LABORATORY REPORT

LAB 1: DIGITAL LOGIC SYSTEM

GROUP B

PROGRAMME: MECHATRONIC ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. MUHAMMAD AFIQ BIN AHMAD	2115203
2. HANIS BINTI MOHD IZANI	2020590
3. SARAH AISHAH BINTI SA'AID HAZLEY	2117600
4. NUR AYU NAZIERA BINTI ROSLI	2119202
5. NOORUL GHOUSIAH BINTI NOORDEEN SAHIB	2118298

DATE OF SUBMISSION:
Wednesday, 25th October 2023

ABSTRACT

In this lab report, the interfacing of a common cathode 7-segment display with an Arduino Uno microcontroller and the manual control of its output using pushbuttons is investigated. The experiment involves building a circuit that connects the 7-segment display to the Arduino Uno, utilising appropriate resistors for current limiting, and integrating pushbuttons for user input. Using a pre-uploaded Arduino code, numbers are displayed sequentially from 0 to 9 on the 7-segment display when pressing an increment button. The other push button, which is the reset button, is pressed to return the display to zero. The experiment offers important insights into the use of digital electronics, microcontroller programming, and sensor interfacing, providing understanding of these fundamental ideas for upcoming projects and applications.

TABLE OF CONTENTS

Table of Contents

MATERIALS AND EQUIPMENT

Materials

- Arduino Uno board
- Common cathode 7-segment display
- 220-ohm resistors (4 of them)
- Pushbuttons (2 of them)
- Jumper wires
- Breadboard

EXPERIMENTAL SETUP

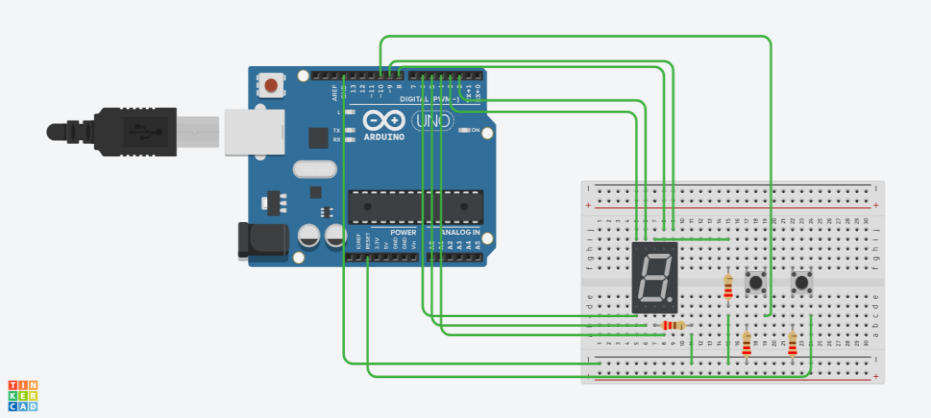
Equipment setup

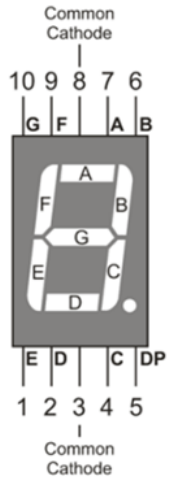
1.The common cathode 7-segment display is connected to the Arduino Uno as following:

- Each of the 7 segments (a, b, c, d, e, f, g) of the display is connected to separate digital pins on the Arduino (e.g., D0 to D6).
- The common cathode pin of the display is connected to one of the GND (ground) pins on the Arduino.
- 220-ohm resistors were used to connect each of the segment pins to the Arduino pins to limit the current.

2. The pushbuttons is connected to the Arduino:

- One leg of each pushbutton is connected to a separate digital pin (e.g., D9 and D10) and connect the other leg of each pushbutton to GND.
- 10K-ohm pull-up resistors were used 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.





METHODOLOGY

1. The circuit was built according to the circuit setup instructions.
2. The Arduino code was uploaded to our Arduino Uno.
3. The Serial Monitor in the Arduino IDE was opened.
4. The increment button was pressed to increase the count. The 7-segment display shows the numbers from 0 to 9 sequentially.
5. Then reset the button to reset the count to 0.

The following was the code that we used in this experimental setup.

```
const int a = 8;  //For displaying segment "a"
const int b = 9;  //For displaying segment "b"
const int c = 4;  //For displaying segment "c"
const int d = 5;  //For displaying segment "d"
const int e = 6;  //For displaying segment "e"
const int f = 2;  //For displaying segment "f"
const int g = 3;  //For displaying segment "g"
bool bPress = false;
const int IncbuttonPin = 10;
const int resetbutton = 11;
// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int IncbuttonState = 0;         // current state of the button
int lastIncbuttonState = 0;     // previous state of the button
int resetstate=0;

void setup() {
  // put your setup code here, to run once:
  pinMode(a, OUTPUT);  //A
  pinMode(b, OUTPUT);  //B
  pinMode(c, OUTPUT);  //C
  pinMode(d, OUTPUT);  //D
  pinMode(e, OUTPUT);  //E
  pinMode(f, OUTPUT);  //F
  pinMode(g, OUTPUT);  //G
  pinMode( IncbuttonPin , INPUT_PULLUP );
  pinMode( resetbutton , INPUT );
  Serial.begin(9600);
  displayDigit(buttonPushCounter);
}
void loop() {
   resetstate=digitalRead(resetbutton);
   //Serial.println(resetstate);
   IncbuttonState = digitalRead(IncbuttonPin);
   checkIncButtonPress();
  if( bPress ){
    bPress = false;
     turnOff();
     displayDigit(buttonPushCounter);
  }

}
void checkIncButtonPress()
{
   // compare the IncbuttonState to its previous state
  if (IncbuttonState != lastIncbuttonState) {
    // if the state has changed, increment the counter
    if (IncbuttonState == LOW) {
```

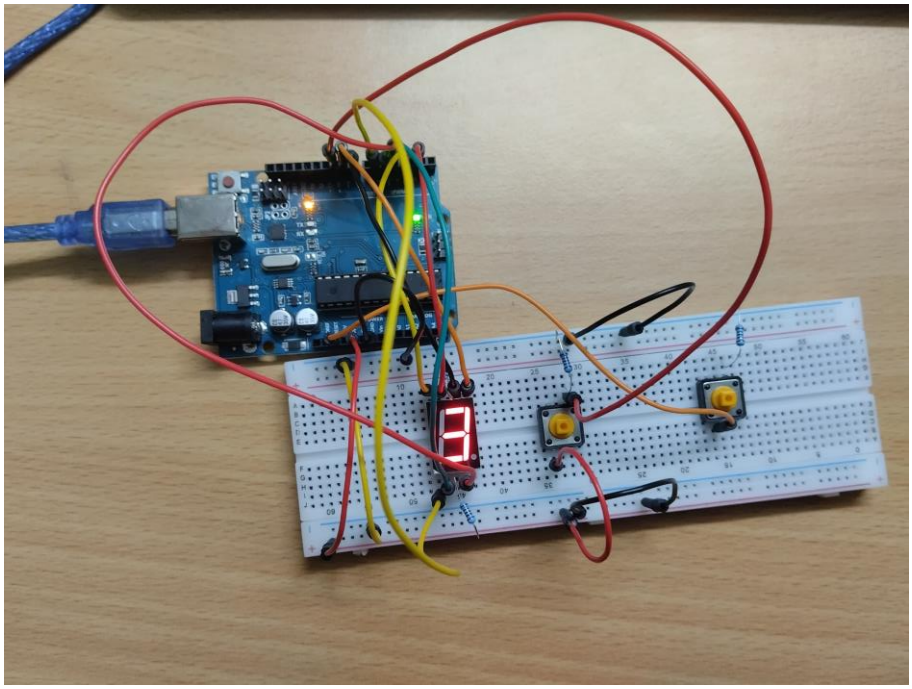
```
      // if the current state is HIGH then the button went from off to on:
      bPress = true;
      buttonPushCounter++;
      if( buttonPushCounter > 9 ) buttonPushCounter =0 ;
      if( resetstate==1) buttonPushCounter =0 ;
      Serial.println("on");
    } else {
      // if the current state is LOW then the button went from on to off:
      Serial.println("off");
    }
    // Delay a little bit to avoid bouncing
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastIncbuttonState = IncbuttonState;
}

void displayDigit(int digit)
{
 //Conditions for displaying segment a
 if(digit!=1 && digit != 4)
 digitalWrite(a,HIGH);
 //Conditions for displaying segment b
 if(digit != 5 && digit != 6)
 digitalWrite(b,HIGH);
 //Conditions for displaying segment c
 if(digit !=2)
 digitalWrite(c,HIGH);
 //Conditions for displaying segment d
 if(digit != 1 && digit !=4 && digit !=7)
 digitalWrite(d,HIGH);
 //Conditions for displaying segment e
 if(digit == 2 || digit ==6 || digit == 8 || digit==0)
 digitalWrite(e,HIGH);
 //Conditions for displaying segment f
 if(digit != 1 && digit !=2 && digit!=3 && digit !=7)
 digitalWrite(f,HIGH);
 if (digit!=0 && digit!=1 && digit !=7)
 digitalWrite(g,HIGH);
}
void turnOff()
{
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
```

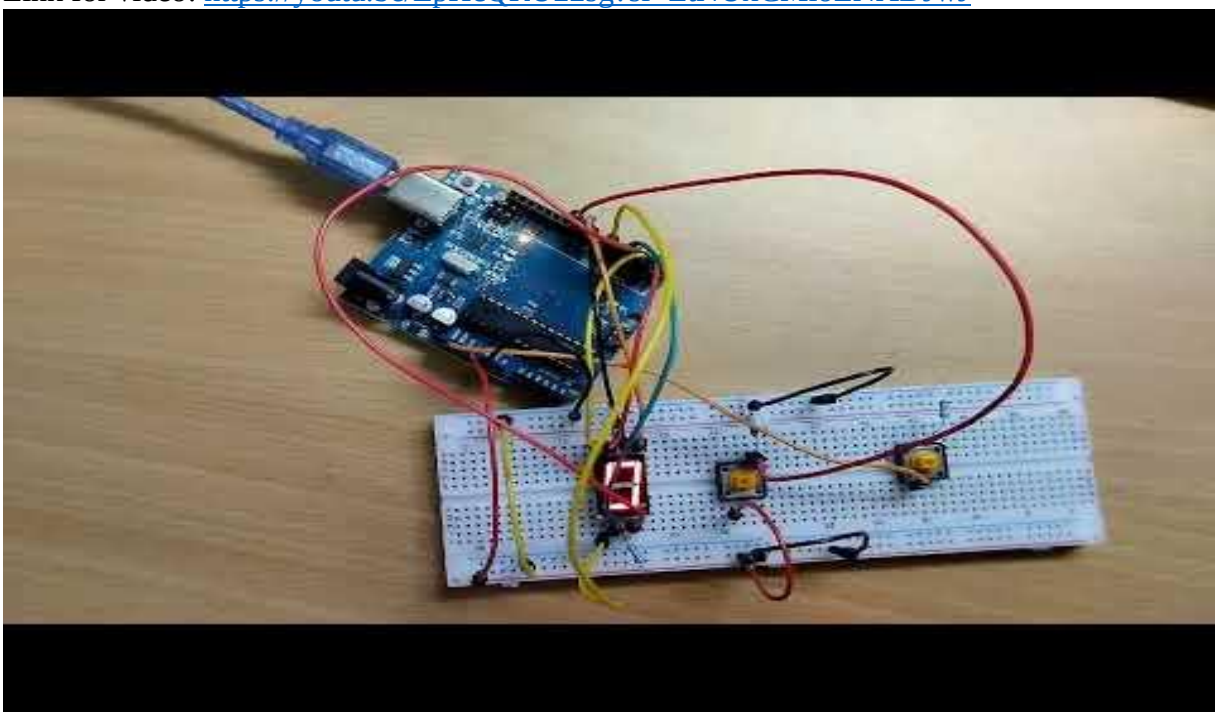
```
  digitalWrite(g,LOW);
}
```

RESULTS (Observation)

After a few attempts, the 7-segment display successfully interfaced with Arduino. As we can see the 7-segment display numerical digits ranging from 0-9 and was arranging in sequential manner. It also displays successfully counted up and down with the response from the push buttons presses. The experiment was a success since it allowed the project to include a mechanism for increasing the count via the plus button while simultaneously facilitating count decrease using the down button. The uses of logic gates are mostly defined by their truth table and mode of operation. Many circuits employ fundamental logic gates, such as a light-activated burglar alarm, a safety thermostat, and an automated watering system. As a result of this successful operation, such a system may be employed in a variety of different sectors where tracking or counting is necessary.



Link for video: https://youtu.be/EpHcQRU22sg?si=ZtNUxCMioZNADJwJ



DISCUSSION

The 7-segment display interfaced with Arduino successfully after a few trials. This is mainly due to the mistake of connecting the 7-segment display to the Arduino board directly without resistors. This causes the component to be burnt out and damaged. After connection is done correctly using a new 7-segment display with resistors included, the new 7-segment display works well.

Interfacing I2C LCD with Arduino

I2C LCD uses I2C communication interface. I2C is a serial communication interface to communicate with other I2C devices. To interface I2C LCD to Arduino, connect SDA pin to SDA pin and SCL pin to SCL pin between the Arduino and the LCD. Then, the VCC pin in LCD is connected to 5V in Arduino, and the ground pin of LCD is connected to the ground pin in Arduino. Then the circuit connection is complete as shown in the diagram below.

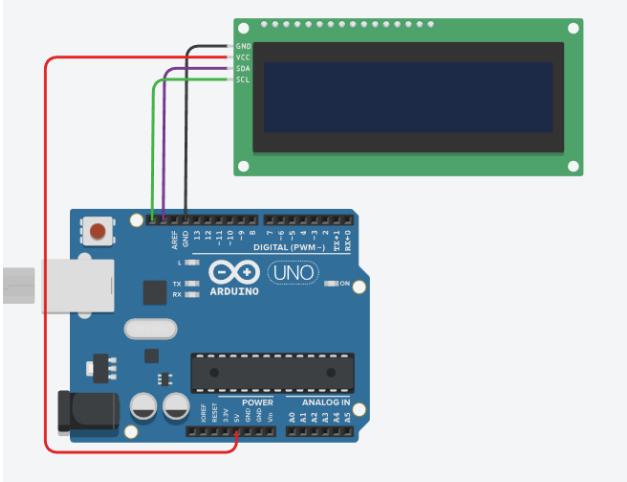


Diagram of LCD display interfacing circuit

Coding principle between I2C LCD, 7-segment display, and matrix LED

I2C LCD needs a library named “LiquidCrystal I2C” to be defined as a header file in the Arduino IDE code. After that, a snippet of code is compiled and run in order to find the I2C address. The address obtained, together with height and width, will be used in defining the display device in the code. In the setup() function, initialization and backlight are enabled. In the loop() function, all appropriate functions can be included in it, this includes, lcd.clear() for clearing the content in the display, lcd.setCursor() for setting the cursor, and lcd.print() for printing the text where the cursor is set.

There are two types of 7 segment displays available, that is common cathode and common anode. If you have a common cathode display, there will be two cathodes and eight anodes for eight segments. And if you have a common anode display, there will be two anodes and eight cathodes for eight segments.

For common cathode display of 7 segment display, firstly, all eight anode pins are defined in the Arduino IDE. In the setup() function, the mode of each pins are set to be output. In the loop() function, all appropriate functions can be included in it, this includes, digitalWrite() for writing ‘HIGH’ or ‘LOW’ value to a digital pin, and delay() for pausing the program for a specified duration.

The LED matrix works with the help of LED driver. The Arduino will connect to an LED driver and send commands to it with the SPI communication protocol. Based on commands from the Arduino, the LED driver will output high or low signals to turn each LED on the LED matrix on and off.

LED matrix needs the following libraries to be defined as header files in the Arduino IDE code: ‘SPI’, ‘Adafruit GFX’, and ‘MAX72xxPanel’. Then. an integer variable called ‘pinCS’ that defines which Arduino pin the CS pin of the LED driver is connected to is declared. Two more variables are also declared, that is, numberOfHorizontalDisplays and

numberOfVerticalDisplays, to set the number of horizontal and vertical displays of LED matrix used. Next, an object called matrix is created for the display, which is a member of the MAX72xxPanel class. The pinCS, numberOfHorizontalDisplays, and numberOfVerticalDisplays are passed as variables for the object. After that, the character to be displayed is defined with an array of bytes, which is also known as bitmap. Each byte corresponds to one row of LEDs on the LED matrix, while each bit corresponds to the column LED in that row. In the setup() function, setIntensity() can be used to set the brightness of the display. In the loop() function, all appropriate function can be used including drawBitmap() and write() for printing the character defined in the bitmap.

CONCLUSION

In conclusion, a useful interface between the 7-segment display and the Arduino was created by building a well-organised circuit and using the proper resistors for current control. We were able to display numbers 0 through 9 sequentially on the 7-segment display by pressing an increment button and to reset the display to 0 by hitting a different button by utilising pre-uploaded Arduino code. Digital electronics, microcontroller programming, and sensor integration may now be understood and applied practically.

RECOMMENDATIONS

While carrying out the experiment, our group faced a problem of our arduino not functioning correctly, as we were not able to upload our code to the arduino board. We were not sure if the arduino board or the USB cable was the faulty one. In the future, testing components separately would be vital before integrating them. This would save time and allow us to find working components.

Next, the problem of the 7-segment display not displaying the correct numbers. This meant that the code uploaded did not coincide with the pins and connections. Next time around, code reviewing and debugging would be essential. We would have to ensure variables, pin assignments, and logic are accurate and appropriately set so that the desired output can be achieved.

REFERENCES

GeeksforGeeks (n.d.). *How to interface I2C LCD display with Arduino?* Retrieved October 20, 2023, from https://www.geeksforgeeks.org/how-to-interface-i2c-lcd-display-with-arduino/

Circuitgeeks (2023, September 7). *Interfacing 7 Segment Display with Arduino*. Circuitgeeks.com. Retrieved October 20, 2023, from https://www.circuitgeeks.com/arduino-7-segment-display-tutorial/

ArduinoGetStarted (n.d.). *Arduino - LED Matrix*. Arduino Get Started. https://arduinogetstarted.com/tutorials/arduino-led-matrix

Scott Campbell (n.d.). *HOW TO SETUP LED MATRIX DISPLAYS ON THE ARDUINO*. Retrieved October 20, 2023, from https://www.circuitbasics.com/how-to-setup-an-led-matrix-on-the-arduino/

Chris (n.d.). *5 Simple Ways to Reset Arduino*. Chip Wired. Retrieved October 24, 2023, from https://chipwired.com/5-simple-ways-to-reset-arduino/

APPENDICES

ACKNOWLEDGEMENT

We would like to express our gratitude to Assoc. Prof. Dr. Zulkifli Bin Zainal Abidin, one of the lecturers of this course, mechatronic system integration 1 (MCTA 3203) for his guidance while the experiment is carried out.

STUDENT'S 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 untaken 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 contributor 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.

Signature: *Ghousiah* Name: Noorul Ghousiah Binti Noordeen Sahib Matric Number: 2118298 Contribution: Discussion, References	Read Understand Agree	/ / /
Signature: *hanis*	Read	/

Name: hanis Matric Number: 2020590 Contribution:	Understand	/
	Agree	/
Signature: *ayunaziera* Name: Nur Ayu Naziera Binti Rosli Matric Number: 2119202 Contribution: Material and Equipment, Experiment setup, Methodology	Read	/
	Understand	/
	Agree	/
Signature: *Afiq* Name: Muhammad Afiq bin Ahmad Matric Number: 2115203 Contribution:	Read	/
	Understand	/
	Agree	/
Signature: *Sarah* Name: Sarah Aishah binti Sa'aid Hazley Matric Number: 2117600 Contribution: Abstract, Recommendations, Conclusion	Read	/
	Understand	/
	Agree	/