Vincent Mary School of Engineering

EE3704 Embedded System

Name: Nathachanon Akesittipong
ID: 6110235

Submission Date: 28/02/2021

Work Contribution

1. Nathachanon Akesittipong 100%

Chapter 1: Introduction to Arduino Board

Ex1.1: Try Basic Command to Arduino Board

- Turn On/Off onboard LED (Pin 13) for 1 second

Diagram:

Pin 13 is the onboard LED itself which exists on the Arduino board.

Coding:

```
void setup()
{
        pinMode(13, OUTPUT);
}

void loop()
{
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
}
```

Result:

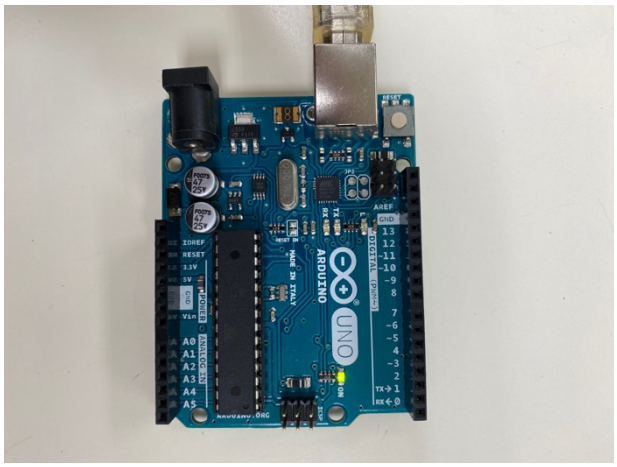


Figure 1.1.1: LED Off

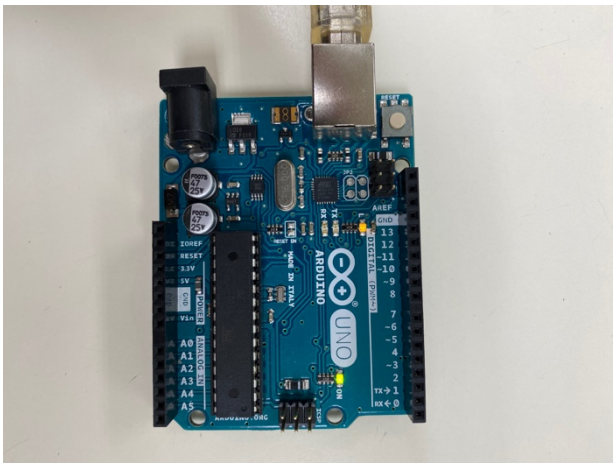


Figure 1.1.2: LED On

Ex 1.2:

- Turn On onboard Led (Pin 13) for 2 second
- Then Turn Off onboard LED (Pin 13) for 1 second
- Do it 5 time (On and Off)
- The onboard LED (Pin 13) turn Off

Diagram:

Pin 13 is the onboard LED itself which exists on the Arduino board.

Coding:

```
void setup()
{
	pinMode(13, OUTPUT);
}

void loop()
{
	for(int a=1; a<6; a:++)
	{
		digitalWrite(13, HIGH);
		delay(2000);
		digitalWrite(13, LOW);
		delay(1000);
	}
	digitalWrite(13, HIGH);
	digitalWrite(13, LOW);
}
```

Result:

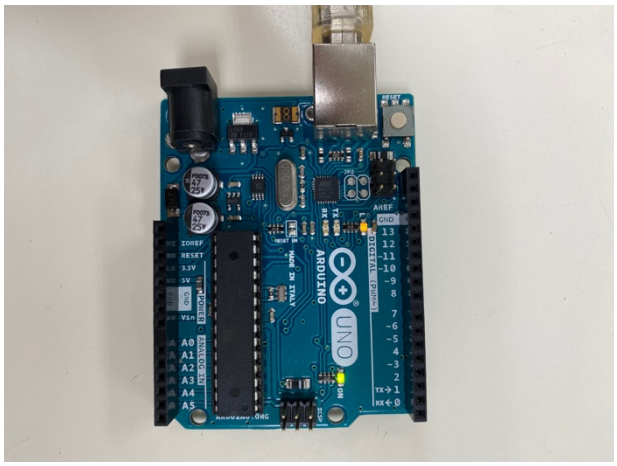


Figure 1.2.1: LED On

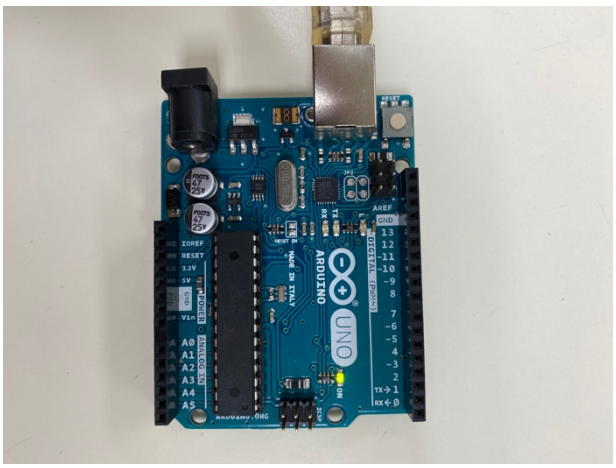


Figure 1.2.2: LED Off

Conclusion:

In chapter one, a basic concept, coding, and usage of Arduino were introduced. pinMode is a function used to declare the pin duty either as output or input. Where digitalWrite is a function used to command status of a particular pin to be either HIGH or LOW depends on the circuit mechanism. Moreover, some fundamental of C language were being reviewed and adapted into the coding process.

Chapter 2: Digital Output

Ex 2.1: Active High load

- Turn On / Off LED (Pin 13) for 5 times
- Check the LED output

Diagram:

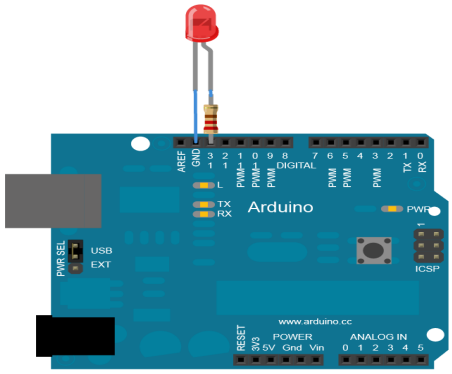


Figure 2.1.1

Coding:

```
void setup()
{
        pinMode(13, OUTPUT);
        blink();
}

void blink()
{
        for(int a=1;a<6;a++)
          {
                digitalWrite(13, HIGH);
                delay(1000);
                digitalWrite(13, LOW);
                delay(1000);
           }
}

void loop()
{}
```

Result:

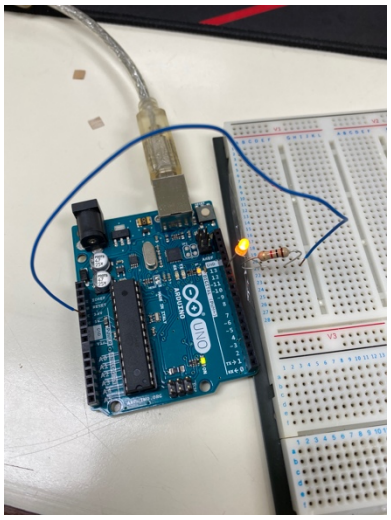


Figure 2.1.2: LED On

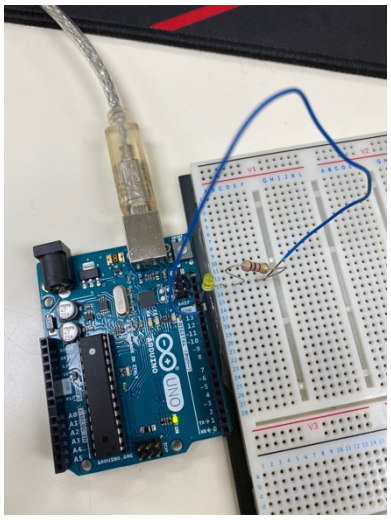


Figure 2.1.3: LED Off

Ex 2.2: Active High and Active Low loads

- Turn On / Off LED (Pin 13 as HIGH and Pin 12 as LOW) for 5 times
- Check the LED output

Diagram:

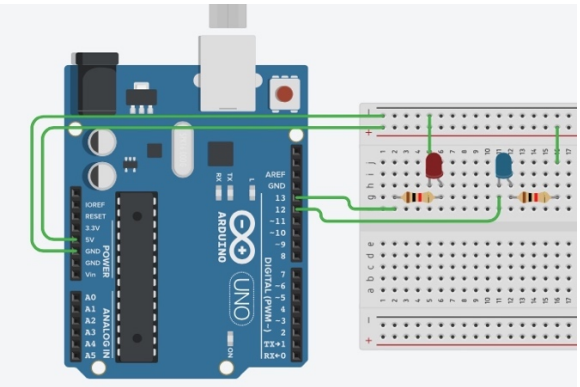


Figure 2.2.1

Coding:

```
void setup()
{
        pinMode(12, OUTPUT);
        pinMode(13, OUTPUT);
        blink();
}

void blink()
{
        for(int a=1;a<6;a++)
          {
                digitalWrite(12, LOW);
                digitalWrite(13, HIGH);
                delay(1000);
                digitalWrite(12, HIGH);
                digitalWrite(13, LOW);
                delay(1000);
           }
}

void loop()
{}
```

Result:

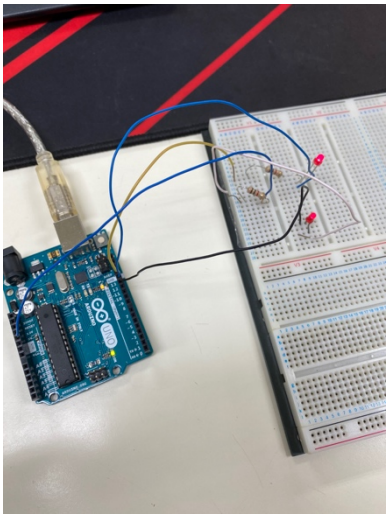


Figure 2.1.2: LED On

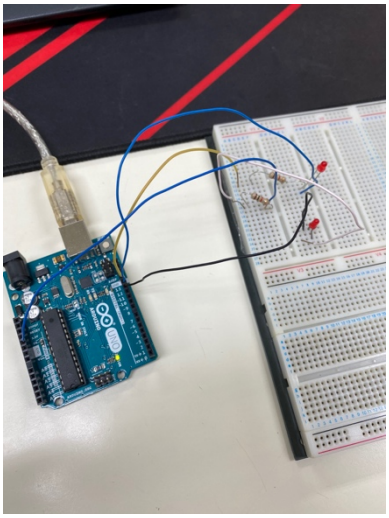


Figure 2.1.3: LED Off

Ex 2.3: (all Active Low)

- Connect LED to Pin 0 to 3
- Turn On / Off LED Pin 0 to 3 with delay of 1 second

Diagram:

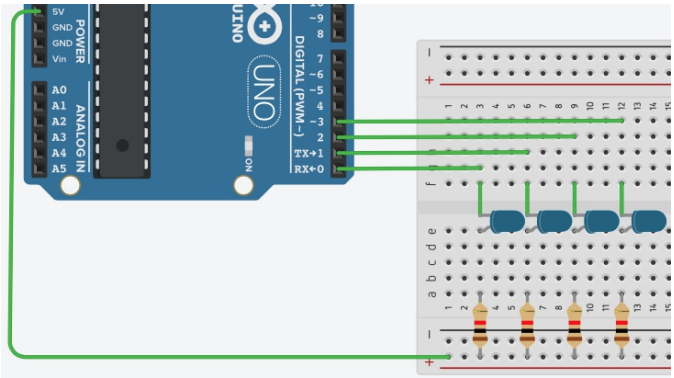


Figure 2.3.1

Coding:

```
void setup()
{
        int a;
        for(a = 0; a <4; a++)
         {
                pinMode(a, OUTPUT);
                 digitalWrite(a, HIGH);
         }
}

void loop()
{
        int a;
          for(a=0; a<4;a++)
        {
                 digitalWrite(a, LOW);
                 delay(1000);
                 digitalWrite(a, HIGH);
        }
}
```

Result:

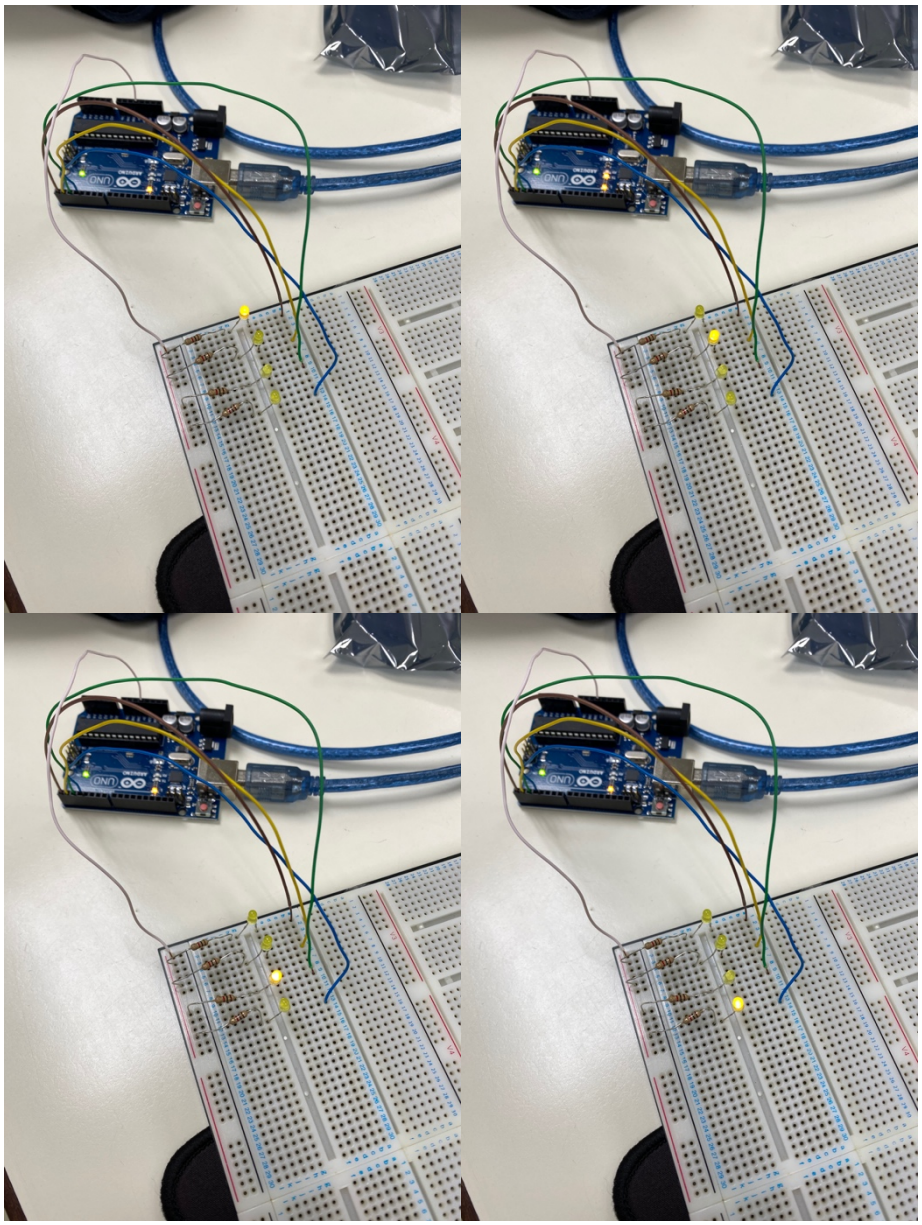


Figure 2.3.2 – 5: LEDs Blinking accordingly

Ex 2.4: 7 Segment (CC)

- Make 7 segment display number 5

Diagram:

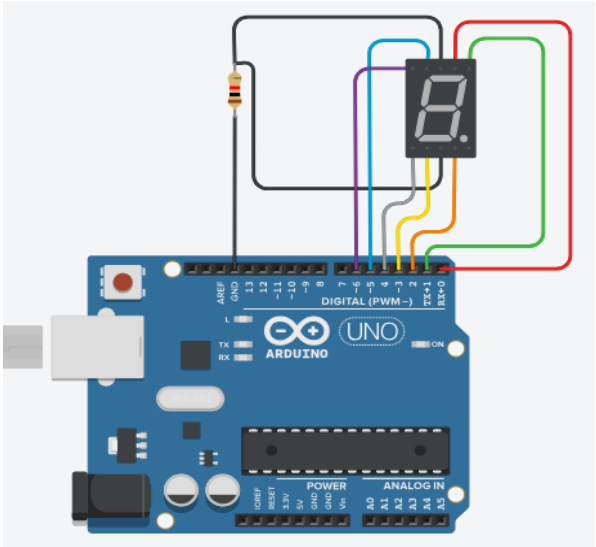


Figure 2.4.1

Coding:

```
void setup()
  {
          int a;
         for(a=0;a<7;a++)
          {
                  pinMode(a, OUTPUT);
         }
}

void loop()
{
         digitalWrite(0,HIGH);
         digitalWrite(2,HIGH);
         digitalWrite(3,HIGH);
         digitalWrite(5,HIGH);
         digitalWrite(6,HIGH);
}
```

Result:

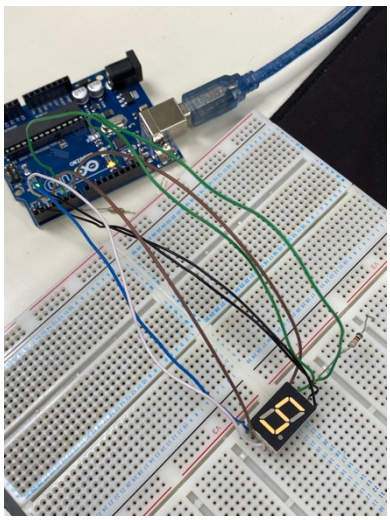


Figure 2.4.2

Ex 2.4 (Extra): 7 Segment (CC or CA)

- Make 7 segment display number 5 using array

Diagram:

Similar to figure 2.4.1

Coding:

```
void setup()
{
        int a;
        for(a=2;a<9;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        int a;
        int number_5[7] = {1,0,1,1,0,1,1};
        for(a=8; a>1; a--)
        {
                digitalWrite(a, number_5[a]);
        }
}
```

Result:

Similar to figure 2.4.2

Ex 2.4 (Extra x2): 7 Segment (CC)

- Make 7 segment display number 5 using array
- Starting from Pin2 to Pin 8

Diagram:

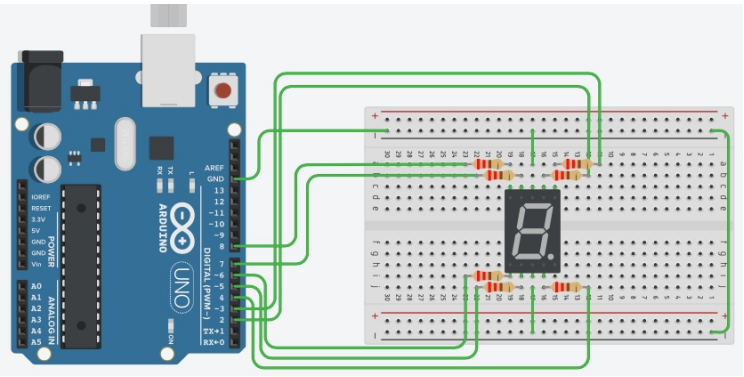


Figure 2.4.3

Coding:

```
void setup()
{
        int a;
         for(a=2;a<9;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        int a;
        int number_5[9] = {1,1,0,1,1,0,1};
        for(a=8; a>1; a--)
        {
                digitalWrite(a, number_5[8-a]);
         }
}
```

Result:

Similar to figure 2.4.2

Ex 2.4 (Extra x3): 7 Segment (CC)

- Make 7 segment display number 4 and 5 with 1s interval using array

Diagram:

Similar to figure 2.4.3

Coding:

```
void setup()
  {
        int a;
        for(a=2;a<9;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        int a, b;
         int number_45[2][7] = {{0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}}; // 4 and 5
        for(a=0;a<2;a++)
        {
                for(b=2;b<9;b++)
                {
                        digitalWrite(b ,number_45[a][b-2]);
                }
                 delay(1000);
        }
}
```

Result:

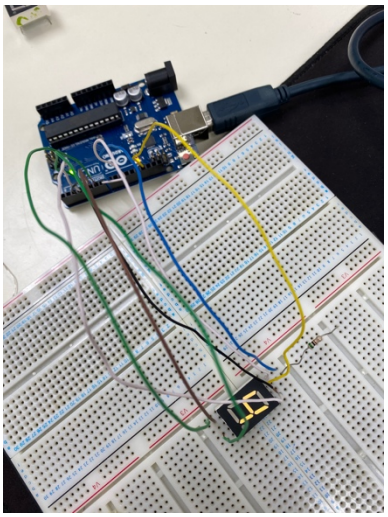


Figure 2.4.4

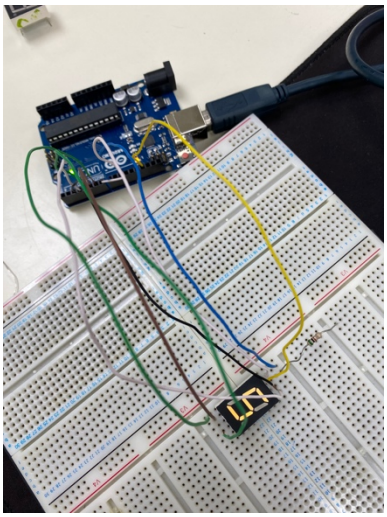


Figure 2.4.5

Ex 2.5:

- 7 Segment running 0 – 9 and repeat with delay of 1 second

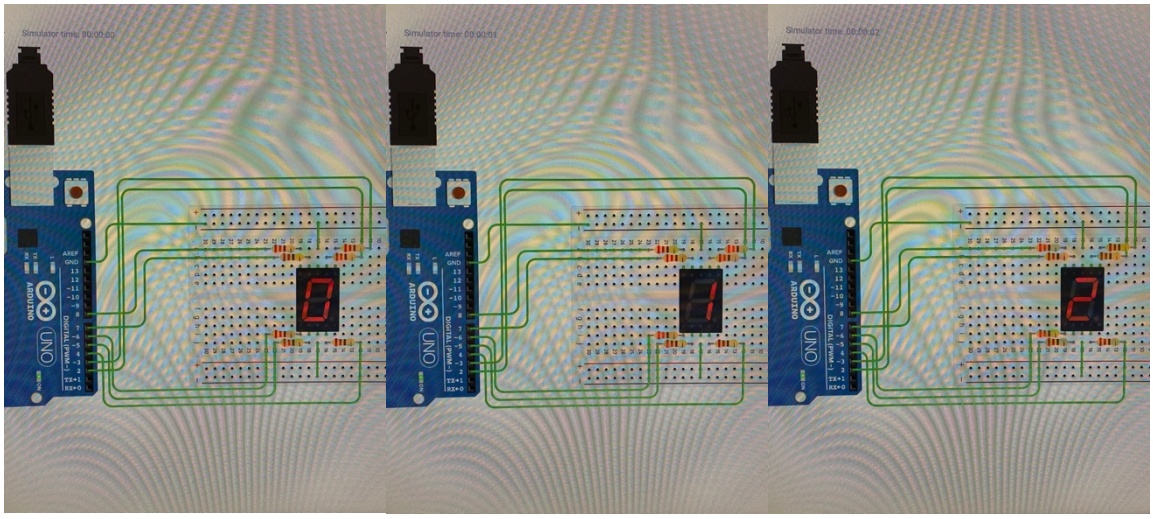
Diagram:

Similar to figure 2.4.3

Coding:

```
void setup()
{
        int a;
        for(a=2;a<9;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
         int a, b;
         int number_45[10][7] =
        {{1,1,1,1,1,1,0},{0,1,1,0,0,0,0},{1,1,0,1,1,0,1},{1,1,1,1,0,0,1},{0,1,1,0,0,1,1},{1,0,1,
        1,0,1,1},{1,0,1,1,1,1,1},{1,1,1,0,0,0,0},{1,1,1,1,1,1,1},{1,1,1,1,0,1,1}};
          for(a=0;a<10;a++)
        {
                 for(b=2;b<9;b++)
                 {
                         digitalWrite(b ,number_45[a][b-2]);
                }
                  delay(1000);
        }
}
```

Result:



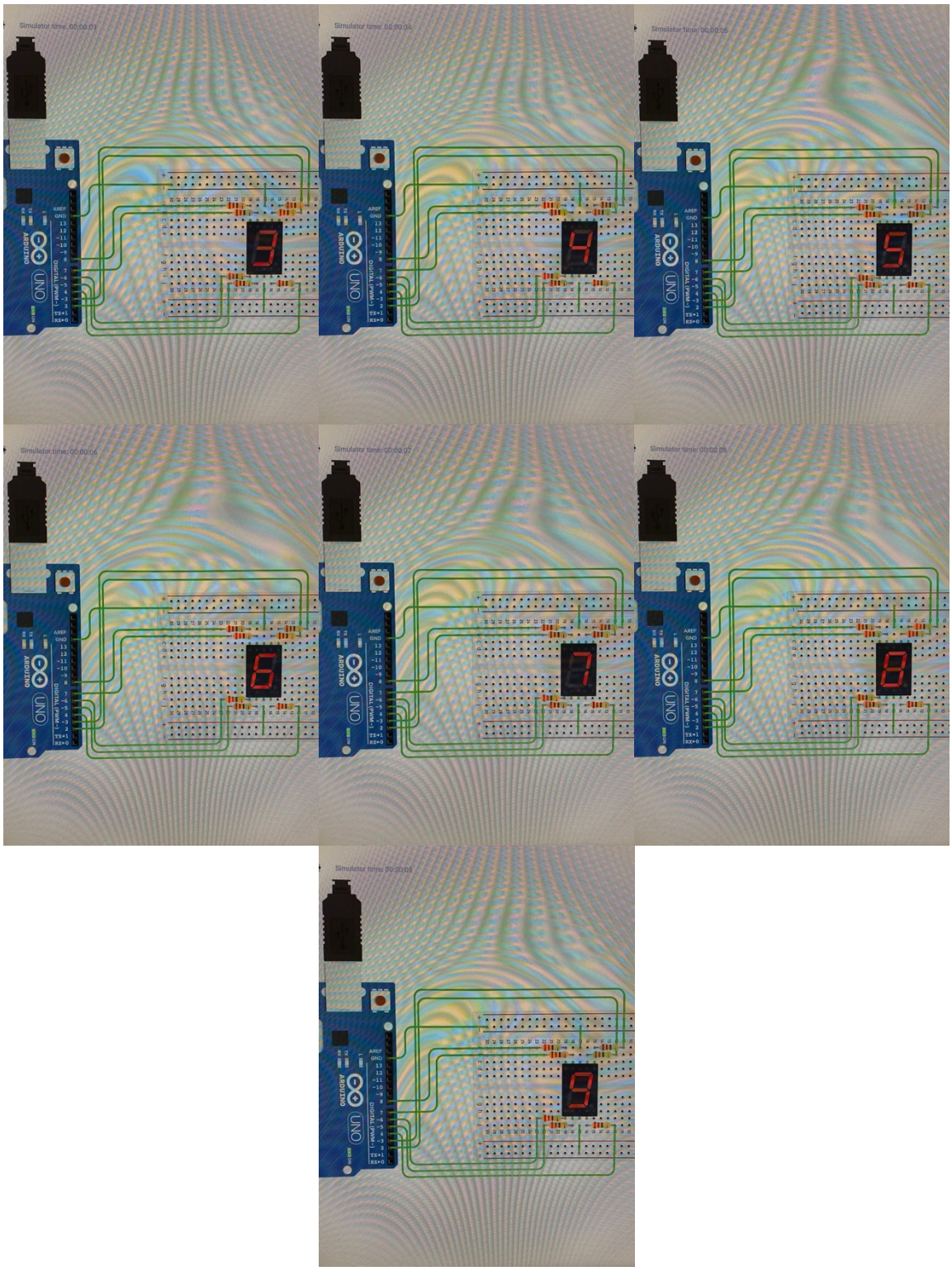


Figure 2.5.1 – 10: 7-segments runs from 0-9 repeatedly

Ex 2.6:

- Make Led on 7 segment start at ‘a’ followed by ‘b’, and so on until ‘g’ with 1 s delay.
- When it reaches ‘g’ ,reverses as follows ‘f’ so on back to ‘a’ with 1 s delay.

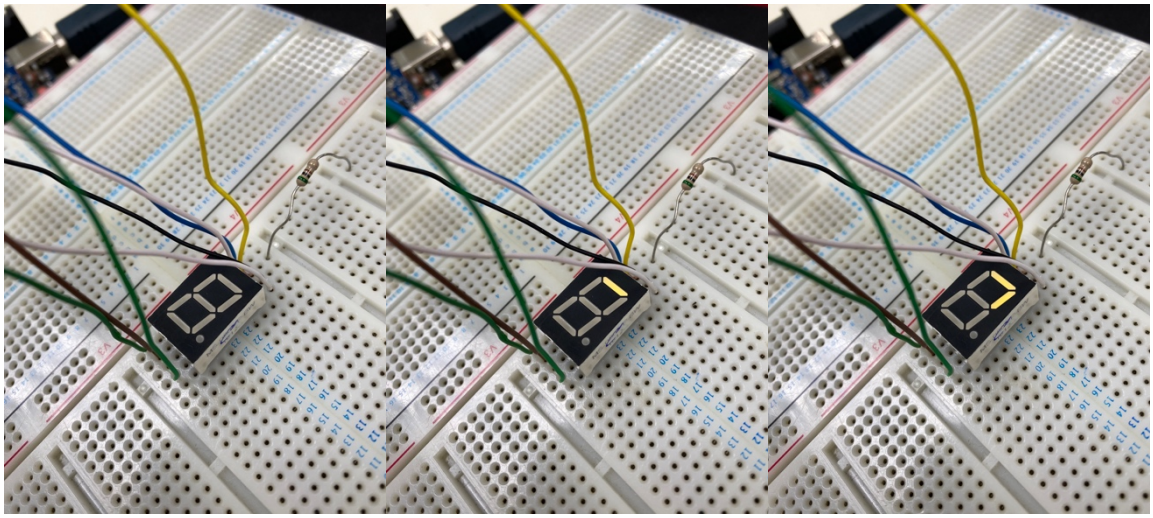
Diagram:

Similar to figure 2.4.3

Coding:

```
void setup()
  {
        int a;
        for(a=2;a<9;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        int a;
        for(a=2;a<9;a++)
        {
                digitalWrite(a, HIGH);
                delay(1000);
        }
        for(a=8;a>1;a--)
        {
                digitalWrite(a, LOW);
                delay(1000);
         }
}
```

Result:



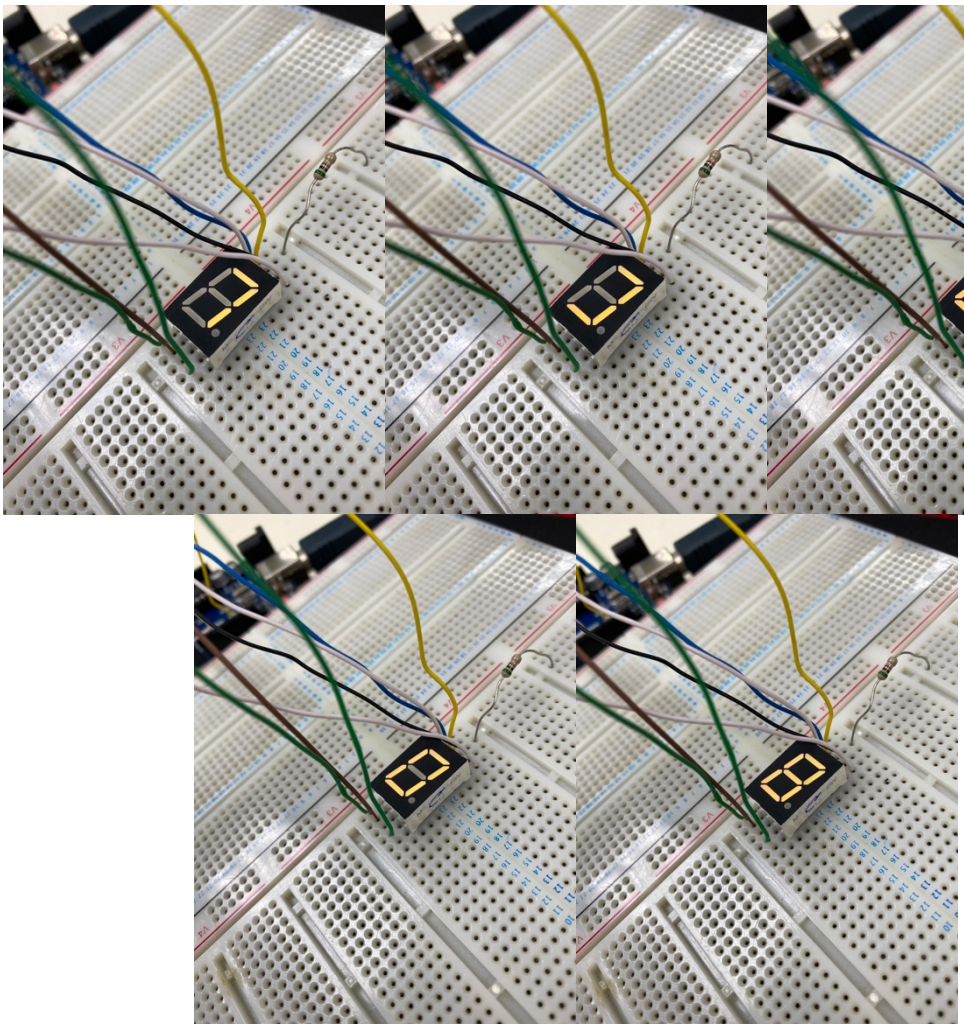


Figure 2.6.1 – 8: LED On from a to g

Ex 2.7: Use DDRx, PORTx registers

- 7 Segment running 0 – 9 and repeat with delay of 1 second

Diagram:

Similar to figure 2.4.3

Coding:

```
void setup()
{
	DDRD = 0xFC; //0b11111100
	DDRB = 0x01;
	PORTB = 0x01;
}

void loop()
{
	PORTD = 0x3F<<2;
	PORTB = 0x00;
	delay(1000);
	PORTD = 0x06<<2;
	PORTB = 0x00;
	delay(1000);
	 PORTD = 0x5B<<2;
	PORTB = 0x01;
	delay(1000);
	 PORTD = 0x4F<<2;
	PORTB = 0x01;
	 delay(1000);
	 PORTD = 0x66<<2;
	 PORTB = 0x01;
	 delay(1000);
	PORTD = 0x6D<<2;
	PORTB = 0x01;
	delay(1000);
	PORTD = 0x7D<<2;
	PORTB = 0x01;
	delay(1000);
	PORTD = 0x07<<2;
	PORTB = 0x00;
	delay(1000);
	PORTD = 0x7F<<2;
	PORTB = 0x01;
	delay(1000);
	PORTD = 0x6F<<2;
	PORTB = 0x01;
	delay(1000);
}
```

Result:

Similar to figure 2.5.1 – 10

Ex 2.8: Any method (DDRx or pinMode, Portx or digitalWrite)

- **CA** 7 Segment running 0 – 9 and repeat with delay of 1 second

Diagram:

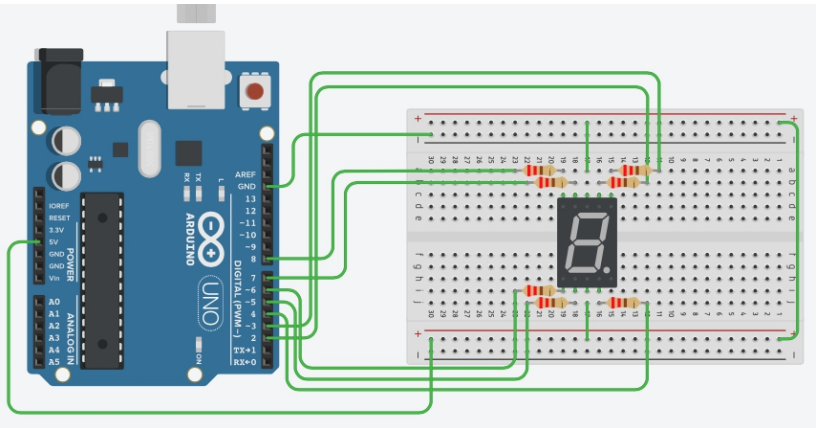


Figure 2.8.1

Coding:

```
void setup()
{
	DDRD = 0xFC; //0b11111100 // make input/output // FC is coversion of binary to hex

	DDRB = 0x01;
	PORTB = 0x01;
}

void loop()
{
	PORTD = ~(0x3F<<2);
	PORTB = 0x01;
	delay(1000);
	PORTD = ~(0x06<<2);
	PORTB = 0x01;
	delay(1000);
	PORTD = ~(0x5B<<2);
	PORTB = 0x00;
	delay(1000);
	PORTD = ~(0x4F<<2);
	PORTB = 0x00;
	delay(1000);
```

```
        PORTD = ~(0x66<<2);
        PORTB = 0x00;
        delay(1000);
        PORTD = ~(0x6D<<2);
        PORTB = 0x00;
        delay(1000);
        PORTD = ~(0x7D<<2);
        PORTB = 0x00;
        delay(1000);
        PORTD = ~(0x07<<2);
        PORTB = 0x01;
        delay(1000);
        PORTD = ~(0x7F<<2);
        PORTB = 0x00;
        delay(1000);
        PORTD = ~0x6F<<2;
        PORTB = 0x00;
        delay(1000);
}
```

Result:

Similar to figure 2.5.1 – 10

Conclusion:

In chapter two, a digital output of Arduino was introduced. It is simply to connect Arduino with external digital output devices such as LEDs and 7- segments. There are two kind of LED connections: Active High (AH) and Active Low (AL). AH connects from a digital output pin to the ground so it can operate when the output is high, where AL connects from 5V pin to a digital output pin so it can operate oppositely with AH (Low). The same concept goes to 7- segments as well with a different name: Common Cathode (AH) and Common Anode (AL). Furthermore, an alternative coding with DDRx and PORTx were introduced along this chapter so as to allow further communication with other devices in a future use.

Chapter 3: Digital Input

Ex 3.1: : Active high connection switch

- When Press Switch at Input (Pin 2)
- Active high LED (Pin 13) turn On
- When Press is not Switch at Input (Pin 2)
- Active high LED (Pin 13) turn Off

Diagram:

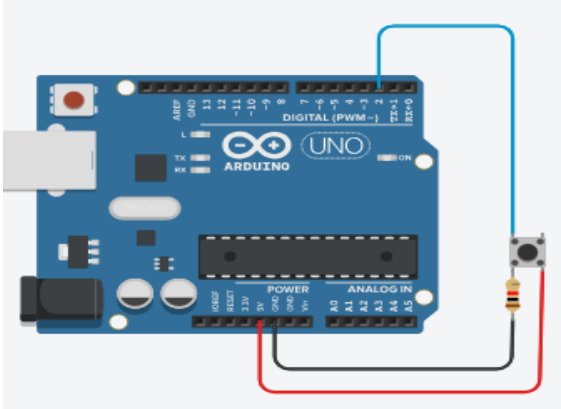


Figure 3.1.1

Coding:

```
void setup()
{
	pinMode(2, INPUT);
	pinMode(13, OUTPUT);
}

void loop()
{
	bool status;
	status = digitalRead(2);
	 if (status==1)
	{
		digitalWrite(13, HIGH);
	}
	else
	{
		digitalWrite(13, LOW);
	}
}
```

Result:

Note: An onboard LED (Pin 13) was used.

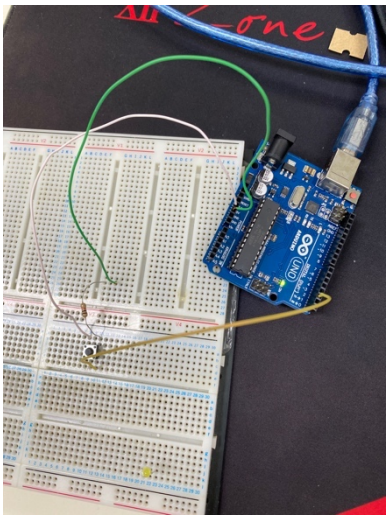


Figure 3.1.2: Unpressed

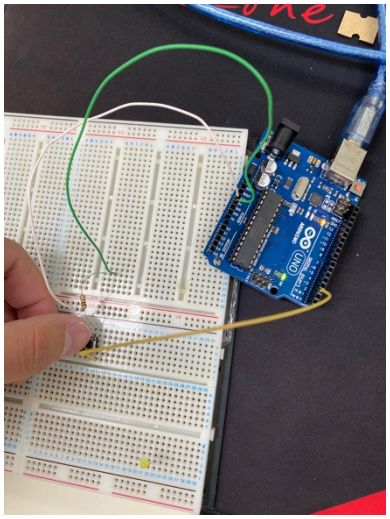


Figure 3.1.3: Pressed

Ex 3.2: Active low connection switch

- When Press Switch at Input (Pin 2)
  - Active high LED (Pin 13) turn On
- When Press is not Switch at Input (Pin 2)
  - Active high LED (Pin 13) turn Off

Diagram:

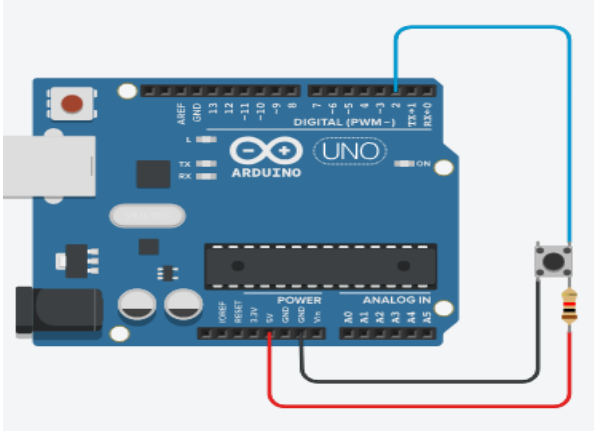


Figure 3.2.1

Coding:

```
void setup()
{
	pinMode(2, INPUT);
	pinMode(13, OUTPUT);
}

void loop()
{
	bool status;
	status = digitalRead(2);
	if (status==1)
	{
		digitalWrite(13, LOW);
	}
	else
	{
		digitalWrite(13, HIGH);
	}
}
```

Result:

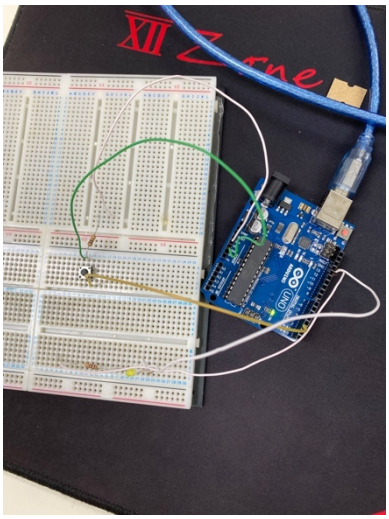


Figure 3.2.2: Unpressed

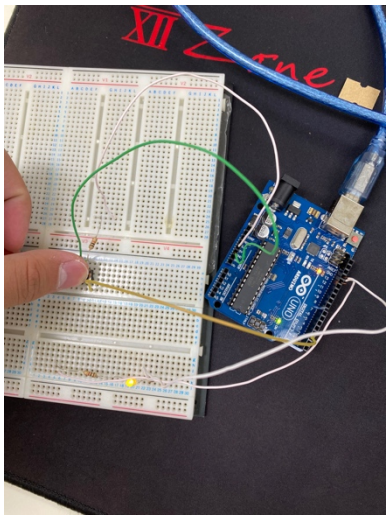


Figure 3.2.3: Pressed

Ex 3.3: Active High connection switch

- Press SW1 (pin 2) : Active high LED1(Pin8) On
  Unpress SW1 : Active high LED1(Pin8) Off
- Press SW2 (pin 3) : Active high LED2(Pin9) On
  Unpress SW2 : Active high LED2(Pin9) Off
- Press SW3 (pin 4) : Active high LED3(Pin10) On
  Unpress SW3 : Active high LED3(Pin10) Off
- Press SW4 (pin 5) : Active high LED4(Pin11) On
  Unpress SW4 : Active high LED4(Pin8) Off

Diagram:

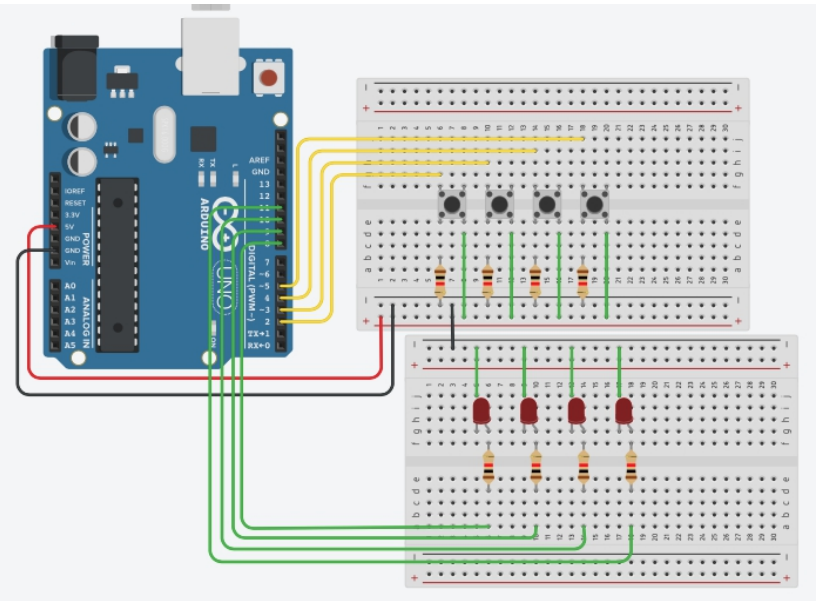


Figure 3.3.1

Coding:

```
void setup()
{
        int a;
        for(a=2;a<6;a++)
        {
        pinMode(a, INPUT);
        }
        for(a=8;a<12;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        int a;
        bool status5,status2,status3,status4;
        status2 = digitalRead(2);
```

```
        status3 = digitalRead(3);
        status4 = digitalRead(4);
        status5 = digitalRead(5);
        if (status2==1)
        {
                digitalWrite(8, HIGH);
        }
        else
        {
                digitalWrite(8, LOW);
        }
        if (status3==1)
         {
                digitalWrite(9, HIGH);
        }
        else
        {
                digitalWrite(9, LOW);
        }
        if (status4==1)
        {
                 digitalWrite(10, HIGH);
        }
        else
        {
                 digitalWrite(10, LOW);
        }
        if (status5==1)
        {
                digitalWrite(11, HIGH);
        }
         else
        {
                digitalWrite(11, LOW);
        }
}
```

Result:

Figure 3.3.2 – 6: Each LED turned ON when pressed

Ex 3.3 (Extra): Active High connection switch
*Use DDRx, PORTx, and PINx

- Press SW1 (pin 2) : Active high LED1(Pin8) On
  Un Press SW1 : Active high LED1(Pin8) Off
- Press SW2 (pin 3) : Active high LED2(Pin9) On
  Un Press SW2 : Active high LED2(Pin9) Off
- Press SW3 (pin 4) : Active high LED3(Pin10) On
  Un Press SW3 : Active high LED3(Pin10) Off
- Press SW4 (pin 5) : Active high LED4(Pin11) On
  Un Press SW4 : Active high LED4(Pin8) Off

Diagram:

Similar to figure 3.3.1

Coding:

```
void setup()
{
	DDRD &= ~(0x3C); //as input/ or (C3) no inverse but & is needed to force 0
	DDRB |= 0x0F; //as output
}

void loop()
{
	bool SW1 = PIND & 0x04;
	bool SW2 = PIND & 0x08;
	bool SW3 = PIND & 0x10;
	bool SW4 = PIND & 0x20;
	if(SW1==1)
	{
		PORTB |= 0x01;
	 }
	 else
	 {
		PORTB &= ~(0x01);
	}
	if(SW2==1)
	{
		PORTB |= 0x02;
	}
	else
	{
		PORTB &= ~(0x02);
	}
	if(SW3==1)
	{
		PORTB |= 0x04;
	}
	else
	{
		PORTB &= ~(0x04);
	}
	if(SW4==1)
	{
		PORTB |= 0x08;
	}
	else
	{
		PORTB &= ~(0x08);
	}
}
```

Result:

Similar to figure 3.3.2 - 6

Ex 3.4: Active High connection DIP SW
*Use DDRx, PORTx, and PINx
SW1 = PIN2, SW2 = PIN3, SW3 = PIN4, SW4 = PIN5

- Toggle SW1 : AH-LED1 (Pin8 only) On
- Toggle SW12 : AH-LED2 (Pin9 only) On
- Toggle SW123 : AH-LED3 (Pin10 only) On
- Toggle SW1234 : AH-LED4 (Pin11 only) On
- Other conditions : AH-ALL LEDs Off

Diagram:

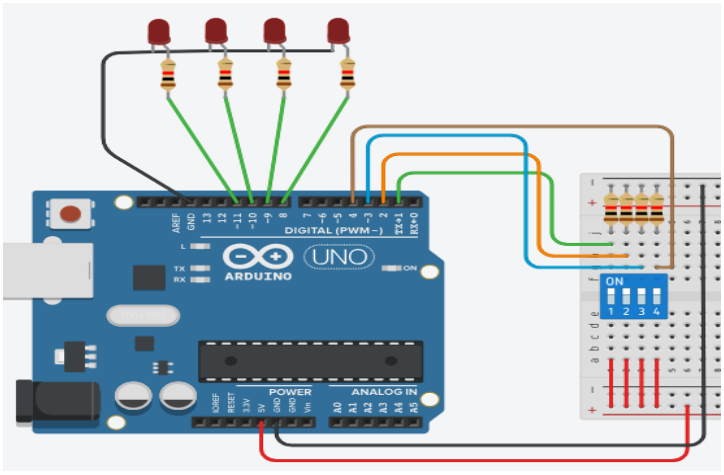


Figure 3.4.1

Coding:

```
void setup()
{
	DDRD &= ~(0x1E); //as input/ or (C3) no inverse but & is needed to force 0
	DDRB |= 0x0F; //as output
}

void loop()
{
	bool SW1 = PIND & 0x02;
	bool SW2 = PIND & 0x04;
	bool SW3 = PIND & 0x08;
	bool SW4 = PIND & 0x10;
	if(SW1==1 && SW2==0 && SW3==0 && SW4==0)
	{
		PORTB = 0x01;
	}
	if(SW1==1 && SW2==1 && SW3==0 && SW4==0)
	{
		PORTB = 0x02;
	}
	if(SW1==1 && SW2==1 && SW3==1 && SW4==0)
	{
```

```
                PORTB = 0x04;
        }
        if(SW1==1 && SW2==1 && SW3==1 && SW4==1)
        {
                PORTB = 0x08;
        }
        else
        {
                PORTB = 0x00;
        }
}
```

Result:

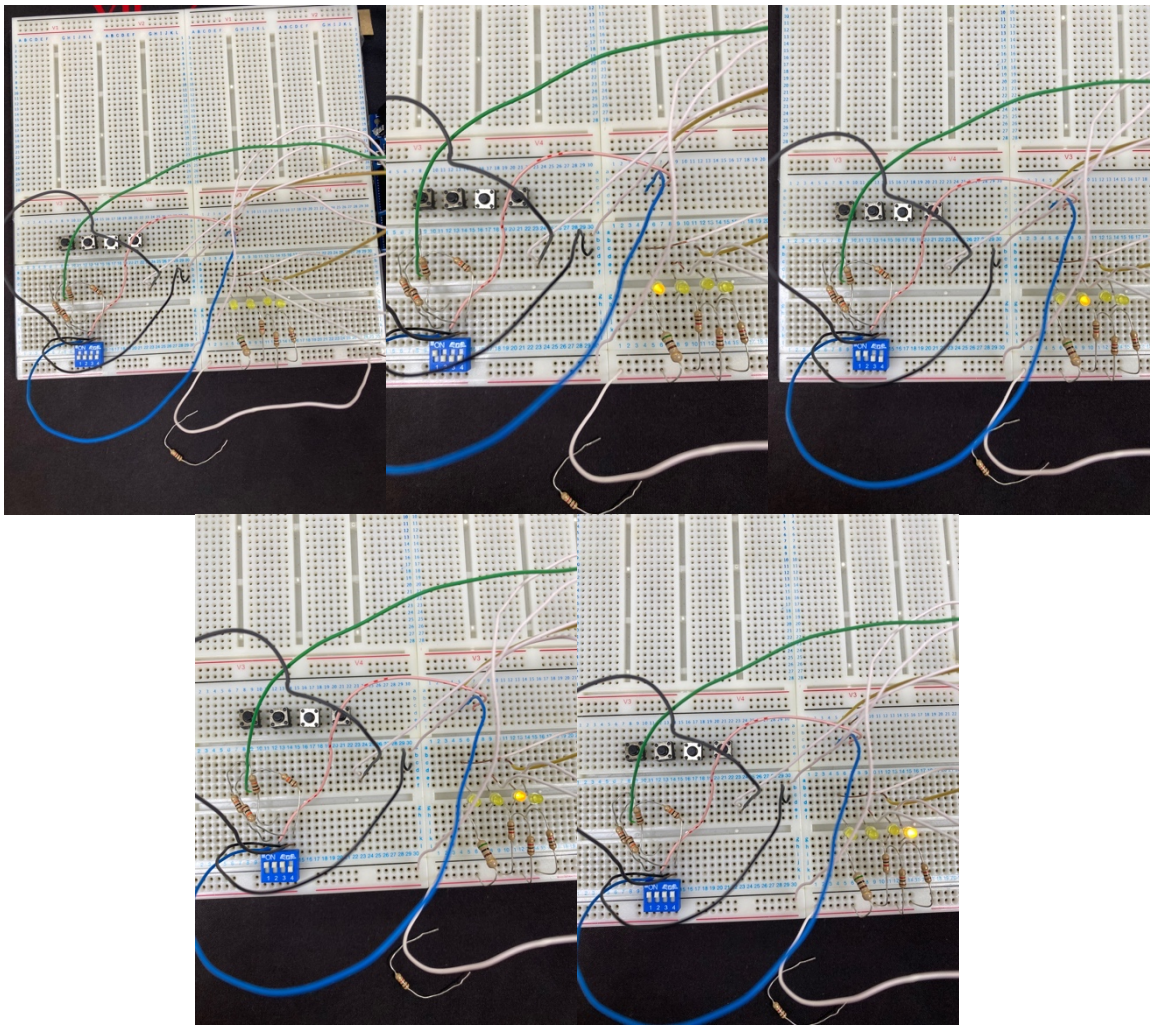


Figure 3.4.2 – 6: Each LED turned ON when more buttons were pushed.

Conclusion:

In chapter three, a digital input of Arduino was introduced. A digital input is a device that can obtain input from physical action and direct a signal (as o or 1) into Arduino board in order to trigger some further actions to the output. For instance, switches (pushbutton switch and DIP switch). The chapter includes a constant use of if-else function to cause change in output as input is received. There are also 2 kind of connections: Active High and Low similarly to LEDs. AH attaches a digital input pin with the ground line. When the switch is pushed, it allows the current to flow from 5V. AL works on the other way round, by being attached to the 5V line. When the switch is pushed, the current flows to ground.

Chapter 4: Serial Communication

Ex 4.1: Reading inputs data with serial monitor

- Connect AH-DIP SW as inputs (Port B pin 8 - 11)
- Reading All Bit data through serial communication

Diagram:

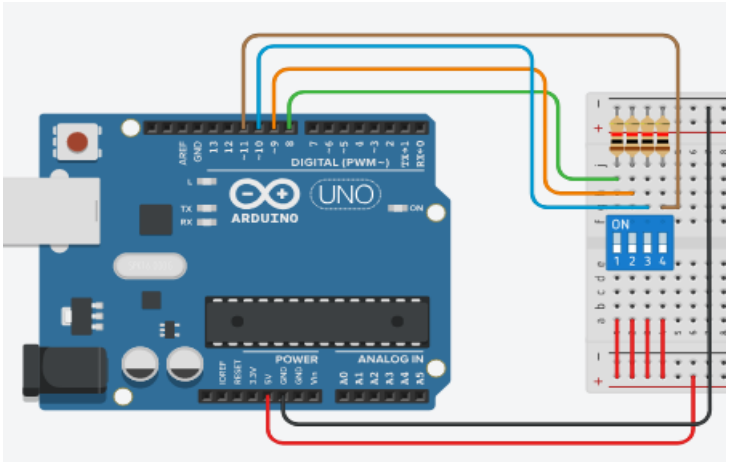
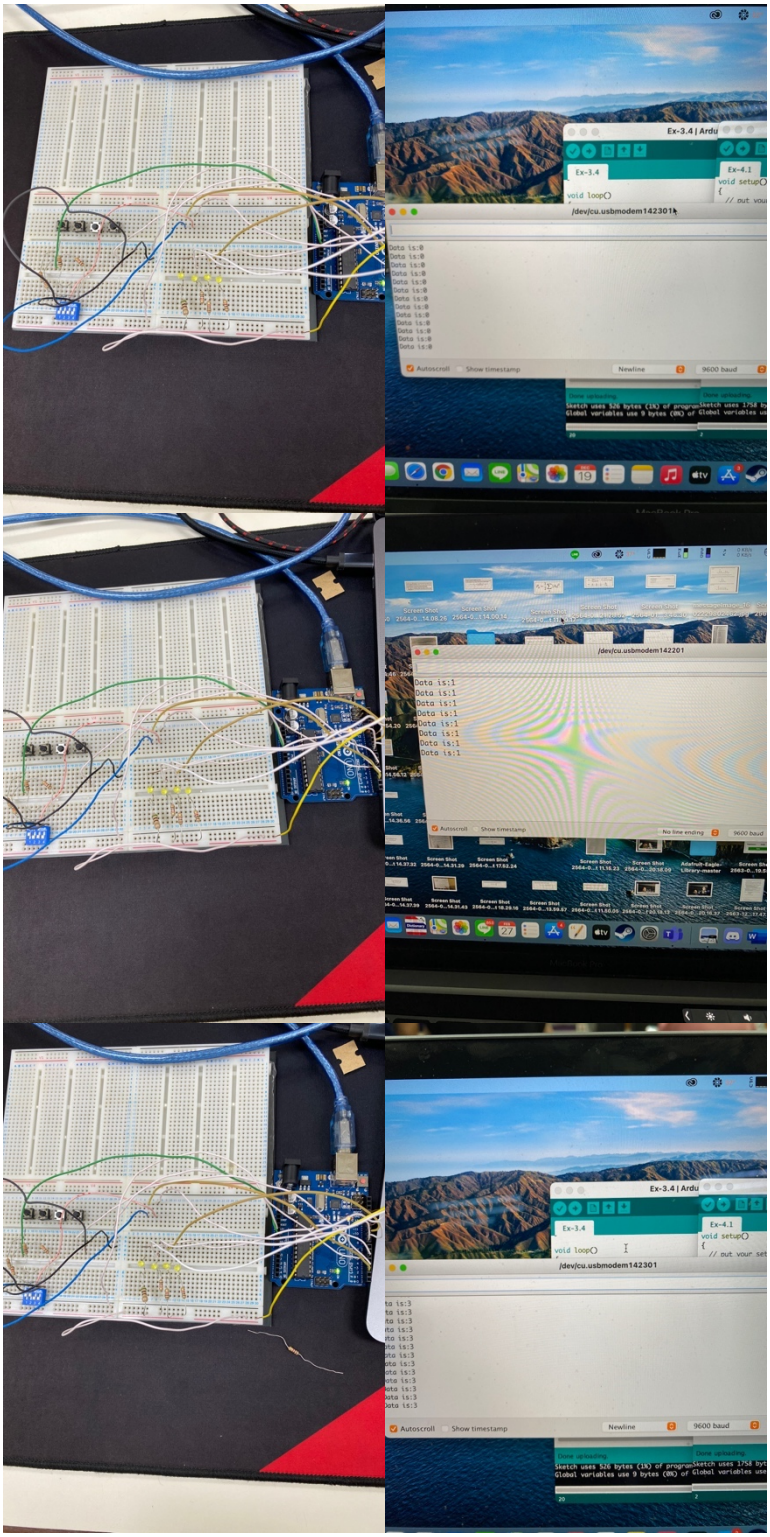


Figure 4.1.1

Coding:

```
void setup()
{
	DDRD = ~(0x3C);
	Serial.begin(9600);
}

void loop()
{
	int data = (PIND)>>2; //as pin 0 and 1 are not being used.
	Serial.print("Data is:");
	Serial.println(data, HEX);
	delay(1000);
}
```

Result:



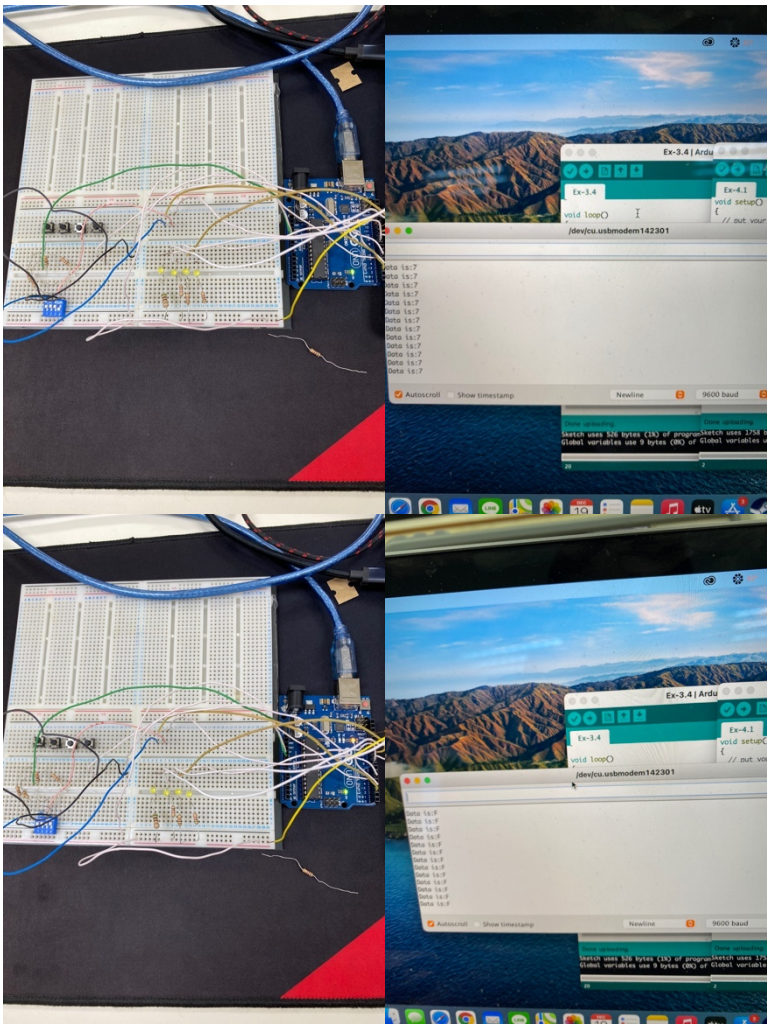


Figure 4.1.2 – 11: Switch Positions and Serial Monitor Outputs

Ex 4.2: Read data from KB

- Reading via Serial Communication
- Show Character serial input

Diagram:

No Diagram provided

Coding:

```
void setup()
{
	Serial.begin(9600);
}

void loop()
{
	if(Serial.available()>0)
	{
		char input;
		input = Serial.read();
		Serial.print("I received: ");
		Serial.println(input);
	}
}
```

Result:

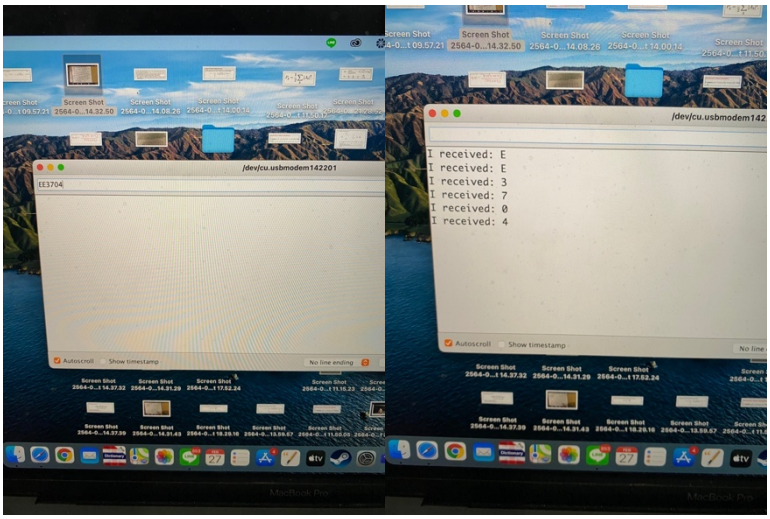


Figure 4.2.1 – 2: Serial Monitor Outputs

Ex 4.3: Reading data from KB and control LEDs (use switch-case)
LEDs are connected at Pins 8,9,10,11

- Press "1" from keyboard AH-LED1 ON (only) show "LED1 is on" in monitor
- Press "2" from keyboard AH-LED2 ON (only) show "LED2 is on" in monitor
- Press "3" from keyboard AH-LED3 ON (only) show "LED3 is on" in monitor
- Press "4" from keyboard AH-LED4 ON (only) show "LED4 is on" in monitor
- Press other characters, Reset all AH-LED to OFF

Show your circuit diagram, coding and results

Diagram:

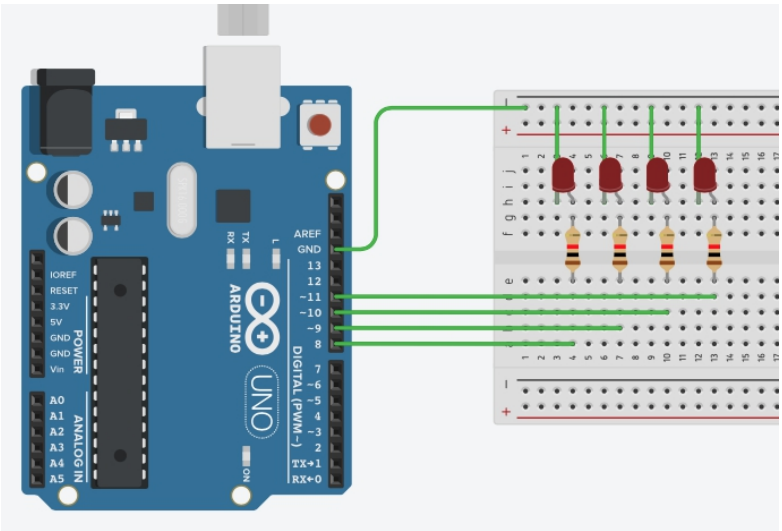


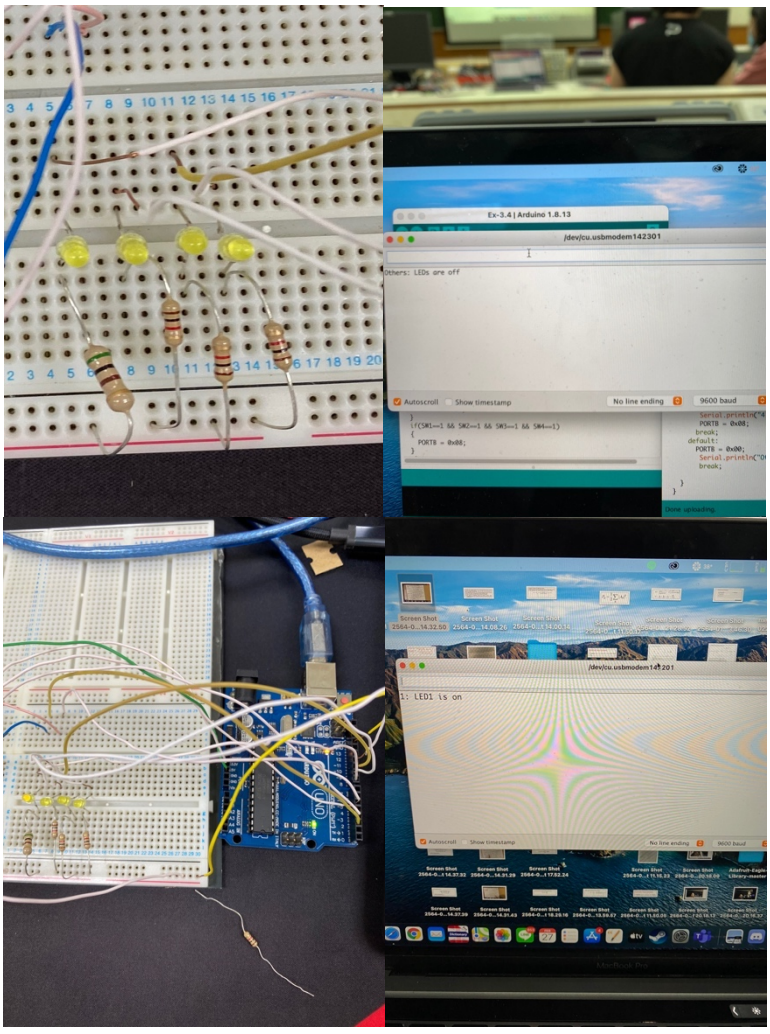
Figure 4.3.1

Coding:

```
void setup()
{
	DDRD = ~(0x3C);
	Serial.begin(9600);
}

void loop()
{
	if(Serial.available()>0)
	{
		char input;
		input = Serial.read();
		switch(input)
		{
			case '1':
				 Serial.println("1: LED1 is on");
				PORTB = 0x01;
				break;
			case '2':
				Serial.println("2: LED2 is on");
				PORTB = 0x02;
```

```
                    break;
                case '3':
                    Serial.println("3: LED3 is on");
                    PORTB = 0x04;
                    break;
                case '4':
                    Serial.println("4: LED4 is on");
                    PORTB = 0x08;
                     break;
                default:
                    PORTB = 0x00;
                    Serial.println("Others: LEDs are off");
                    break;
            }
        }
}
```

Result:



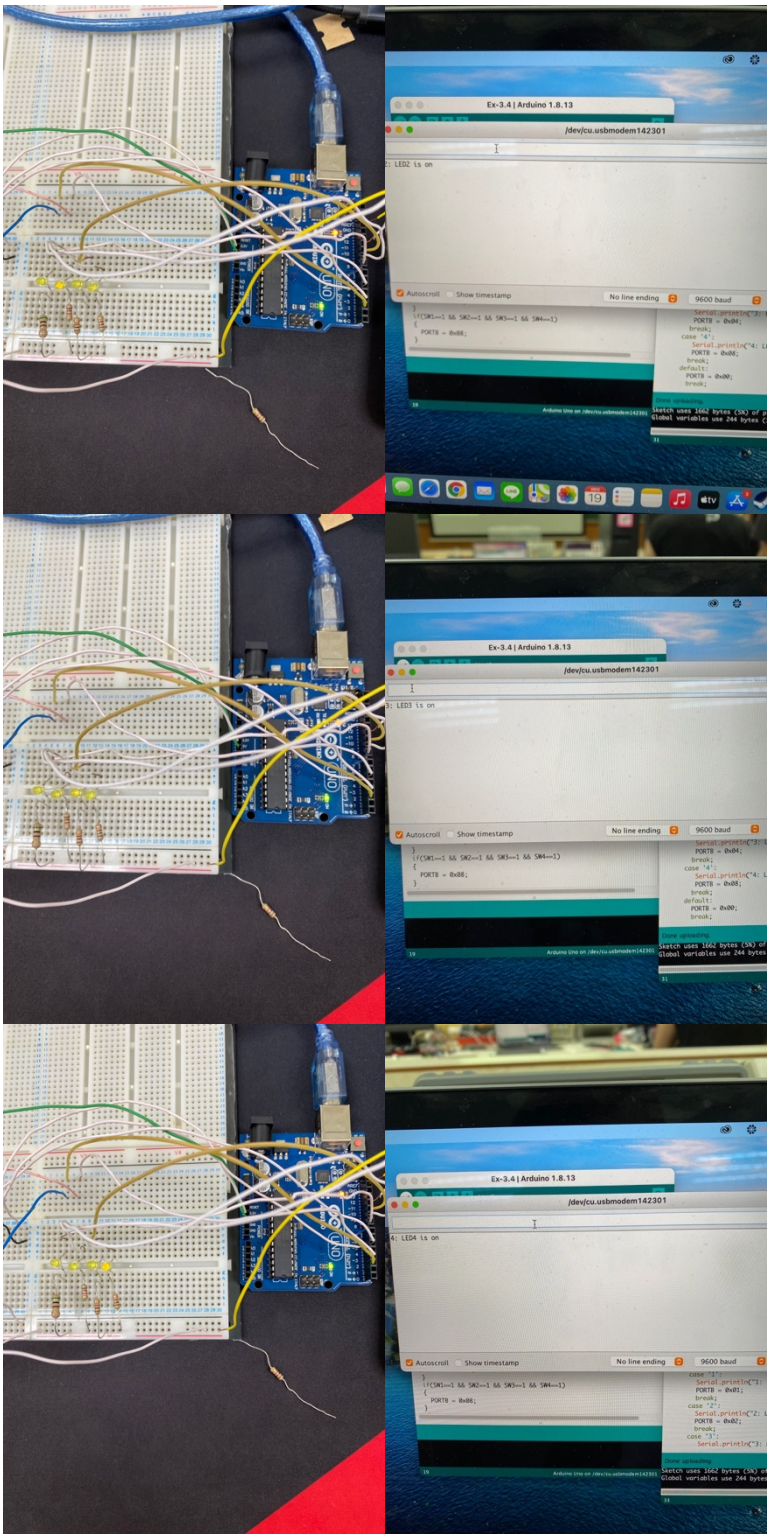


Figure 4.3.2 – 13: LED Statuses and Serial Monitor Outputs

Ex 4.4: Use Buzzer/Speaker as output to create music note

- Press “d” from keyboard produce “Do”
- Press “r” from keyboard produce “Re”
- Press “m” from keyboard produce “Mi”
- Press “f” from keyboard produce “Fa”

- Press “z” from keyboard produce “Zol”
- Press “l” from keyboard produce “La”
- Press “t” from keyboard produce “Ti”

Diagram:

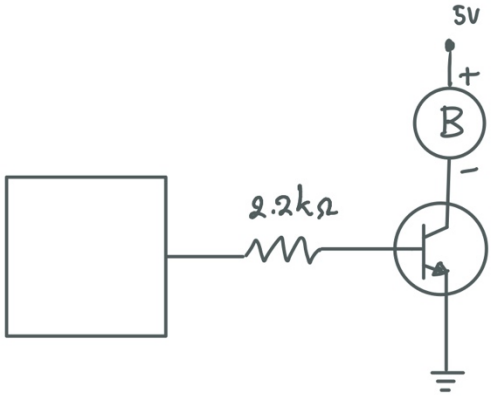


Figure 4.4.1

Coding:

```
void NOTE_DO()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(478);
		digitalWrite(6,LOW);
		delayMicroseconds(478);
	}
	Serial.println("DO");
	delay(5);
}

void NOTE_RE()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(426);
		digitalWrite(6,LOW);
		delayMicroseconds(426);
	}
	Serial.println("RE");
	delay(5);
}

void NOTE_ME()
```

```
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(379);
		digitalWrite(6,LOW);
		delayMicroseconds(379);
	}
	Serial.println("ME");
	delay(5);
}

void NOTE_FA()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(358);
		digitalWrite(6,LOW);
		delayMicroseconds(358);
	}
	Serial.println("FA");
	delay(5);
}

void NOTE_SO()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(319);
		digitalWrite(6,LOW);
		delayMicroseconds(319);
	}
	Serial.println("SO");
	delay(5);
}

void NOTE_LA()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(284);
		digitalWrite(6,LOW);
		delayMicroseconds(284);
	}
	Serial.println("LA");
	delay(5);
}
```

```
void NOTE_TI()
{
	for(int a=0;a<500;a++)
	{
		digitalWrite(6,HIGH);
		delayMicroseconds(253);
		digitalWrite(6,LOW);
		delayMicroseconds(253);
	}
	Serial.println("TI");
	delay(5);
}

void setup()
{
	Serial.begin(9600);
	pinMode(6, OUTPUT);
}

void loop()
{
	if(Serial.available()>0)
	{
		char  input = Serial.read();
		switch(input)
		{
			case 'd':
				NOTE_DO();
				break;
			case 'r':
				NOTE_RE();
				 break;
			case 'm':
				NOTE_ME();
				break;
			case 'f':
				NOTE_FA();
				break;
			case 's':
				NOTE_SO();
				break;
			case 'l':
				NOTE_LA();
				break;
			case 't':
				NOTE_TI();
				break;
		}
	}
}
```

Result:

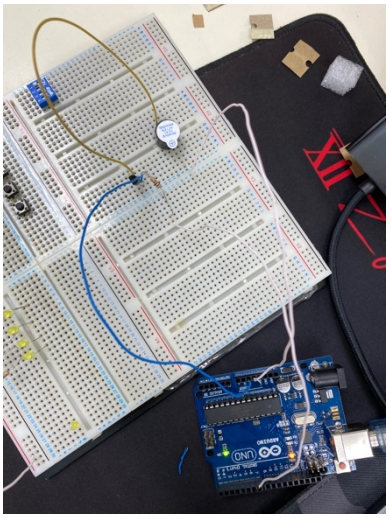


Figure 4.4.2: Buzzer Connection

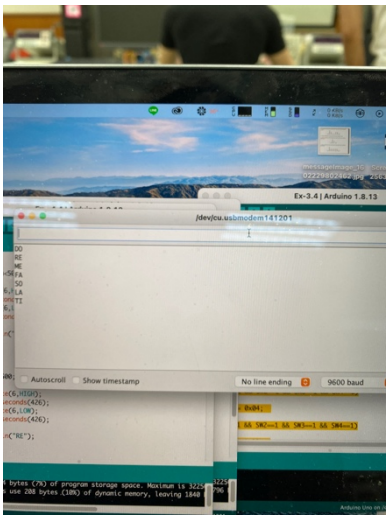


Figure 4.4.3: Serial Monitor Outputs

Ex 4.5: Random LED game when SW is pressed
**LED pins 2-9, Buzzer 10, SW pin 11;

- Make 8 LEDs (AL) go incrementally with fastest speed (example 10ms)
- When a SW is Pressed, a random number is calculated and show on a LED randomly, a buzzer sound is audible for 1s
- LED is on for 3s and go off
- Loop is keep repeating

Diagram:

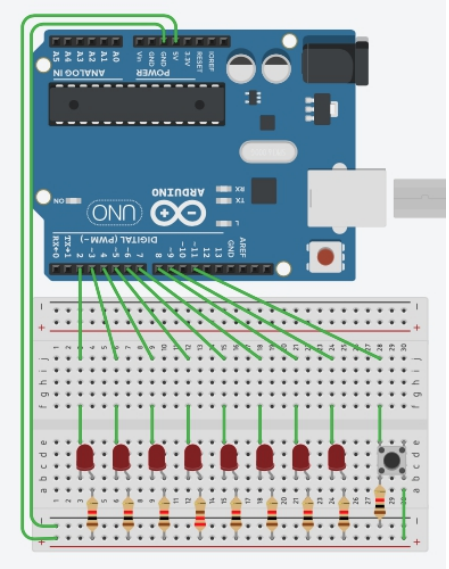


Figure 4.5.1: LEDs and Switch connection

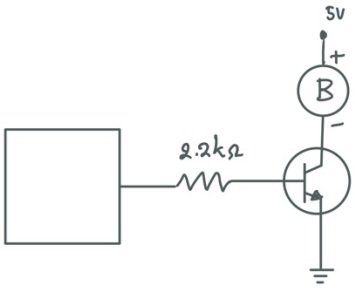


Figure 4.5.2: Buzzer Connection

Note: Buzzer is not presented in Tinker cad!

Coding:

```
void setup()
{
	for(int a=2;a<11;a++)
	{
		pinMode(a, OUTPUT);
	}
	 pinMode(11, INPUT);
}

void loop()
{
	bool SW = digitalRead(11);
	if(SW==1)
	{
		int number = random(2,10); //max exclusive, min inclusive
		digitalWrite(number,HIGH);
		digitalWrite(10, HIGH);
		delay(1000);
```

```
                digitalWrite(10, LOW);
                delay(2000);
                digitalWrite(number,LOW);
        }
        else
        {
                for(int pin=2;pin<10;pin++)
                {
                        digitalWrite(pin, HIGH);
                        delay(10);
                        digitalWrite(pin, LOW);
                        delay(10);
                }
        }
}
```

Result:

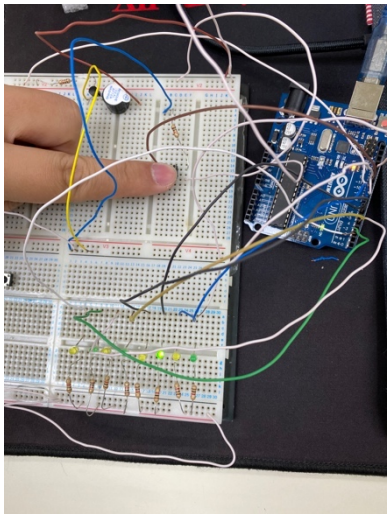


Figure 4.5.3: Random Output Resulted from Switch Press

Ex 4.6: Random LED game when SW is pressed (with higher random range)

- From Example 4.5, make random number between 0-100.
- Separate 0-100 range into 8 slots for LED with 1st LED of high odds, and 8th LED of low odds. Other LEDs can be your own choice.

Diagram:

Similar to figure 4.5.1 – 2

Coding:

```
void setup()
{
	for(int a=2;a<11;a++)
	{
		pinMode(a, OUTPUT);
	}
	pinMode(11, INPUT);
}

void loop()
{
	bool SW = digitalRead(11);
	if(SW==1)
	{
		int number = random(0,101); //max exclusive, min inclusive
		if(number<51 && number>-1)
		{
			digitalWrite(10, HIGH);
			digitalWrite(2, HIGH);
			delay(1000);
			digitalWrite(10, LOW);
			delay(2000);
			digitalWrite(2,LOW);
		}
		else if(number>50 && number<57)
		{
			digitalWrite(10, HIGH);
			digitalWrite(3, HIGH);
			delay(1000);
			digitalWrite(10, LOW);
			delay(2000);
			digitalWrite(3, LOW);
		}
		else if(number>55 && number<61)
		{
			digitalWrite(10, HIGH);
			digitalWrite(4, HIGH);
			delay(1000);
			digitalWrite(10, LOW);
```

```
                delay(2000);
                digitalWrite(4, LOW);
        }
        else if(number>60 && number<66)
        {
                digitalWrite(10, HIGH);
                digitalWrite(5, HIGH);
                delay(1000);
                digitalWrite(10, LOW);
                delay(2000);
                digitalWrite(5, LOW);
        }
        else if(number>65 && number<76)
        {
                digitalWrite(10, HIGH);
                digitalWrite(6, HIGH);
                delay(1000);
                digitalWrite(10, LOW);
                delay(2000);
                digitalWrite(6, LOW);
        }
        else if(number>75 && number<86)
        {
                digitalWrite(10, HIGH);
                digitalWrite(7, HIGH);
                delay(1000);
                digitalWrite(10, LOW);
                delay(2000);
                digitalWrite(7, LOW);
        }
        else if(number>85 && number<98)
        {
                digitalWrite(10, HIGH);
                digitalWrite(8, HIGH);
                delay(1000);
                 digitalWrite(10, LOW);
                delay(2000);
                 digitalWrite(8, LOW);
        }
         else if(number>97 && number<101)
        {
                 digitalWrite(10, HIGH);
                digitalWrite(9, HIGH);
                delay(1000);
                digitalWrite(10, LOW);
                 delay(2000);
                digitalWrite(9, LOW);
        }
}
else
```

```
        {
                for(int pin=2;pin<10;pin++)
                {
                        digitalWrite(pin, HIGH);
                        delay(10);
                        digitalWrite(pin, LOW);
                        delay(10);
                }
        }
}
```

Result:

Similar to the case of figure 4.5.3 but differs in probabilities of each LED.

Conclusion:

In chapter four, a serial communication of Arduino was introduced. It can be used to read a desired value as well as to input a value itself for triggering. The baud rate use was fixed to 9600 baud rate, and switch case function was being used for the ease of coding regarding serial communication. Additionally, a buzzer device was being introduced to generate a music note which as well introduced the use of sub void function.

Chapter 5: Analog Input

Ex 5.1: With Potentiometer

- Read the Value from Analog port (Pin A0)
- 0 to 5 V (10bit ADC)

Diagram:

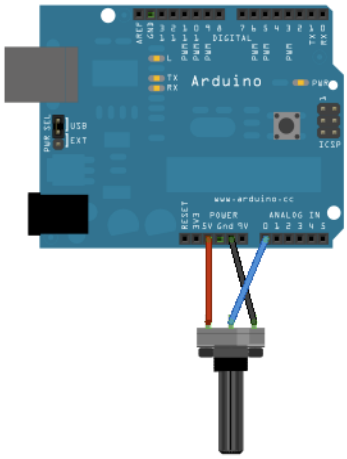


Figure 5.5.1

Coding:

```
void setup()
{
	//pinMode(A0, INPUT);
	Serial.begin(9600);
}

void loop()
{
	int AnalogValue;
	AnalogValue = analogRead(A0);
	Serial.println(AnalogValue);
	delay(1000);
}
```

Result:

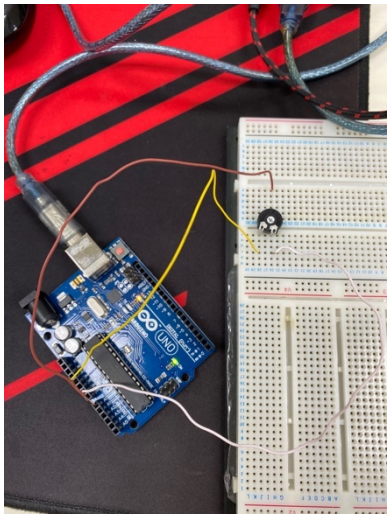


Figure 5.1.2: POT Connection

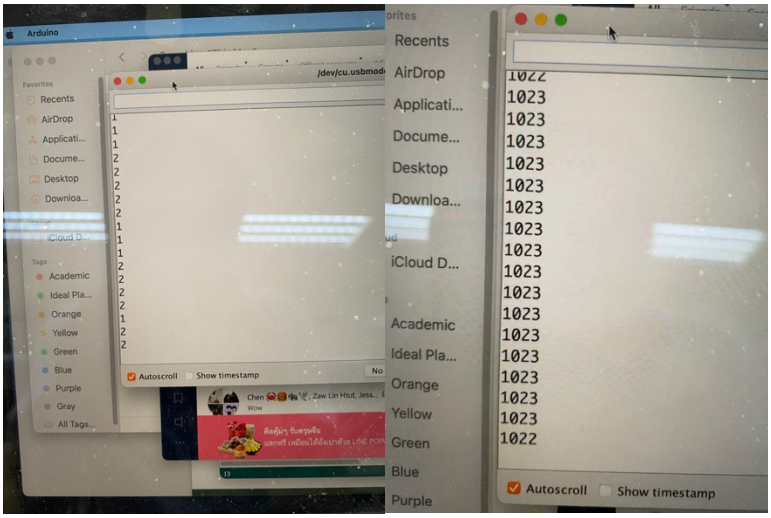


Figure 5.1.3 – 4: Serial Monitor Outputs When Reaching Maximum and Minumum

Ex 5.2: Perform LED operation when Analog data is as follow
Analog Input (A0)

- Value 0 - 255 : only AH-LED1 ON (Pin2)
- Value 256 - 511 : only AH-LED2 ON (Pin3)
- Value 512 - 767 : only AH-LED3 ON (Pin4)
- Value 768 - 1023 : only AH-LED4 ON (Pin5)

Diagram:

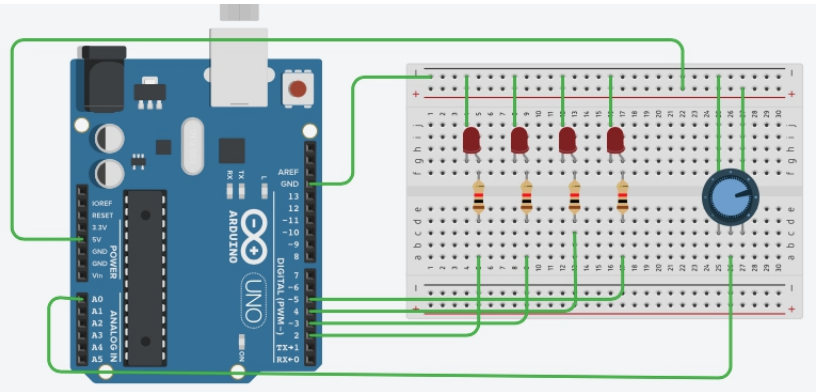


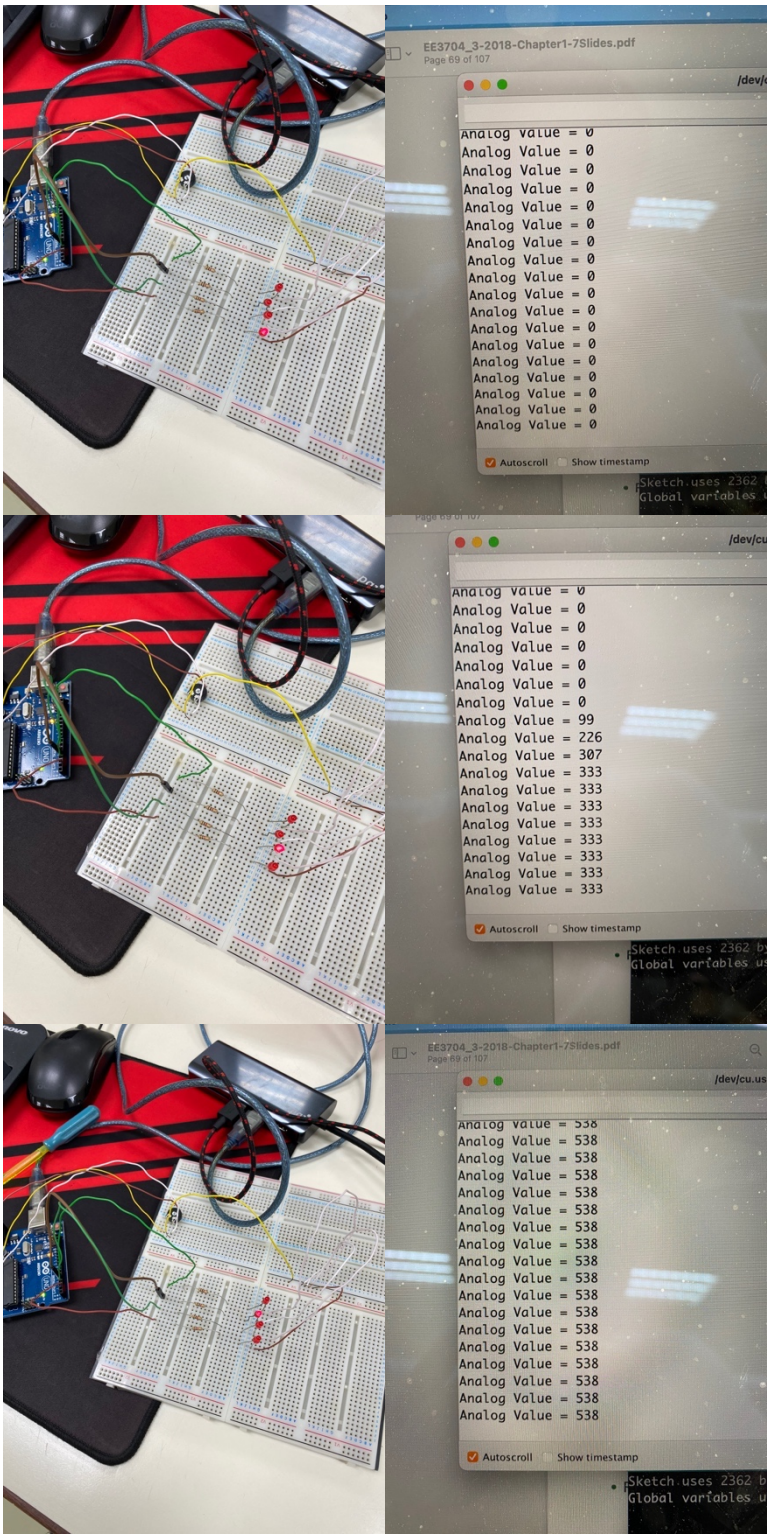
Figure 5.2.1

Coding:

```
void setup()
{
        //pinMode(A0, INPUT);
        int a;
        for(a=2;a<6;a++)
        {
                pinMode(a,OUTPUT);
        }
         Serial.begin(9600);
}

void loop()
{
        int AV;
        AV = analogRead(A0);
        Serial.print("Analog Value = ");
        Serial.println(AV);
        delay(1000);
        if(AV<256)
        {
                digitalWrite(2, HIGH);
                int b;
                for(b=3;b<6;b++)
```

```
                {
                        digitalWrite(b, LOW);
                }
        }
         else if(AV>255 && AV<512)
        {
                digitalWrite(3, HIGH);
                int b;
                for(b=4;b<6;b++)
                {
                        digitalWrite(b, LOW);
                }
                digitalWrite(2, LOW);
        }
        else if(AV>511 && AV<768)
        {
                digitalWrite(4, HIGH);
                int b;
                for(b=3;b>1;b--)
                {
                        digitalWrite(b, LOW);
                }
                digitalWrite(5, LOW);
        }
        else
        {
                digitalWrite(5, HIGH);
                int b;
                for(b=4;b>1;b--)
                {
                        digitalWrite(b, LOW);
                }
        }

}
```

Result:


EE3704_3-2018-Chapter1-7Slides.pdf
Page 69 of 107
Analog Value = 0
Autoscroll
Show timestamp
Analog Value = 99
Analog Value = 226
Analog Value = 307
Analog Value = 333
Analog Value = 538

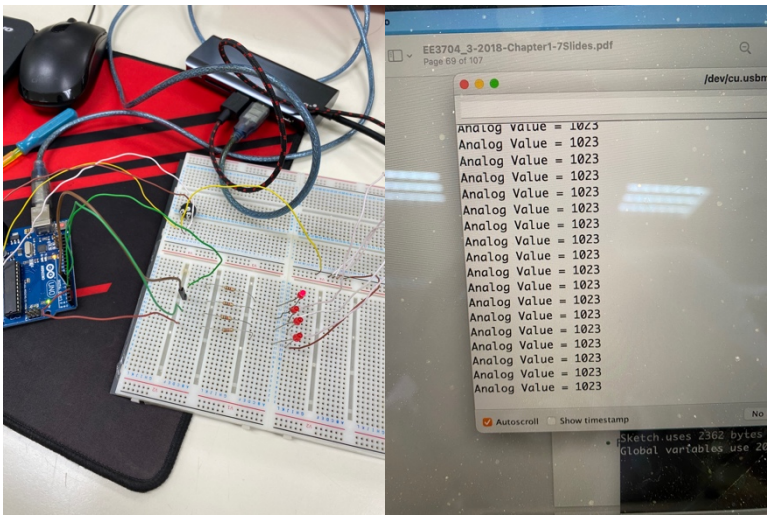


Figure 5.2.2 – 9: Digital Outputs and Serial Outputs

Ex 5.3: Perform LED operation when Analog data from LDR is

- LDR is at A0
  - Light is bright : (AH) LED1 ON
  - Light is moderate : (AH) LED1, LED2 ON
  - Light is dim : (AH) LED1, LED2 and LED 3 ON

*AL-LEDS: Pin 2,3,4

Diagram:

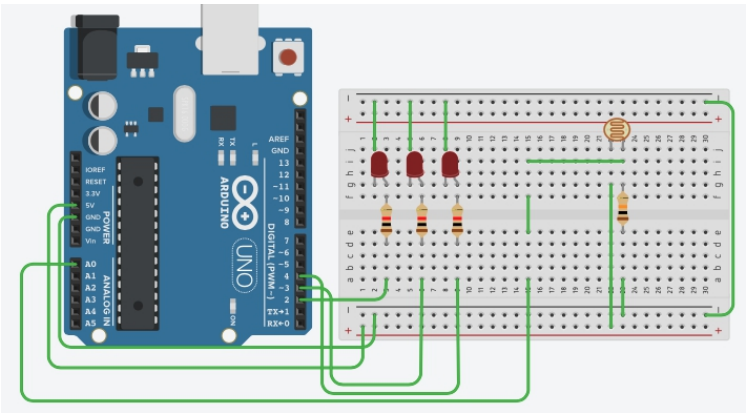


Figure 5.3.1

Coding:

```
void setup()
{
        for(int a=2;a<5;a++)
        {
```

```
                pinMode(a, OUTPUT);
        }
         // Full light >> 650
        // Partial light >> 440
        // Dim light >> 100
}

void loop()
{
        int AV;
        AV = analogRead(A0);
        Serial.print("Analog Value = ");
        Serial.println(AV);
        delay(100);
        if(AV>440 && AV<651)
        {
                 digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        }
        else if(AV>100 && AV<441)
        {
                digitalWrite(2, HIGH);
                digitalWrite(3, HIGH);
                digitalWrite(4, LOW);
        }
        else
        {
                int b;
                for(b=2;b<5;b++)
                {
                        digitalWrite(b, HIGH);
                }
         }
}
```

Result:

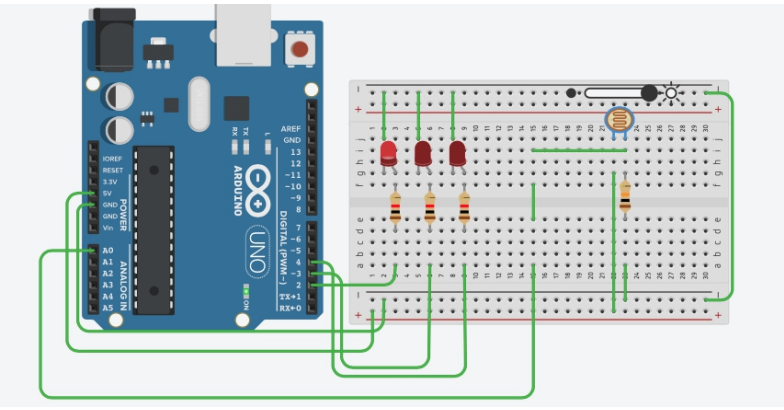


Figure 5.3.2: Full Light Condition

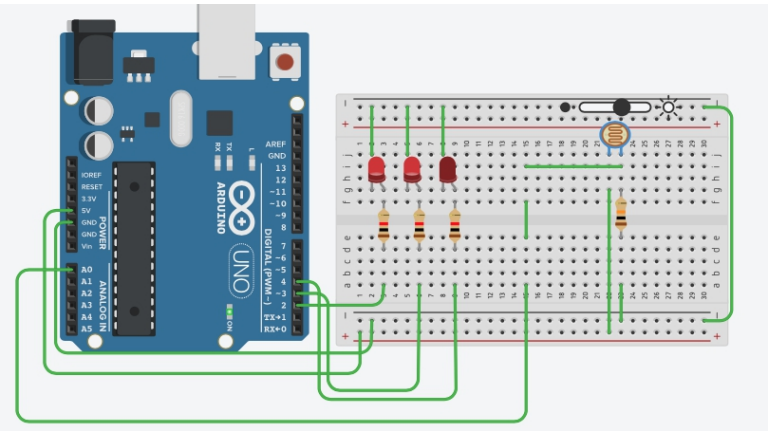


Figure 5.3.3: Moderate Light Condition

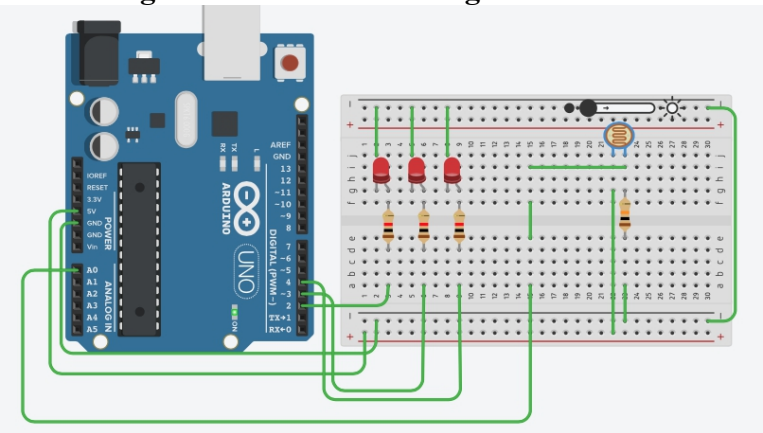


Figure 5.3.4: Dim to No Light Condition

Conclusion:

In chapter five, an analog input of Arduino was introduced. In this course, a potentiometer and LDR device were used. A potentiometer is a device that can vary it resistance value (0 -1023 Ohms) by the rotator, where LDR is as well a variable resistance that can be altered based on the amount of light it receives. Therefore, each condition of resistance value can be written to perform different actions, turn on LEDs for example.

Chapter 6: Analog Output

Ex 6.1: Use PIN 5 as PWM

- Measure with oscilloscope
    - Set duty cycle of PWM = 20%
    - Set duty cycle of PWM = 50%
    - Set duty cycle of PWM = 75%

Diagram:

No diagram provided. Simply measure a particular pin with a probe.

Coding:

```
void setup()
{
	Serial.begin(9600);
	pinMode(5, OUTPUT);
}

void loop()
{
	analogWrite(5, 51); //20%
	//analogWrite(5, 128); //50%
	//analogWrite(5, 191);  //75%		//switch when measures
}
```

Result:

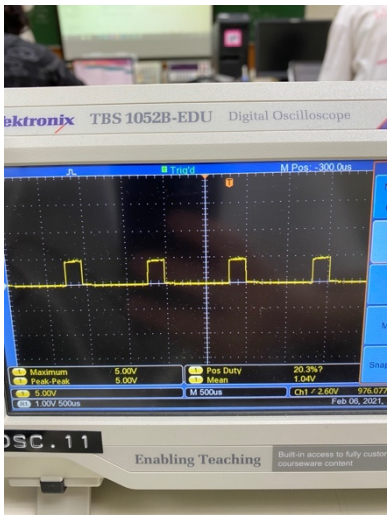


Figure 6.1.1: Output Graph for PWM = 20%
Frequency = 976 Hz

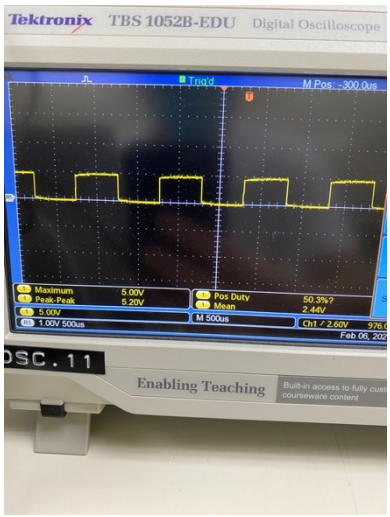


Figure 6.1.2: Output Graph for PWM = 50%
Frequency = 976 Hz

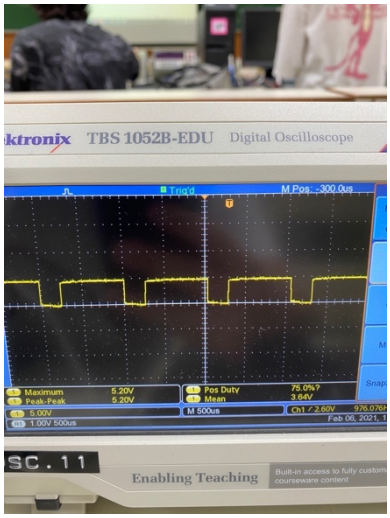


Figure 6.1.3 : Output Graph for PWM = 75%
Frequency = 976 Hz

Ex 6.2: Use PIN 9 as PWM

Measure with oscilloscope

- Set duty cycle of PWM = 20%
- Set duty cycle of PWM = 50%
- Set duty cycle of PWM = 75%

Diagram:

No diagram provided. Simply measure a particular pin with a probe.

Coding:

```
void setup()
{
	Serial.begin(9600);
	pinMode(9, OUTPUT);
}

void loop()
{
	analogWrite(9, 51); //20%
	//analogWrite(9, 128); //50%
	//analogWrite(9, 191); //75%
}
```

Result:

Similar output shapes with figure 6.1.1, 6.1.2, and 6.1.3 for PWM of 20%, 50%, and 75% respectively. The difference is frequency they represented at 488 Hz.

Ex 6.3: Control duty cycle of PWM with DIP SW (active high connection), use switching frequency as 500 Hz.

- Press Sw1 : AH-LED1 Blink 25%, PWM data show in serial and measure analog voltage output from multimeter.
- Press Sw2 : AH-LED1 Blink 50%, PWM data show in serial and measure analog voltage output from multimeter.
- Press Sw3 : AH-LED1 Blink 75%, PWM data show in serial and measure analog voltage output from multimeter.
- Press Sw4 : AH-LED1 Blink 100%, PWM data show in serial and measure analog voltage output from multimeter.
- Others : AH-LED1 OFF

Diagram:

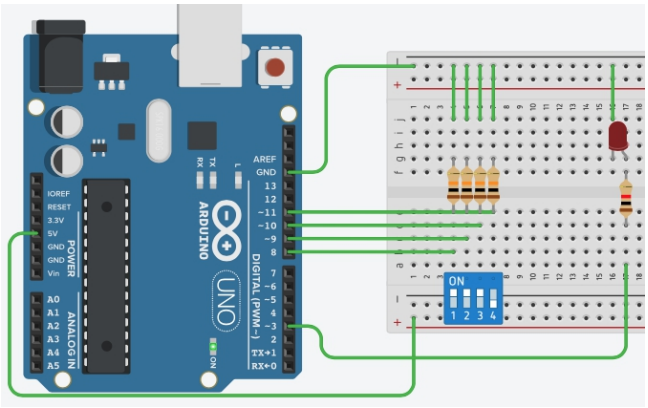


Figure 6.3.1

Coding:

```
void setup()
{
        Serial.begin(9600);
      for(int a=8;a<12;a++)
      {
            pinMode(a, INPUT);
      }
      pinMode(3, OUTPUT);
}

void loop()
{
      bool SW1 = digitalRead(8);
      bool SW2 = digitalRead(9);
      bool SW3 = digitalRead(10);
      bool SW4 = digitalRead(11);
       if(SW1==1 && SW2==0 && SW3==0 && SW4==0)
      {
            analogWrite(3, 64);
            Serial.println("SW1:");
            Serial.println("analogWrite = 64");
```

```
        }
        else if(SW1==0 && SW2==1 && SW3==0 && SW4==0)
        {
                 analogWrite(3, 128);
                Serial.println("SW2:");
                Serial.println("analogWrite = 128");
        }
        else if(SW1==0 && SW2==0 && SW3==1 && SW4==0)
        {
                analogWrite(3, 191);
                Serial.println("SW3");
                Serial.println("analogWrite = 191");
        }
        else if(SW1==0 && SW2==0 && SW3==0 && SW4==1)
        {
                analogWrite(3, 255);
                Serial.println("SW4:");
                Serial.println("analogWrite = 255");
        }
         else
        {
                 analogWrite(3, 0);
        }
        delay(100);
}
```

Result:

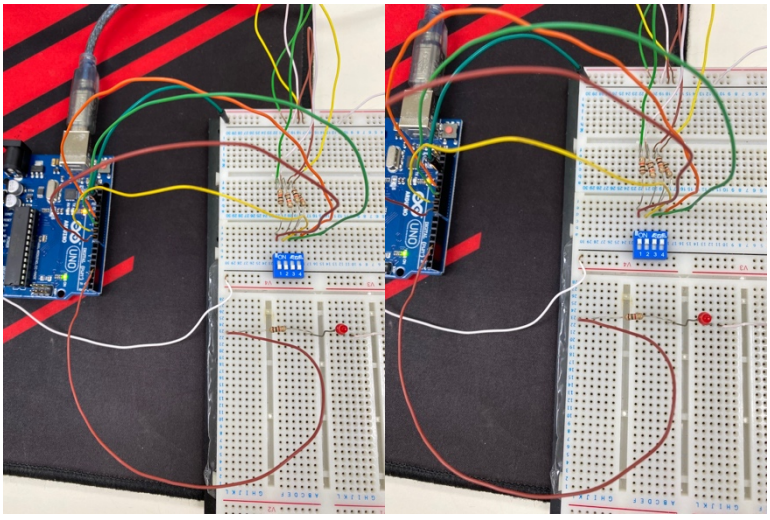


Figure 6.3.2: Others Condition

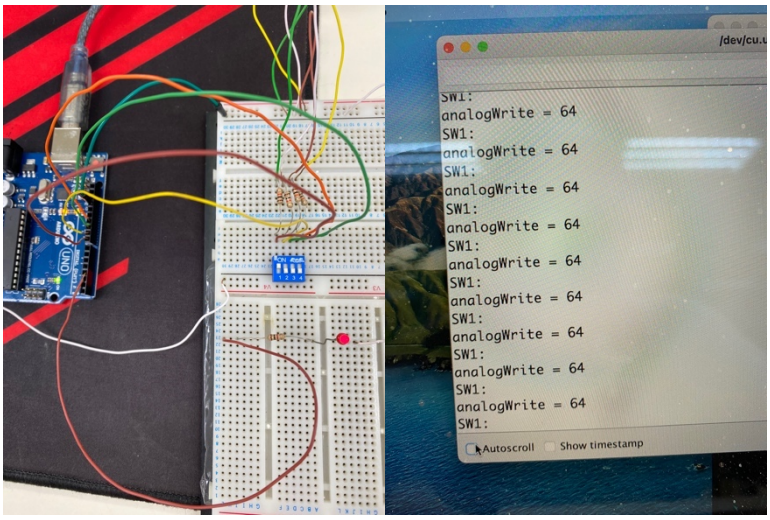


Figure 6.3.3: Switch 1 Condition

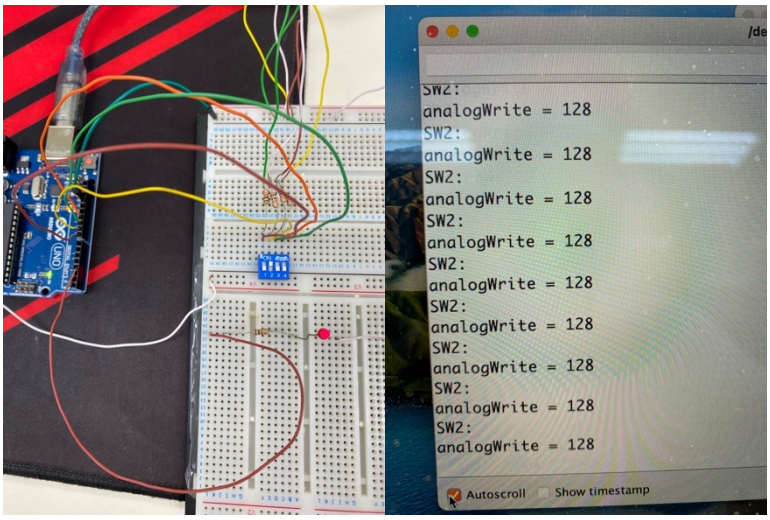


Figure 6.3.4: Switch 2 Condition

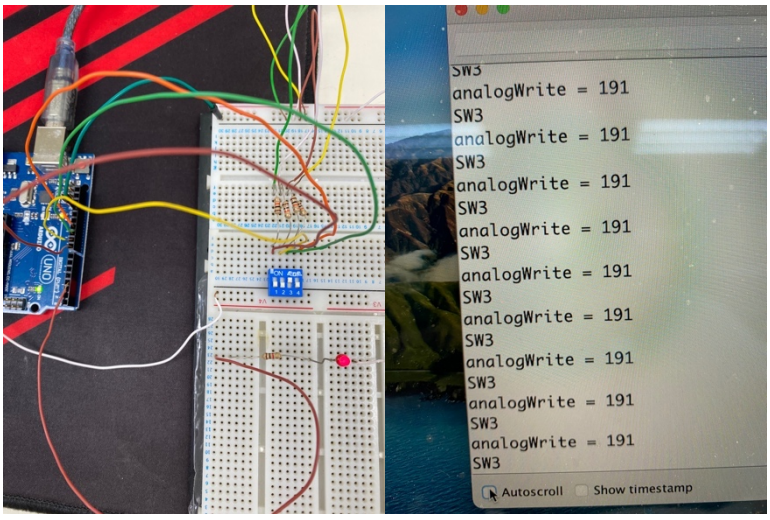


Figure 6.3.5: Switch 3 Condition

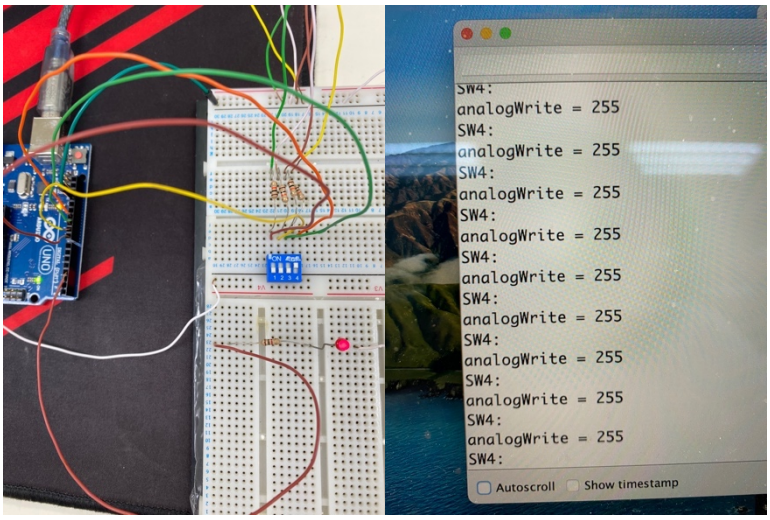


Figure 6.3.6: Switch 4 Condition

Ex 6.4: Control duty cycle of PWM with analog reading

- Analog read XX% : AH-LED1 Blink XX%, analog reading value show in serial monitor.

Diagram:

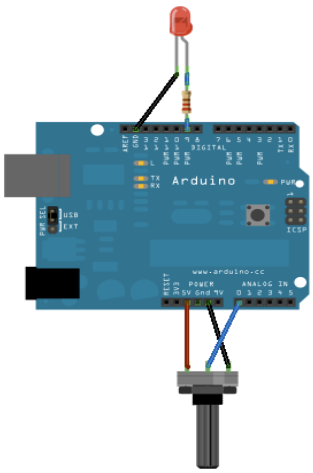


Figure 6.4.1

Coding:

```
void setup()
{
        Serial.begin(9600);
        pinMode(9, OUTPUT);
}

void loop()
{
        int AV1,AV2,X;
        AV1 = analogRead(A0);
        AV2 = AV1/4; // From 1023 >> 255
        X = (AV2*100)/255;
        analogWrite(9,AV2);
        Serial.println(X);
}
```

Result:

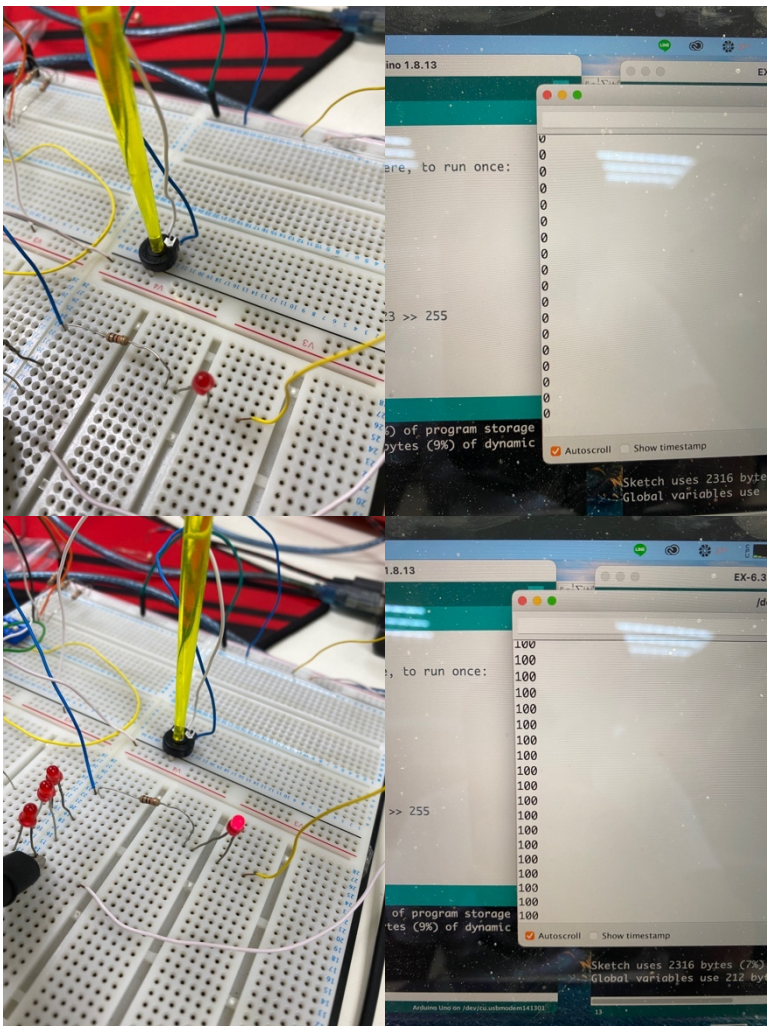


Figure 6.4.2 – 5

Ex 6.5:

- Use PIN 5 as PWM
- Set divisor to 8, measure the waveform
- Set divisor to 256, measure the waveform

Diagram:

No diagram provided. Simply measure a particular pin with a probe.

Coding:

```
void setup()
{
	//setPwmFrequency(5,8);
	setPwmFrequency(5,256);
	pinMode(5, OUTPUT);
}

void loop()
{
	analogWrite(5, 128);
}

void setPwmFrequency(int pin, int divisor)
{
	byte mode;
	if(pin == 5 || pin == 6 || pin == 9 || pin == 10)
	{
		switch(divisor)
		{
			case 1: mode = 0x01; break;
			case 8: mode = 0x02; break;
			case 64: mode = 0x03; break;
			case 256: mode = 0x04; break;
			case 1024: mode = 0x05; break;
			default: return;
		}
		if(pin == 5 || pin == 6)
		{
			 TCCR0B = TCCR0B & 0b11111000 | mode;
		}
		else
		{
		TCCR1B = TCCR1B & 0b11111000 | mode;
		}
	}
	else if(pin == 3 || pin == 11)
	{
		switch(divisor)
		{
```

```
                    case 1: mode = 0x01; break;
                    case 8: mode = 0x02; break;
                    case 32: mode = 0x03; break;
                    case 64: mode = 0x04; break;
                    case 128: mode = 0x05; break;
                    case 256: mode = 0x06; break;
                    case 1024: mode = 0x07; break;
                    default: return;
                }
                TCCR2B = TCCR2B & 0b11111000 | mode;
        }
}
```

Result:

Figure 6.5.1: Divisor of 8 Output

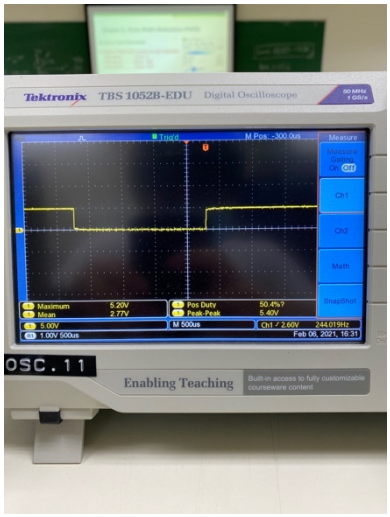


Figure 6.5.2: Divisor of 256 Output

Ex 6.6:

- Use PIN 9 as PWM
- Set divisor to 8, measure the waveform
- Set divisor to 256, measure the waveform

Diagram:

No diagram provided. Simply measure a particular pin with a probe.

Coding:

```
void setup()
{
	//setPwmFrequency(9,8);
	setPwmFrequency(9,256);
	pinMode(9, OUTPUT);
}

void loop()
{
	analogWrite(9, 128);
}

void setPwmFrequency(int pin, int divisor)
{
	byte mode;
	if(pin == 5 || pin == 6 || pin == 9 || pin == 10)
	{
		switch(divisor)
		{
			case 1: mode = 0x01; break;
			case 8: mode = 0x02; break;
			case 64: mode = 0x03; break;
			case 256: mode = 0x04; break;
			case 1024: mode = 0x05; break;
			default: return;
		}
		if(pin == 5 || pin == 6)
		{
			 TCCR0B = TCCR0B & 0b11111000 | mode;
		}
		else
		{
		TCCR1B = TCCR1B & 0b11111000 | mode;
		}
	}
	else if(pin == 3 || pin == 11)
	{
		switch(divisor)
		{
```

```
                    case 1: mode = 0x01; break;
                    case 8: mode = 0x02; break;
                    case 32: mode = 0x03; break;
                    case 64: mode = 0x04; break;
                    case 128: mode = 0x05; break;
                    case 256: mode = 0x06; break;
                    case 1024: mode = 0x07; break;
                    default: return;
                }
                TCCR2B = TCCR2B & 0b11111000 | mode;
        }
}
```

Result:

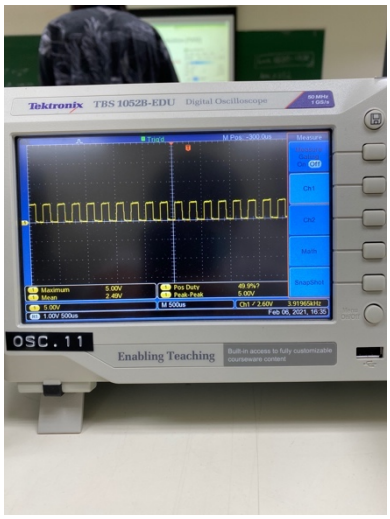


Figure 6.6.1: Divisor of 8 Output

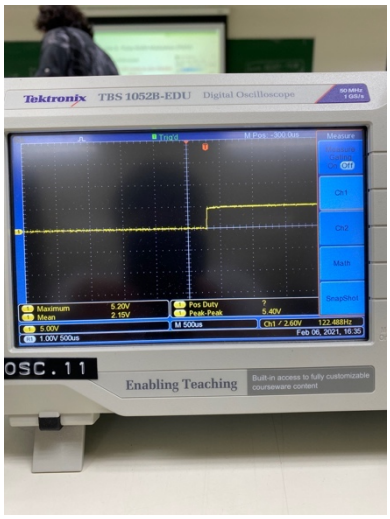


Figure 6.6.2: Divisor of 256 Output

Ex 6.7: Use RGB module

- Perform RGB LED module as color indicated
  - Pin for R ……., Value 0
  - Pin for G ……., Value 108
  - Pin for B ……., Value 255

Diagram:

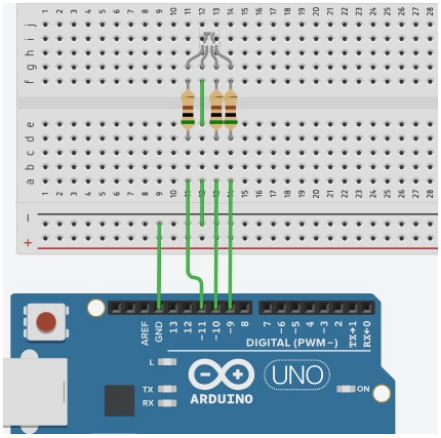


Figure 6.7.1

Coding:

```
void setup()
{
        for(int a=9;a<12;a++)
        {
                pinMode(a, OUTPUT);
        }
}

void loop()
{
        analogWrite(9, 255-0);
        analogWrite(10, 255-108);
        analogWrite(11, 255-255);
}
```

Result:

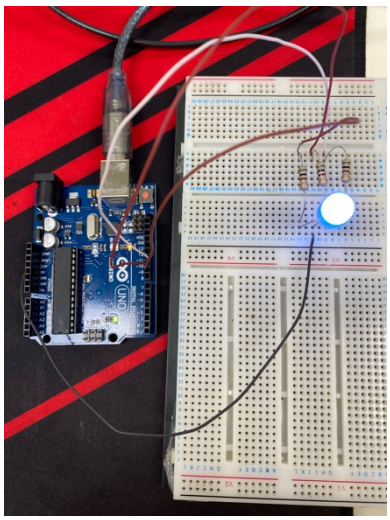


Figure 6.7.2

Ex 6.8: Speed control of motor
Set f = 3.9 KHz

- Press “a” from KB, set motor 100% CW
- Press “b” from KB, set motor 60% CW
- Press “s” from KB, set motor 0%
- Press “d” from KB, set motor 100% CCW
- Press “e” from KB, set motor 60% CCW

Diagram:

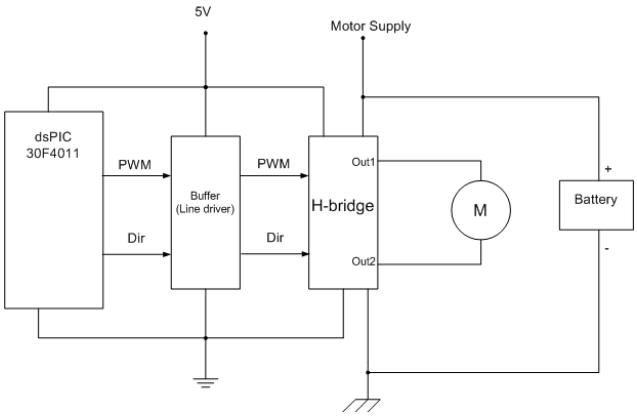


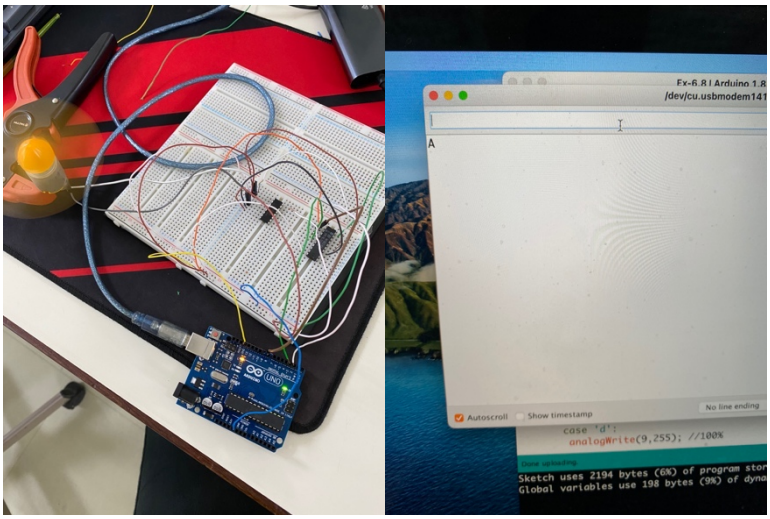
Figure 6.8.1

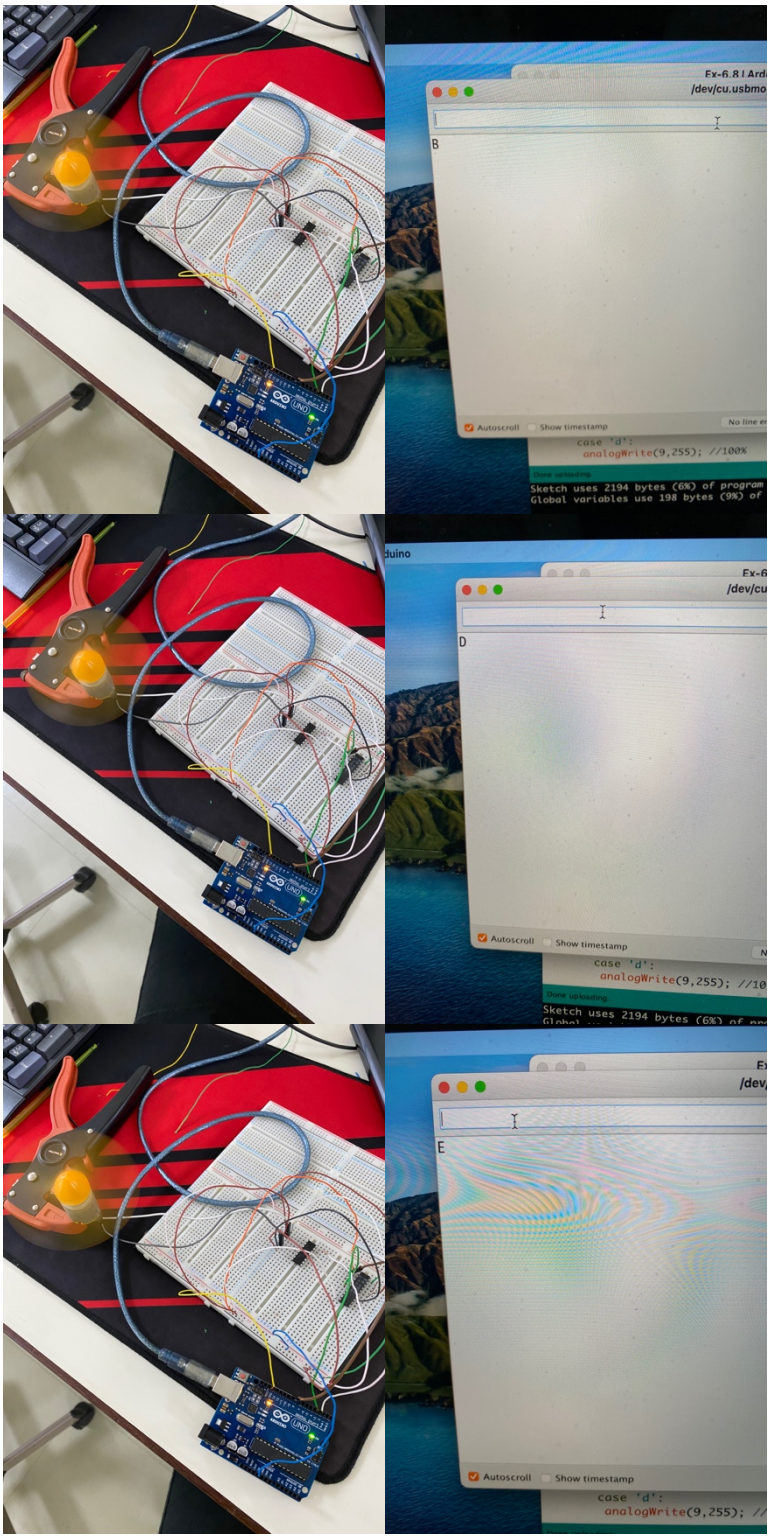
Coding:

```
void setup()
{
        Serial.begin(9600);
        pinMode(2,OUTPUT);
        pinMode(3,OUTPUT);
        pinMode(9,OUTPUT);
}

void loop()
{
        if(Serial.available()>0)
        {
                char input;
                input = Serial.read();
                switch(input)
                {
                        case 'a':
                        analogWrite(9,255); //100%
                        digitalWrite(2, LOW);
                        digitalWrite(3, HIGH);
                        Serial.println("A");
                        break;
                        case 'b':
                        analogWrite(9,153); //60%
                        digitalWrite(2, LOW);
```

```
                digitalWrite(3, HIGH);
                Serial.println("B");
                break;
                case 'd':
                analogWrite(9,255); //100%
                digitalWrite(2, HIGH);
                digitalWrite(3, LOW);
                Serial.println("D");
                break;
                case 'e':
                analogWrite(9,153); //60%
                digitalWrite(2, HIGH);
                digitalWrite(3, LOW);
                Serial.println("E");
                break;
                case 's':
                analogWrite(9,0); //0%
                digitalWrite(2, LOW);
                digitalWrite(3, LOW);
                Serial.println("S");
                break;
            }
        }
}
```

Result:




B
Autoscroll
Show timestamp
case 'd':
analogWrite(9,255); //100%
Sketch uses 2194 bytes (6%) of program
Global variables use 198 bytes (9%) of
D
E

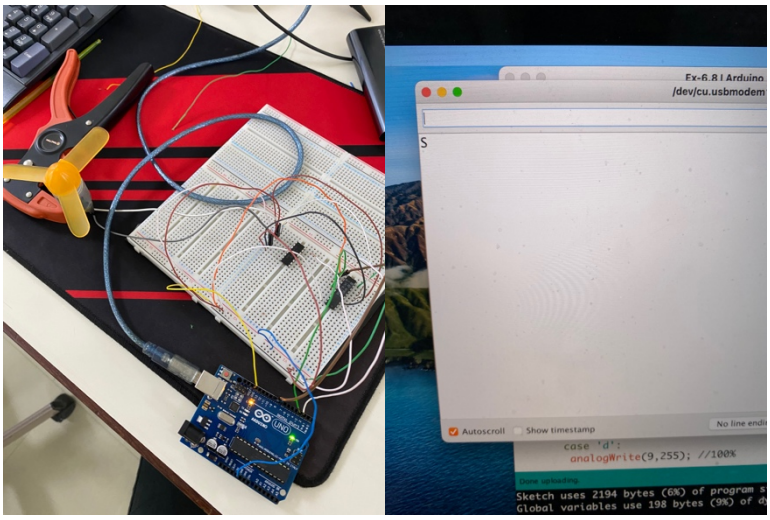


Figure 6.8.2 – 11

Conclusion:

In chapter 6, an analog output of Arduino was introduced. Pulse Width Modulation (PWM) is a method used to vary the frequency of output by causing change to the duty cycle from Arduino. PWM can commonly be found on the Arduino board itself. The symbol can be indicated as "~" in front of the pin numbers that are capable of performing such action. For this case, pin 5 and 6 with 1 kHz of frequency and pin 3, 9, 10, and 11 with 500 Hz of frequency. This can be used through several kind of devices and circuit such as RGB and Fan Control circuit.

Chapter 7: Timer and External Interrupts

Ex 7.1: Interrupt at 0.5s

- ON buzzer a beep 0.1s for every 0.5s

T = 0.5s, → f = 2Hz
16M/256= 62500
Therefore 62500/2 = 31250

Diagram:

Similar to figure 4.4.1

Coding:

```
void setup()
{
	pinMode(13, OUTPUT);
	Serial.begin(9600);
	noInterrupts();
	//Make a timer as 0.5s
	TCCR1A = 0; //timer 1
	TCCR1B = 0;
	TCNT1  = 0; //clear 0

	OCR1A  = 31250; //Set time value
	TCCR1B |=(1<<WGM12); // clear timer
	TCCR1B |=(1<<CS12); //Set prescaler as 256
	TIMSK1 |=(1<<OCIE1A); //Enable timer compare

	interrupts(); //Enable interrupts
}

ISR(TIMER1_COMPA_vect)
{
	digitalWrite(13,HIGH);
	delay(100);
	digitalWrite(13,LOW);
}

void loop()
{}
```

Result:

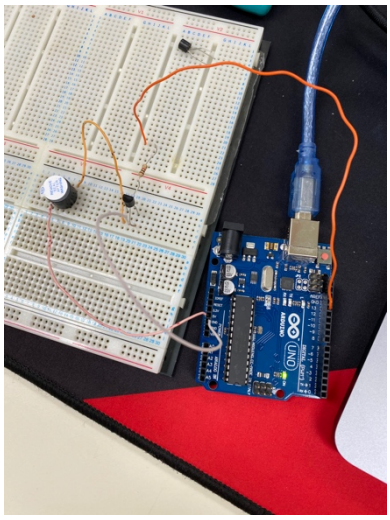


Figure 7.1.2

Ex 7.2: Interrupt at 4 s

- ON buzzer a beep 0.1s for every 4 s

T = 4 s, → f = 0.25 Hz
16M/256= 62500
Therefore 62500/0.25 = 31250

Diagram:

Similar to figure 4.4.1

Coding:

```
void setup()
{
        pinMode(13, OUTPUT);
        Serial.begin(9600);
        noInterrupts();
        //Make a timer as 4 s
        TCCR1A = 0;
        TCCR1B = 0;
        TCNT1  = 0; //clear 0

        OCR1A  = 62500; //Set time value
        TCCR1B |=(1<<WGM12); // clear timer
        TCCR1B |=(1<<CS12);
        TCCR1B |=(1<<CS10); //Set prescaler as 1024
        TIMSK1 |=(1<<OCIE1A); //Enable timer compare

        interrupts(); //Enable interrupts
}
```

```
ISR(TIMER1_COMPA_vect)
{
        digitalWrite(13,HIGH);
        delay(100);
        digitalWrite(13,LOW);
        delay(100);
}

void loop()
{}
```

Result:

Similar to figure 7.1.2 but differs in interruption time

Ex 7.3: Running (CC) 7 segment 0 - 9 delay 1 second

- Timer interrupt every 1 second, switch ON buzzer pin4 for 0.2s, and switch OFF for 0.8s.

T = 1 s, → f = 1 Hz
16M/256= 62500
Therefore 62500/1 = 62500

Diagram:

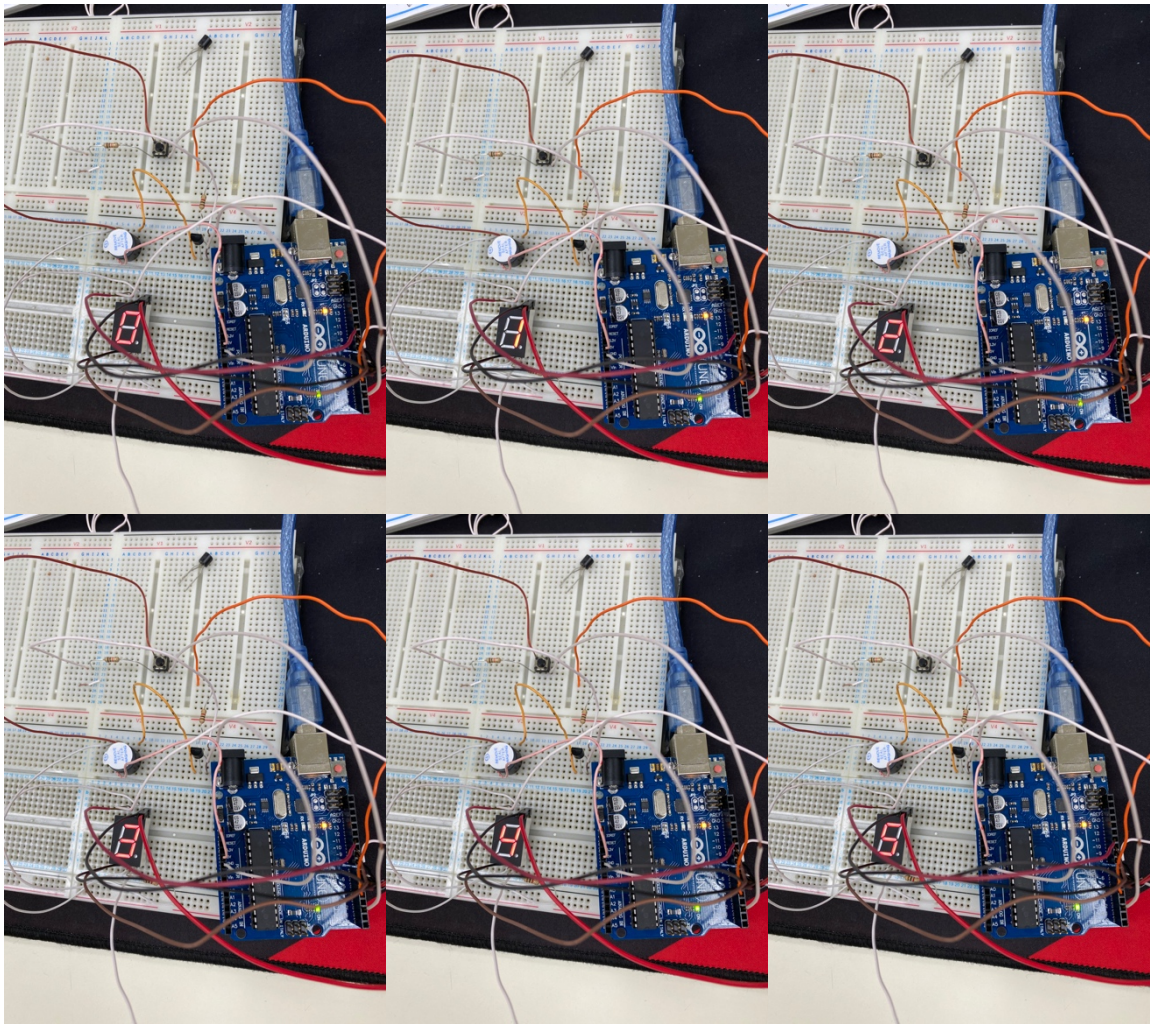
Similar to figure 2.4.1 for 7 segments and figure 4.4.1 for buzzer connection

Coding:

```
void setup()
{
	for(int a=4;a<12;a++)
	{
		pinMode(a, OUTPUT);
	}
	Serial.begin(9600);
	noInterrupts();
	//Make a timer as 1s
	TCCR1A = 0;
	TCCR1B = 0;
	TCNT1  = 0; //clear 0

	OCR1A  = 62500; //Set time value
	TCCR1B |=(1<<WGM12); // clear timer
	TCCR1B |=(1<<CS12); //Set prescaler as 256
	TIMSK1 |=(1<<OCIE1A); //Enable timer compare

	 interrupts(); //Enable interrupts
}

ISR(TIMER1_COMPA_vect)
{
	digitalWrite(4,HIGH);
	delay(200);
	digitalWrite(4,LOW);
	delay(800);
}

void loop()
{
	int number[10][7] =
	{{0,0,0,0,0,0,1},{1,0,0,1,1,1,1},{0,0,1,0,0,1,0},{0,0,0,0,1,1,0},{1,0,0,1,1,0,0},{0,1,0,
	0,1,0,0},{0,1,0,0,0,0,0},{0,0,0,1,1,1,1},{0,0,0,0,0,0,0},{0,0,0,0,1,0,0}};

	for(int a=0;a<10;a++)
```

```
        {
                for(int b=5;b<12;b++)
                {
                        digitalWrite(b ,number[a][b-5]);
                }
                delay(1000);
        }
}
```

Result:



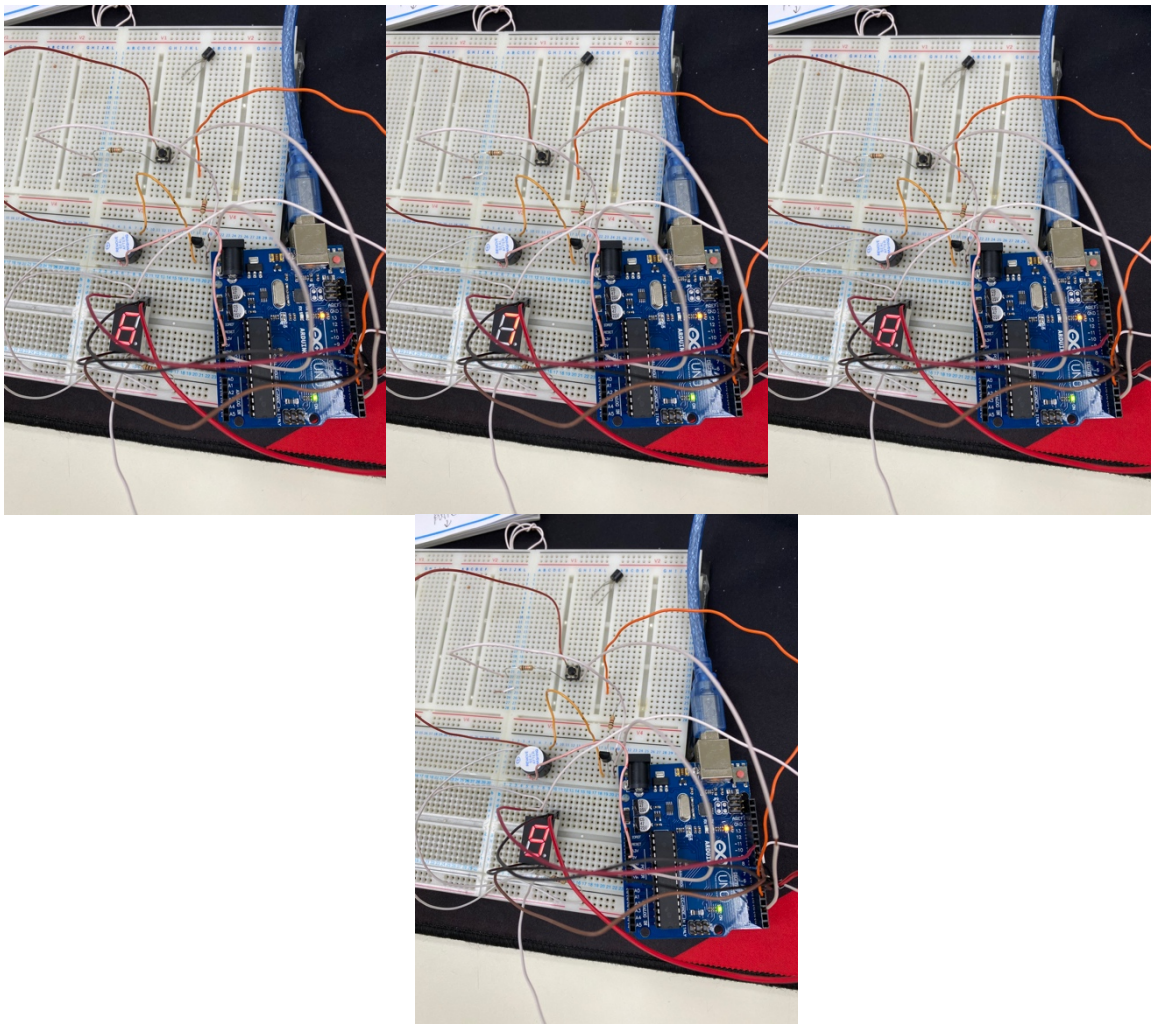


Figure 7.3.1 – 10

Ex 7.4: Running (CC) 7 segment 0 - 9 with delay 1 second

- If interrupt SW is pressed
    - Buzzer On for a short beep

Diagram:

Similar to figure 2.4.1 for 7 segments and figure 4.4.1 for buzzer connection

Coding:

```
void buzzer()
{
	digitalWrite(4, HIGH);
	delay(1000);
	digitalWrite(4, LOW);
}
void setup()
{
	for(int a=4;a<12;a++)
	{
		pinMode(a, OUTPUT);
	}
	pinMode(2, INPUT);
	 attachInterrupt(digitalPinToInterrupt(2), buzzer, RISING);
}

void loop()
{
	int number[10][7] =
	{{0,0,0,0,0,0,1},{1,0,0,1,1,1,1},{0,0,1,0,0,1,0},{0,0,0,0,1,1,0},{1,0,0,1,1,0,0},{0,1,0,
	0,1,0,0},{0,1,0,0,0,0,0},{0,0,0,1,1,1,1},{0,0,0,0,0,0,0},{0,0,0,0,1,0,0}};

	 for(int a=0;a<10;a++)
	{
		for(int b=5;b<12;b++)
		{
			digitalWrite(b ,number[a][b-5]);
		}
		delay(1000);
	}
}
```

Result:

The result is distinctly similar to figure 7.3.1 – 10 but the addition of a switch is installed to carry out interruption.

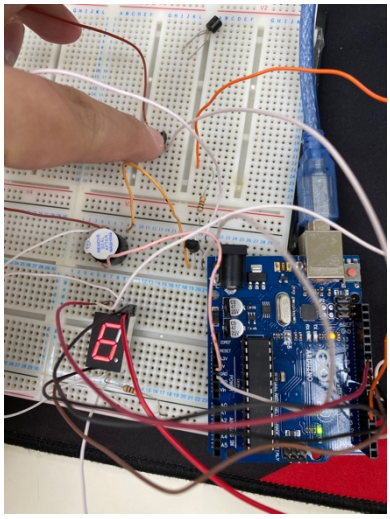


Figure 7.4.1: Additional Switch to trigger interrupt.

Conclusion:

In chapter seven, a timer and external interrupts were introduced. Timer function allows an action to be performed repeatedly through a certain period of time without trigger requirement. However, a prescaler is required to be calculated based on the amount of time the action lasts. Whereas, interrupts function allows an action to be performed immediately as soon as a certain interruption occurs which considered a top priority action among all.

Chapter 0: PCB Design

Result:

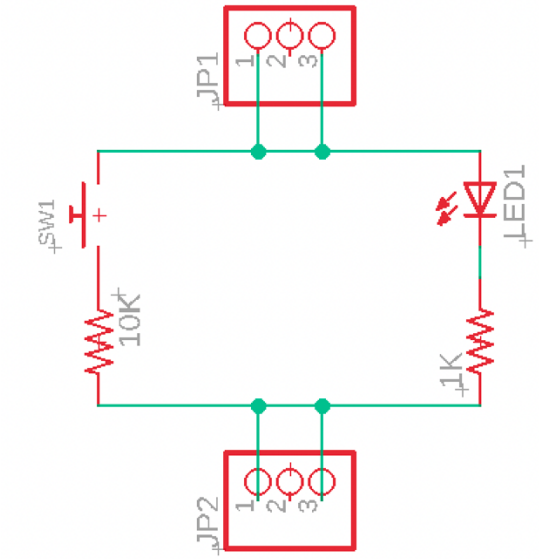


Figure 0.1: Schematic Design

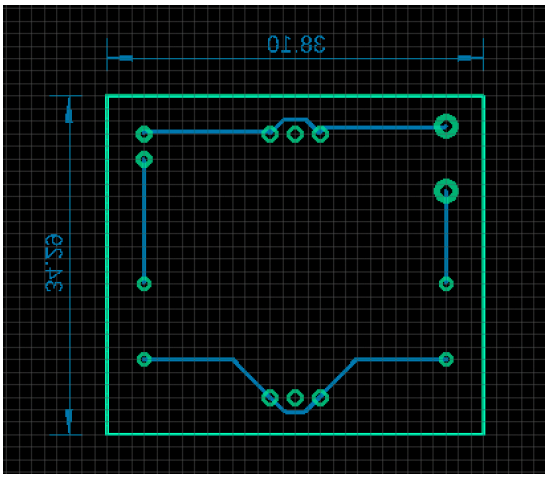


Figure 0.2: Board Design

Conclusion:

In chapter 0, a PCB design was introduced by using EAGLE application. A PCB is the use of computer programming in order to eliminate wirings and replace them with thin metal plates embedded on a plastic board. The process began with the creation of schematic. In this step, footprint selection was crucial. After assembled all devices into a circuit, board can be generated. However, arrangement is required again on the board before autorouting. This was the most time consuming as it may not be run into a 100% in a first attempt. Some arrangements

had to be done additionally to ensure that the metal plates didn't overlap each other causing short circuit to some connection. As soon as the autorouter is made completely, the dimension was created to indicate size of the PCB board. Finally, some layers has to be arranged and mirrored before saving as pdf file.