the realm of microcontroller programming and electronic hardware interfacing.

Introduction

The purpose of the experiment is to explore the functionality of the 7 segment display by integrating it with the Arduino UNO and managing it through push buttons. The Arduino UNO will illuminate specific segments on the 7 segment display based on the programmed code. Additionally, the use of the push button enables us to increase the displayed number.

Materials Needed:

- Arduino Uno board
- Common cathode 7-segment display
- 220-ohm resistors (7 of them)
- Pushbuttons (2 or more)
- Jumper wires
- Breadboard

Experimental Setup:

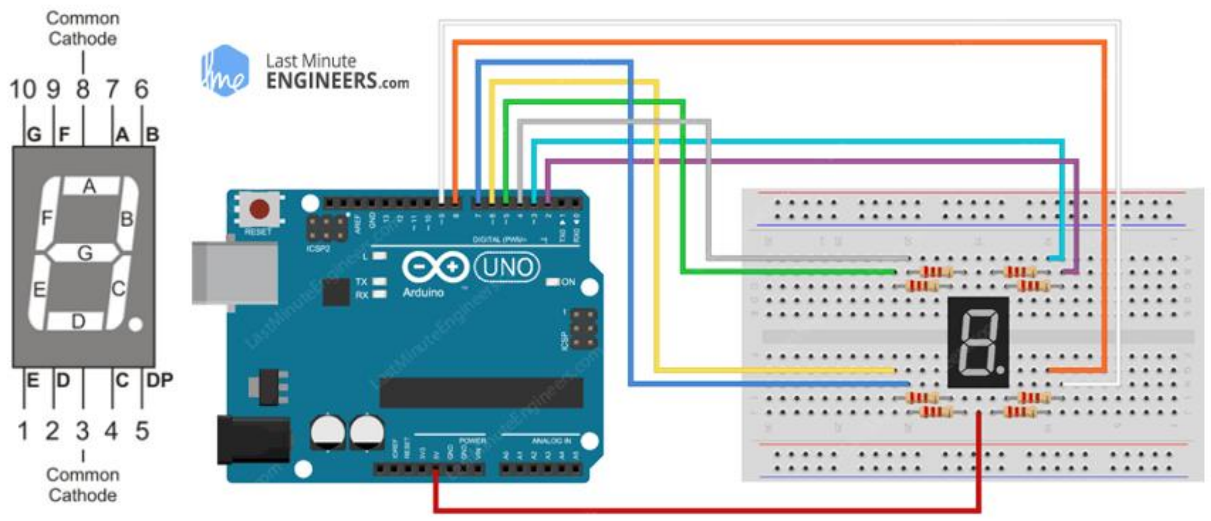
1. Connect the common cathode 7-segment display to the Arduino Uno as follows:
    - Connect each of the 7 segments (a, b, c, d, e, f, g) of the display to separate digital pins on the Arduino (e.g., D0 to D6).
    - Connect the common cathode pin of the display to one of the GND (ground) pins on the Arduino.
    - Use 220-ohm resistors to connect each of the segment pins to the Arduino pins to limit the current.
2. Connect the pushbuttons to the Arduino:
    - Connect one leg of each pushbutton to a separate digital pin (e.g., D9 and D10) and connect the other leg of each pushbutton to GND.
    - Use 10K-ohm pull-up resistors for each pushbutton by connecting one end of each resistor to the digital pin and the other end to the 5V output of the Arduino.

Experiment Steps:

1. Build the circuit according to the circuit setup instructions.
2. Upload the provided Arduino code to your Arduino Uno.
3. Open the Serial Monitor in the Arduino IDE.
4. Press the increment button to increase the count. The 7-segment display should show the numbers from 0 to 9 sequentially.
5. Press the reset button to reset the count to 0.

Methodology:

After constructing the required circuit setup, we proceeded to upload Arduino code onto the Arduino Mega. Utilizing the pushbutton, we incremented the displayed number on the 7-segment display sequentially from 0 to 9. Once the display reached 9, it looped back to 0, creating a continuous cycle of numerical display.



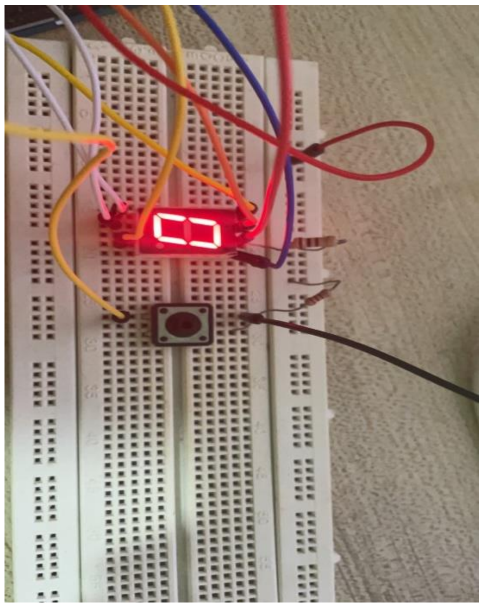
Arduino Code:

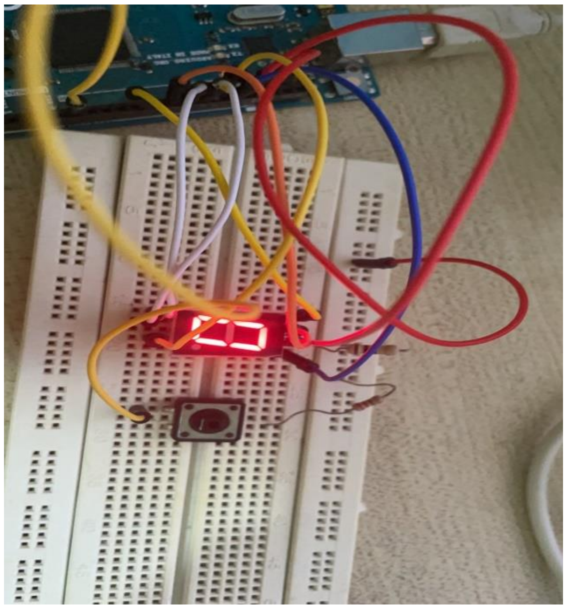
```
int pinA = 13;

int pinB = 12;

int pinC = 11;

int pinD = 10;

int pinE = 9;

int pinF = 8;

int pinG = 7;

int i = 0;

int j = 0;

int Arduino_Pins[7] = {pinA, pinB, pinC, pinD, pinE, pinF, pinG};

int Segment_Pins[10][7] = {{1, 1, 1, 1, 1, 1, 0}, // 0

                    {0, 1, 1, 0, 0, 0, 0}, // 1

                    {1, 1, 0, 1, 1, 0, 1}, // 2

                    {1, 1, 1, 1, 0, 0, 1}, // 3

                    {0, 1, 1, 0, 0, 1, 1}, // 4

                    {1, 0, 1, 1, 0, 1, 1}, // 5

                    {1, 0, 1, 1, 1, 1, 1}, // 6

                    {1, 1, 1, 0, 0, 0, 0}, // 7

                    {1, 1, 1, 1, 1, 1, 1}, // 8

                    {1, 1, 1, 1, 0, 1, 1}, // 9

};


void setup() {

  // put your setup code here, to run once:

  pinMode(pinA, OUTPUT);

  pinMode(pinB, OUTPUT);

  pinMode(pinC, OUTPUT);

  pinMode(pinD, OUTPUT);
```

```
 pinMode(pinE, OUTPUT);
 pinMode(pinF, OUTPUT);
 pinMode(pinG, OUTPUT);
 pinMode(2, INPUT_PULLUP);
}

void loop() {
 // put your main code here, to run repeatedly:
  for (j = 0; j<7; j++)
 {
  digitalWrite(Arduino_Pins[j], Segment_Pins[i][j]);
 }

 if(digitalRead(2)==0)
 {
  while(digitalRead(2)==0)
  {
  }
  i++; // i = i + 1;
 }
 if(i == 10)
 {
  i = 0;
 }
 delay(100);
}
```

RESULT

The conducted experiment resulted in the successful creation of a functional counter display system, capable of showcasing numbers ranging from 0 to 9. This system was complemented by the inclusion of a pushbutton mechanism, strategically designed to reset the display back to 0 while enabling incremental increases in numerical value by increments of 1. This integrated button functionality not only facilitated the reset feature but also enhanced the interactivity and usability of the display, providing users with a straightforward means to control and manipulate the displayed numbers with ease.





DISCUSSION

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

To interface an I2C LCD with an Arduino:

1- connecting the SDA (data) and SCL (clock) pins of the LCD module to the corresponding pins on the Arduino.

2- Download and install the LiquidCrystal_I2C library for Arduino.

3- Initialize the LCD object by specifying the I2C address of your LCD module and the number of columns and rows your display has.

4- Use functions like lcd.print( ) or lcd.setCursor() to display text on the LCD screen.

As for the coding principle behind it compared with 7-segment displays and matrix LED:

I2C LCD vs. 7-Segment Display:

- I2C LCD: With an I2C LCD, communication between the Arduino and the display module occurs over the I2C protocol, which requires fewer pins compared to direct control. This makes it simpler to connect and frees up more pins for other components or sensors.
- 7-Segment Display: In contrast, a 7-segment display typically requires more pins for direct control, with each segment connected individually or in groups. This can be more complex to set up and control directly but may offer better performance in terms of refresh rate and brightness control.

I2C LCD vs. Matrix LED:

- I2C LCD: The I2C LCD communicates with the Arduino using the I2C protocol, which allows for serial communication over longer distances with fewer wires. This is advantageous for projects requiring displays at a distance from the main Arduino board.
- Matrix LED: Matrix LEDs, such as LED dot matrices, are arrays of individually addressable LEDs. Controlling them typically involves multiplexing techniques to light up specific LEDs in patterns to display characters or images. While matrix LEDs offer more flexibility in terms of display options, they also require more complex control logic and often more pins compared to an I2C LCD.

CONCLUSION

In conclusion, the experiment integrating a 7-segment display with Arduino and pushbuttons has provided a hands-on exploration of fundamental concepts in digital interfacing and user interaction. By employing pushbuttons for count control, the experiment showcased the versatility of the 7-segment display in practical applications, such as numerical displays and basic user input systems. This exercise not only demonstrated the technical aspects of hardware integration but also emphasized the importance of user-centric design in creating interactive electronic systems. Through this experiment, participants gained valuable experience in microcontroller programming, circuitry, and the seamless integration of input and output devices, laying a solid foundation for further exploration in the realm of embedded systems and electronic prototyping.

REFERENCES

- https://www.circuitgeeks.com/arduino-7-segment-display-tutorial/
- https://circuitdigest.com/microcontroller-projects/interfacing-seven-segment-display-with-Arduino