Arduino Interfacing

SY_A_AMM_UNIT4_SJM

Unit 4

Unit III	Programming	06 Hours
Learning Arduino code basics: Arduino C- Arduino Program Structure, variables, Using Mathematical Operators, using Arduino String Functionality, Repeating a Sequence of Statements		
Unit IV	**Interfacing**	**06 Hours**
Interfacing digital inputs and outputs, Connecting and Using LED, interfacing 7-segment display, Interfacing keypad , Measuring Distance using IR sensor, Detecting Light using LDR		

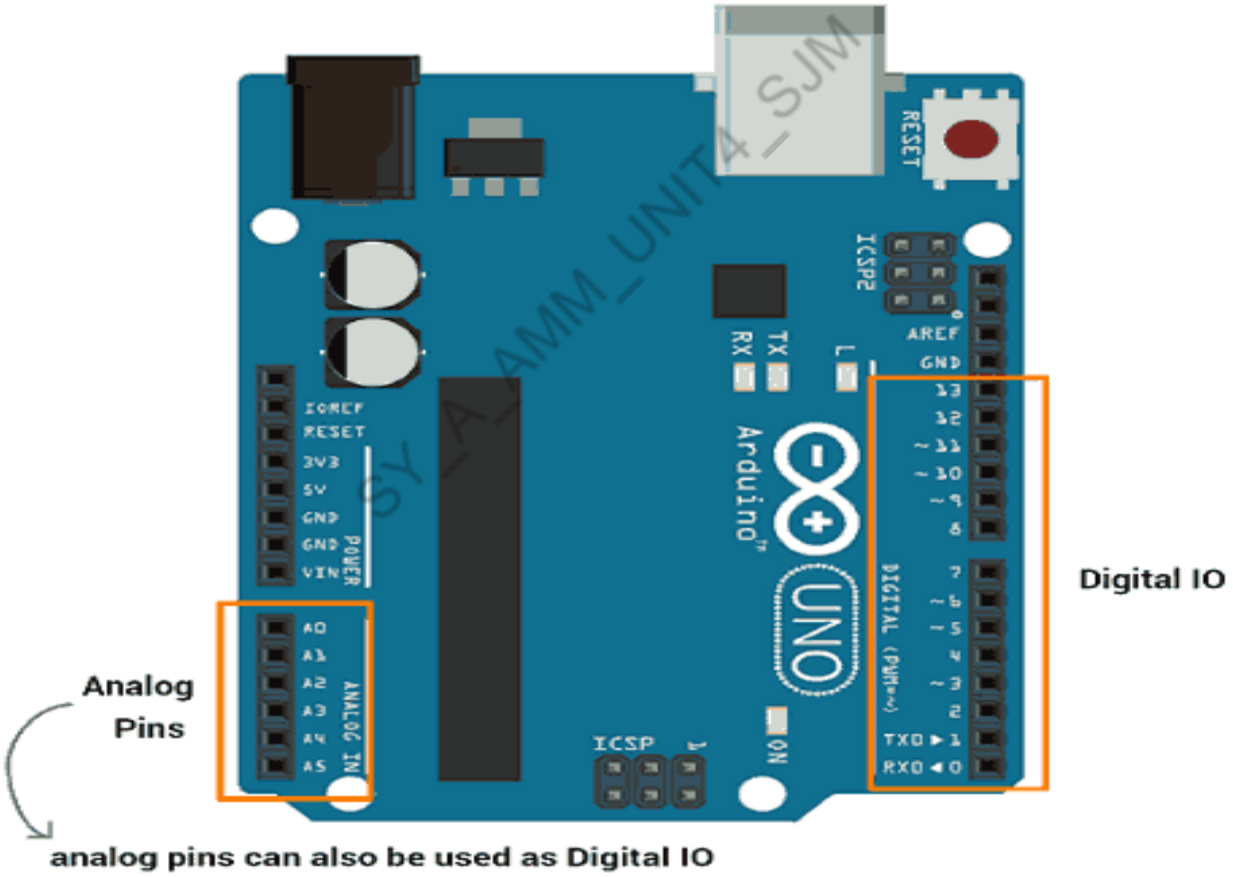
Interfacing digital inputs and outputs

- Digital GPIO of Arduino
  - **G**eneral-**P**urpose **I**nput **O**utput (GPIO) is a digital pin of an IC. It can be used as input or output for interfacing devices.
  - If we want to read switch's state, sensor data, etc then we need to configure it as input.
  - if we want to control the LED brightness, motor rotation, show text on display, etc then we need to configure it as output.
  - Arduino Uno board has various digital IO pins which can be used for input/output devices

SY / AMM_UNIT4_PPT

Interfacing digital inputs and outputs

Digital IO pins of Arduino Uno



Interfacing digital inputs and outputs

- **Digital Output**
  - Arduino digital pins can be configured as output to drive output devices.
  - To configure these pins, **pinMode ()** function is used which set direction of pin as input or output.
  - **E.g.** pinMode (3, OUTPUT) …..set pin 3 as output.
  - These Arduino (ATmega) pins can source or sink current up to 40 mA which is sufficient to drive led, LCD display but not sufficient for motors, relays, etc.
  - while connecting devices to Arduino output pins use resistor. If any connected device to Arduino withdraw current more than 40 mA from the Arduino then it will damage the Arduino IC.
  - These pin produce output in terms of HIGH (5 V or 3.3 V) or LOW (0 V).
  - We can set output on these pins using **digitalWrite ()** function.
- **E.g. digitalWrite (3, HIGH)**

SY_A_AM_UNIT4_SJM

Interfacing digital inputs and outputs

- **Digital Input**
  - To read data from senor or from any Device/circuit, we need to configure digital pin as input. Arduino pin are set as digital input (default). So, there is no need to configure pin as input.
  - We can read data from GPIO pin using digitalRead() function.

SY_A_AMM_UNIT4_SJM

Interfacing digital inputs and outputs

- **Digital Input with Pull-up Resistor**

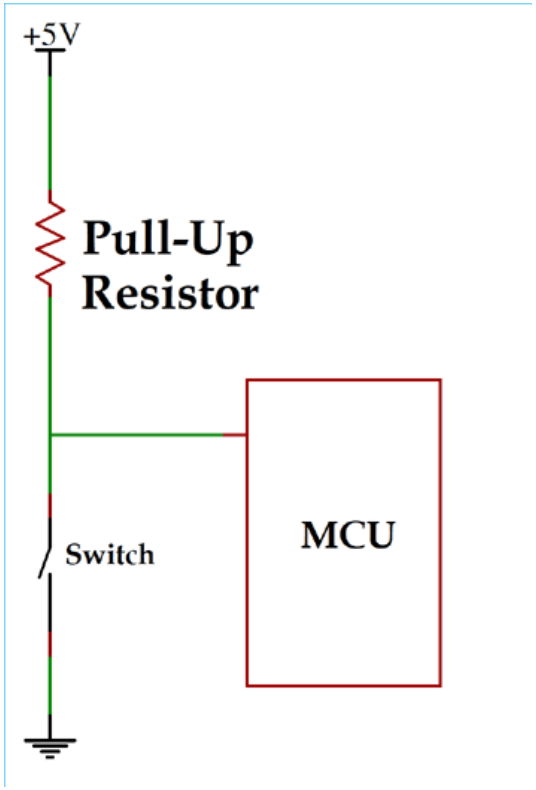
  As digital circuit works in low current, connecting the logic pins directly to the supply voltage or the ground is not a good choice.

  – As direct connection eventually increase current flow just like the short circuit and could damage the sensitive logic circuit which is not advisable.
  – To control the current flow, we need those pull-down or pull up resistors.
  – A pull-up resistor allow controlled current flow from supply voltage source to the digital input pins.
  – Arduino (ATmega) has in-built configurable pull-up resistor.
  – If we set direction of pin as input and then write HIGH value on that pin will turn on the pull-up resistor.

Example:

pinMode (3, INPUT) //set pin as input

digitalWrite (3, HIGH) //setting high on input pin enables pull up



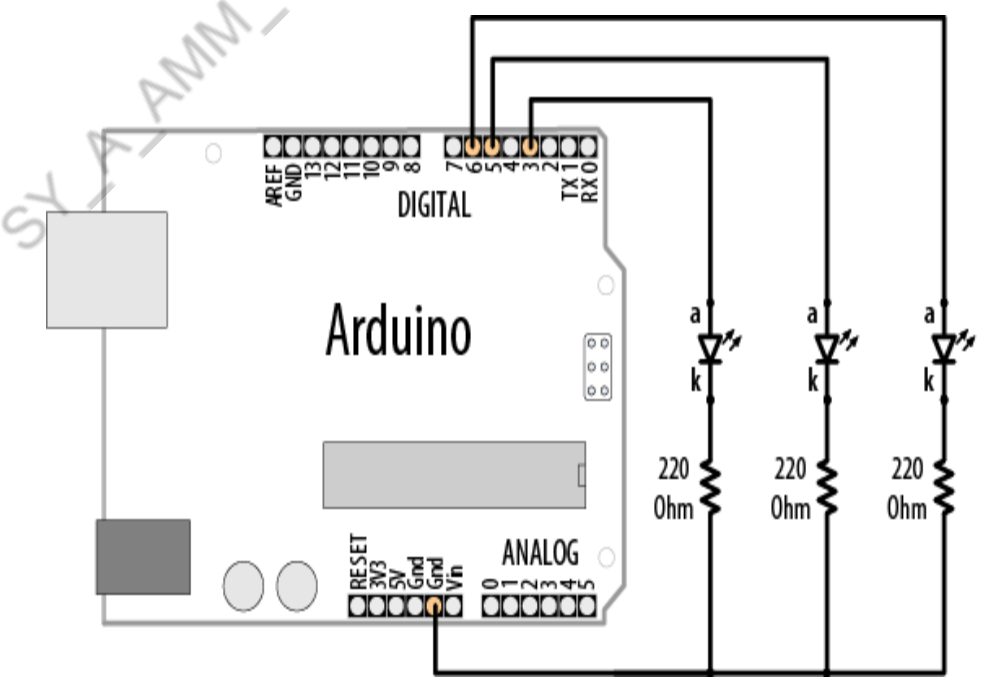
Switch = close	Pin logic=0
Switch = Open	Pin logic = 1

2) Connecting and using LED

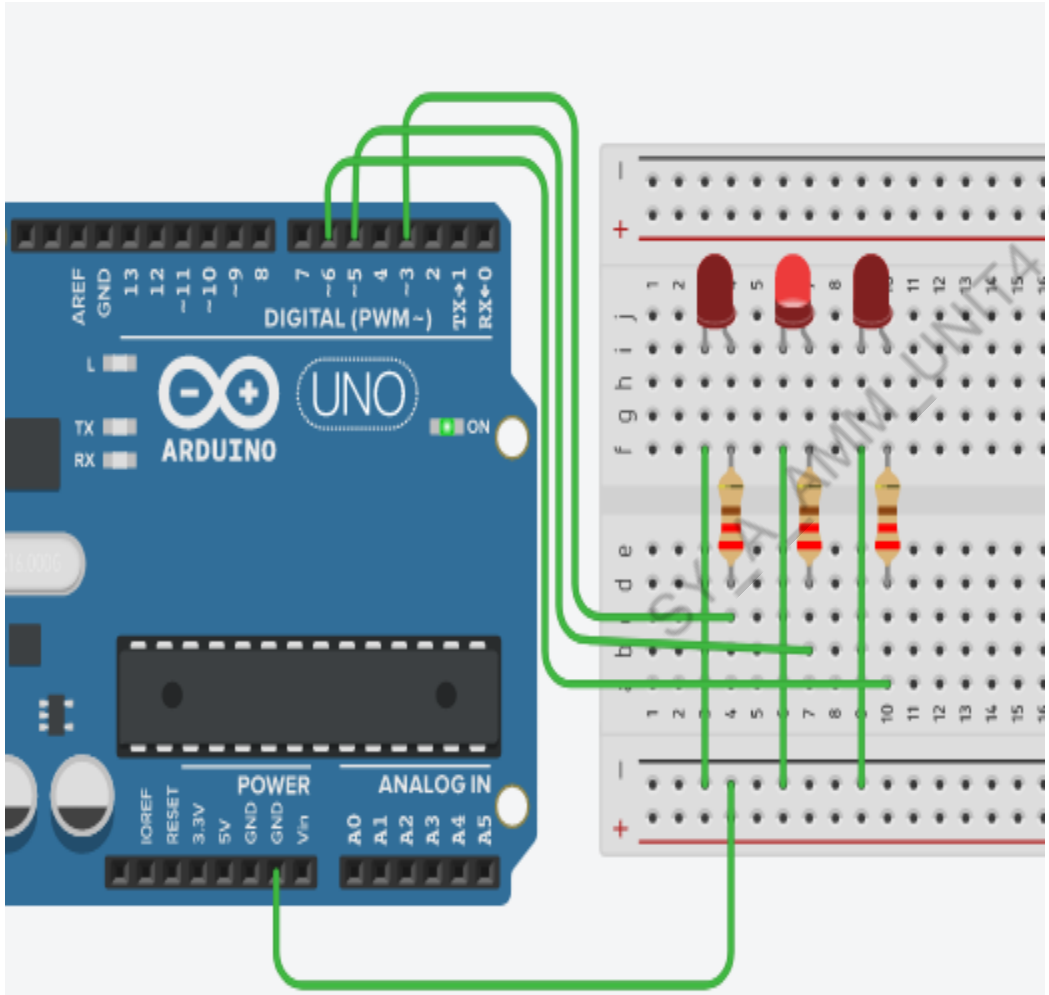
```
int  ledPins[] =  {3,5,6};
void setup()
{
for(int i = 0; i < 3; i++)
{
pinMode(ledPins[i], OUTPUT);

}
}
void loop()
{
for(int  i = 0; i < 3; i++)
{
digitalWrite(ledPins[i], HIGH);
 delay(1000);
digitalWrite(ledPins[i], LOW);
delay(1000);
}
}
```

Following sketch lights up three LED's connected to pins 3,5, and 6 in sequence for one second.

SY_A_AMM_UNIT4_SJM



Connecting and using LED's




```
int  ledPins[] =  {3,5,6};
void setup()
{
for(int i = 0; i < 3; i++)
{
pinMode(ledPins[i], OUTPUT);

}
}
void loop()
{
for(int  i = 0; i < 3; i++)
{
digitalWrite(ledPins[i], HIGH);
 delay(1000);
digitalWrite(ledPins[i], LOW);
delay(1000);
}
}
```

Connecting and using LED

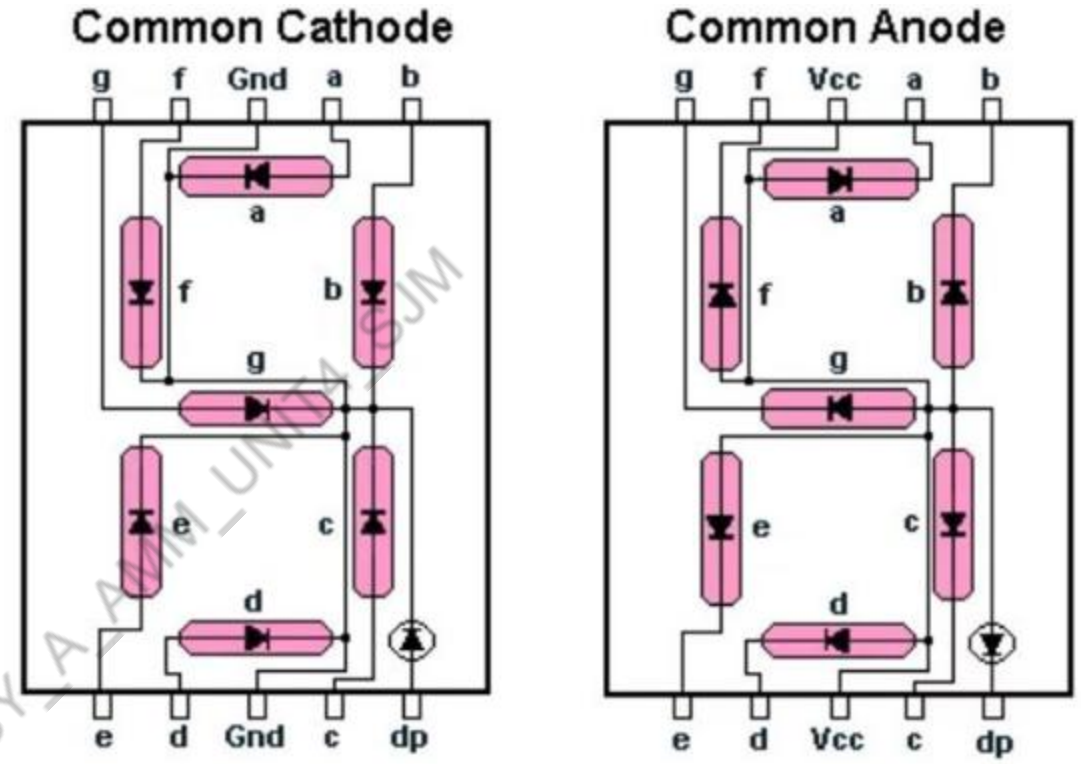
- A resistor in series with the led is used to control the amount of current that will flow when the LED conducts.
- To calculate the resistor value
- $R = (V_S - V_F)/I$
- VS = Supply voltage = 5v for Arduino board.
- Vf= 1.8 V Forward voltage of an led
- I=15ma
- R=(5-1.8)/15ma
- R= 213 ohms
- We can round this up to 220ohms.

SY_I_AM_UNIT4_SJM

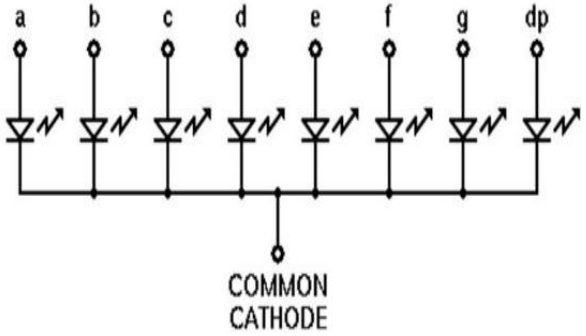
Interfacing 7 segment display.

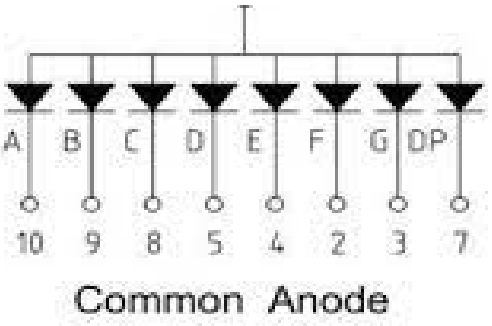
- Seven segment display consists of **seven LED's** arranged in a rectangular fashion as shown.
- Forward biasing the appropriate pins of the LED segments in a particular order, some segments will be **light** and others will be **dark**.
- This allow us to display each of te decimal **digits 0 through 9** on the same seven segment display.
- The display **common pin** is genera used to identify type of seven segment display.
- For common cathode seven segment display you have to connect the centre common pin of display to GND.
- For common anode seven segment display you have to connect the centre common pin of display to+5V

.



SY_A_AMM_UNIM_SJM





Interfacing 7 segment display(Common Cathode) to Arduino Uno

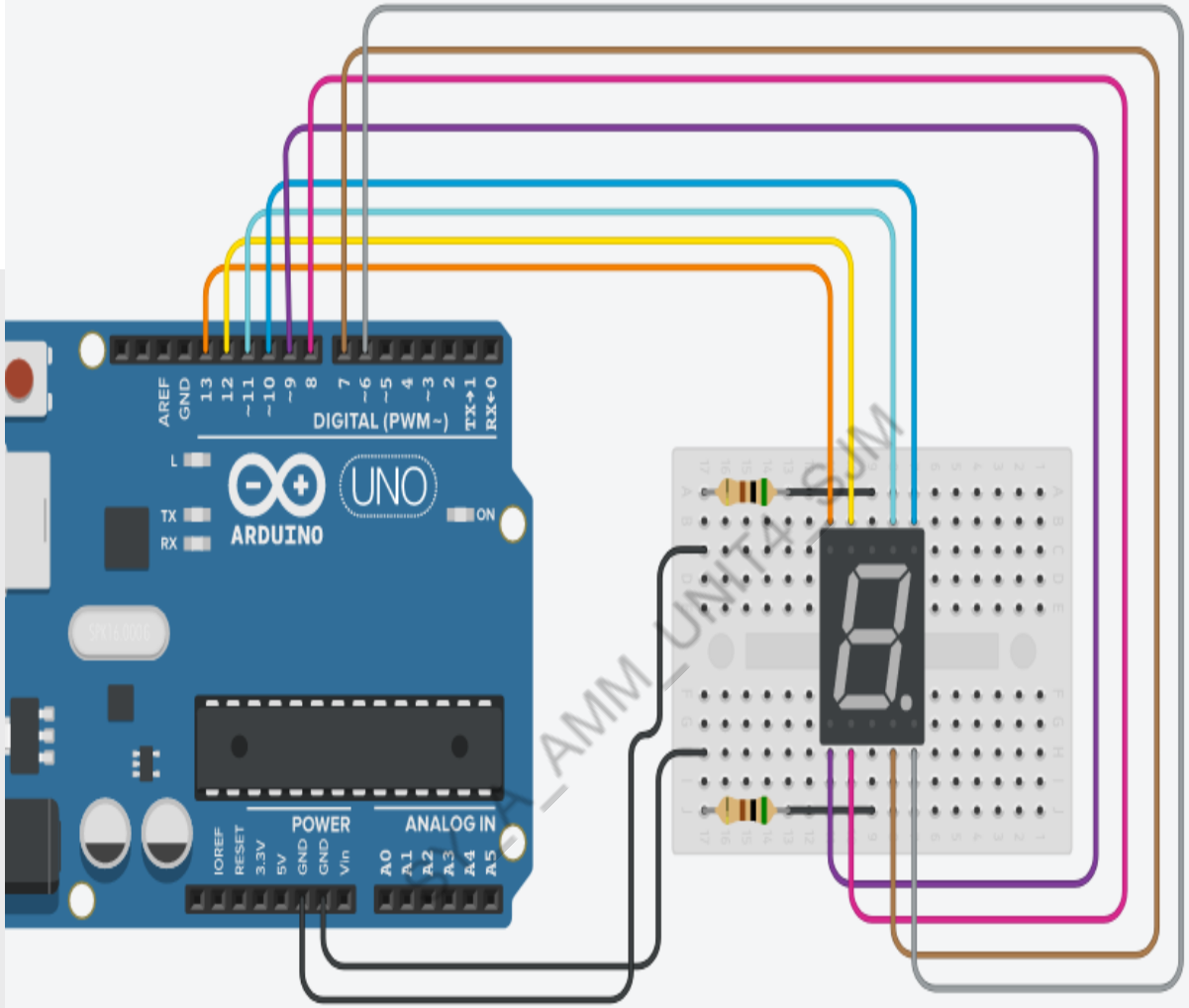
```
const int  G = 13;
const int  F = 12;
const int  A= 11;
const int  B = 10;
const int  E = 09;
const int  D = 08;
const int  C = 07;
const int  H = 06;
```

```
void setup()
{
pinMode(A,OUTPUT);
pinMode(B,OUTPUT);
pinMode(C,OUTPUT);
pinMode(D,OUTPUT);
pinMode(E,OUTPUT);
pinMode(F,OUTPUT);
pinMode(G,OUTPUT);
pinMode(H,OUTPUT);
}
```

```
 void zero()
{
digitalWrite(A,HIGH);
digitalWrite(B,HIGH);
digitalWrite(C,HIGH);
digitalWrite(D,HIGH);
digitalWrite(E,HIGH);
digitalWrite(F,HIGH);
digitalWrite(G,LOW);
digitalWrite(H,LOW);
}

Void one()
{

}

 void loop()
{
 zero();
 one();
 two();
 three();
 four();
 five();
 six();
 seven();
 eight();
 nine();
}
```

SY_A_AMM_UNIT4.pdf

```
// Start
void loop(void)
{
  zero();
  delay(1000);

  one();
  delay(1000);

  two();
  delay(1000);

  three();
  delay(1000);

  four();
  delay(1000);

  five();
  delay(1000);

  six();
  delay(1000);

  seven();
  delay(1000);

  eight();
  delay(1000);

  nine();
  delay(1000);
}
```




```
unsigned const int G = 13;
unsigned const int F = 12;
unsigned const int A = 11;
unsigned const int B = 10;
unsigned const int E = 9;
unsigned const int D = 8;
unsigned const int C = 7;
unsigned const int H = 6;


void setup(void)
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(H, OUTPUT);
}

//My Functions

void zero(void) {
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, LOW);
}
```

Interfacing 7 segment display(Common Anode) to Arduino Uno

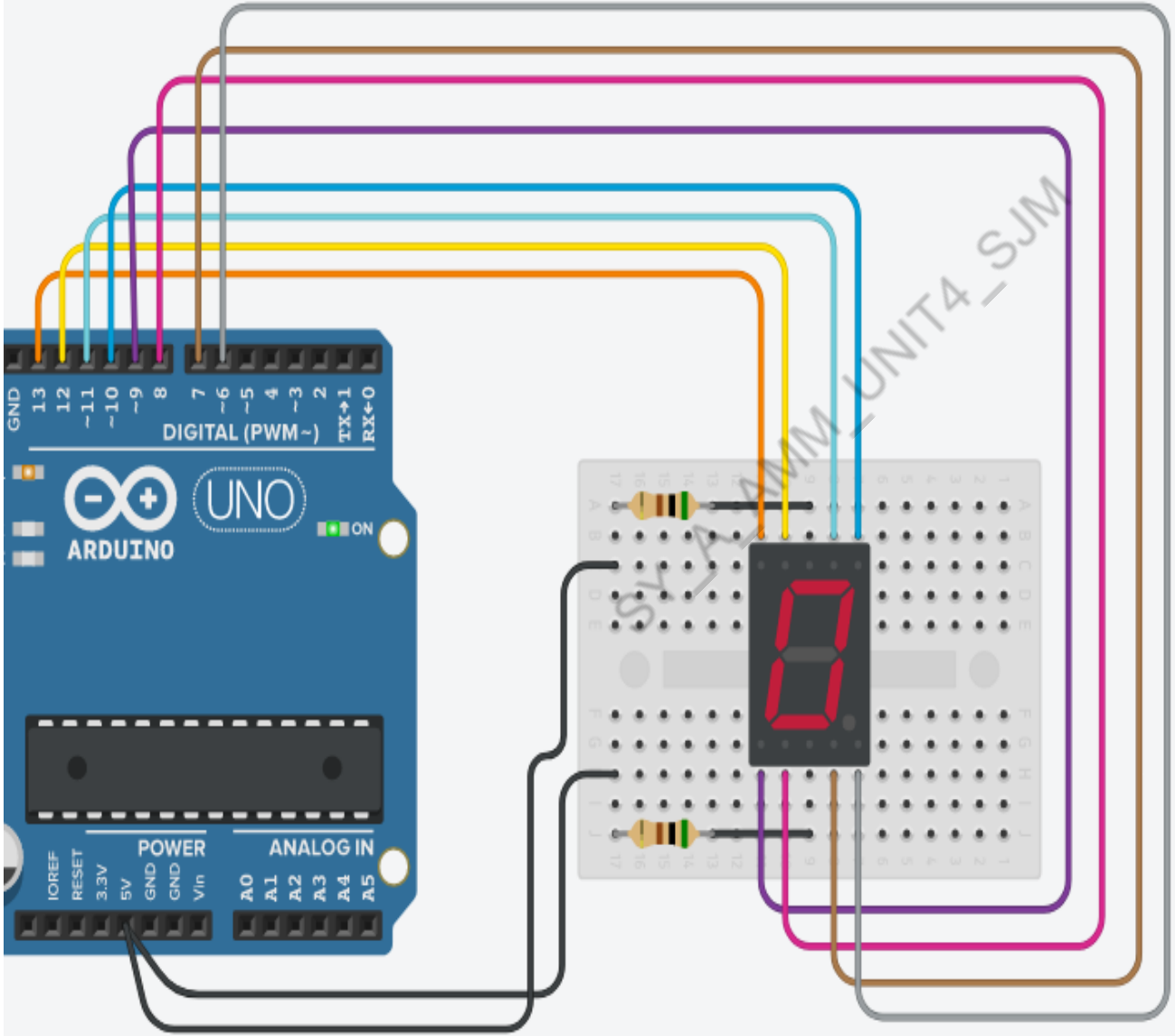
```
const int  G = 13;
const int  F = 12;
const int  A= 11;
const int  B = 10;
const int  E = 09;
const int  D = 08;
const int  C = 07;
const int  H = 06;

void setup()
{
pinMode(A,OUTPUT);
pinMode(B,OUTPUT);
pinMode(C,OUTPUT);
pinMode(D,OUTPUT);
pinMode(E,OUTPUT);
pinMode(F,OUTPUT);
pinMode(G,OUTPUT);
pinMode(H,OUTPUT);
}
```

```
 void zero()
{
digitalWrite(A,LOW);
digitalWrite(B,LOW);
digitalWrite(C,LOW);
digitalWrite(D,LOW);
digitalWrite(E,LOW);
digitalWrite(F,LOW;
digitalWrite(G,HIGH;
digitalWrite(H,HIGH;
}
```

```
 void loop()
{
 zero();
 one();
 two();
 three();
 four();
 five();
 six();
 seven();
 eight();
 nine();
}
```

SY_A_AMM_UNIT_4.pptx

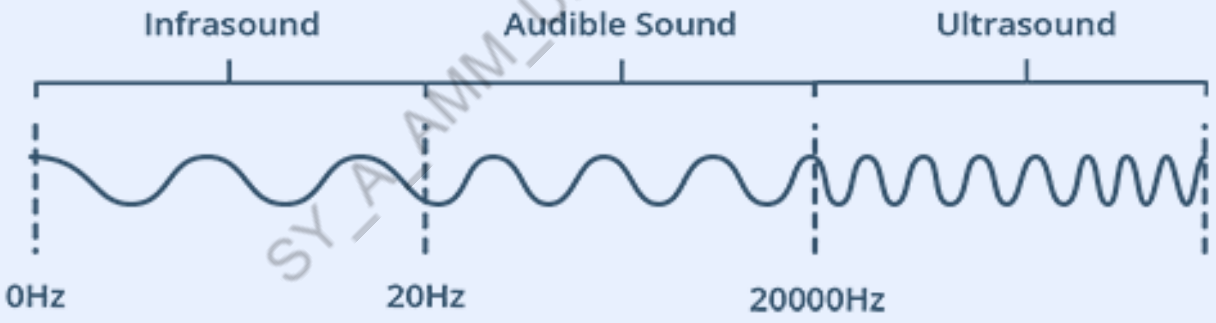
Interfacing 7 segment display (Common anode) to Arduino UNO




```
7  unsigned const int C = 7;
8  unsigned const int H = 6;
9
10
11 void setup(void)
12 {
13   pinMode(A, OUTPUT);
14   pinMode(B, OUTPUT);
15   pinMode(C, OUTPUT);
16   pinMode(D, OUTPUT);
17   pinMode(E, OUTPUT);
18   pinMode(F, OUTPUT);
19   pinMode(G, OUTPUT);
20   pinMode(H, OUTPUT);
21 }
22
23 //My Functions
24
25 void zero(void) {
26   digitalWrite(A, LOW);
27   digitalWrite(B, LOW);
28   digitalWrite(C, LOW);
29   digitalWrite(D, LOW);
30   digitalWrite(E, LOW);
31   digitalWrite(F, LOW);
32   digitalWrite(G, HIGH);
33   digitalWrite(H, HIGH);
34 }
35
```

Ultrasonic Distance Sensor

- Ultrasound is high-pitched sound waves with frequencies higher than the audible limit of human hearing.



SY_A_AMM_UNIT4_SJ

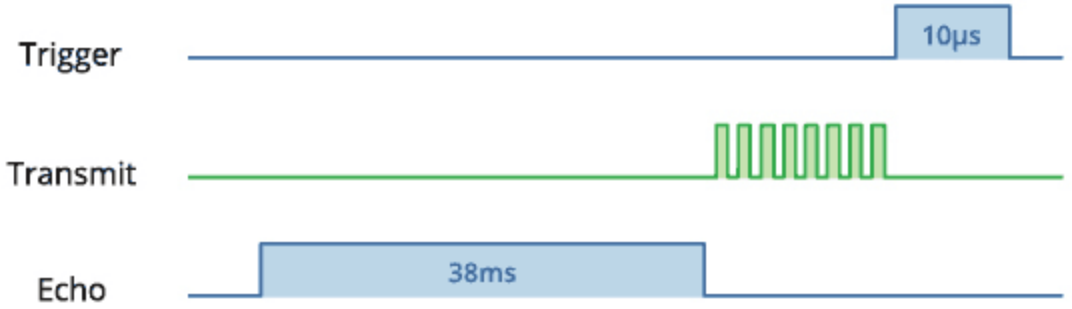
Human ears can hear sound waves that vibrate in the range from about 20 times a second (a deep rumbling noise) to about 20,000 times a second (a high-pitched whistling). However, ultrasound has a frequency of over 20,000 Hz and is therefore inaudible to humans.

Ultrasonic Distance Sensor

- HC-SR04 Ultrasonic distance sensor consists of two ultrasonic transducers.
- The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses.
- The receiver listens for the transmitted pulses. If it receives them it produces an output pulse whose width can be used to determine the distance the pulse travelled.
- when a pulse of at least 10 µS (10 microseconds) in duration is applied to the Trigger pin. In response to that the sensor transmits a sonic burst of eight pulses at 40 KHz
- The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes HIGH to start forming the beginning of the echo-back signal.
- pulses are reflected back the Echo pin goes low as soon as the signal is received. This produces a pulse whose width varies between 150 µS to 25 mS, depending upon the time it took for the signal to be received.

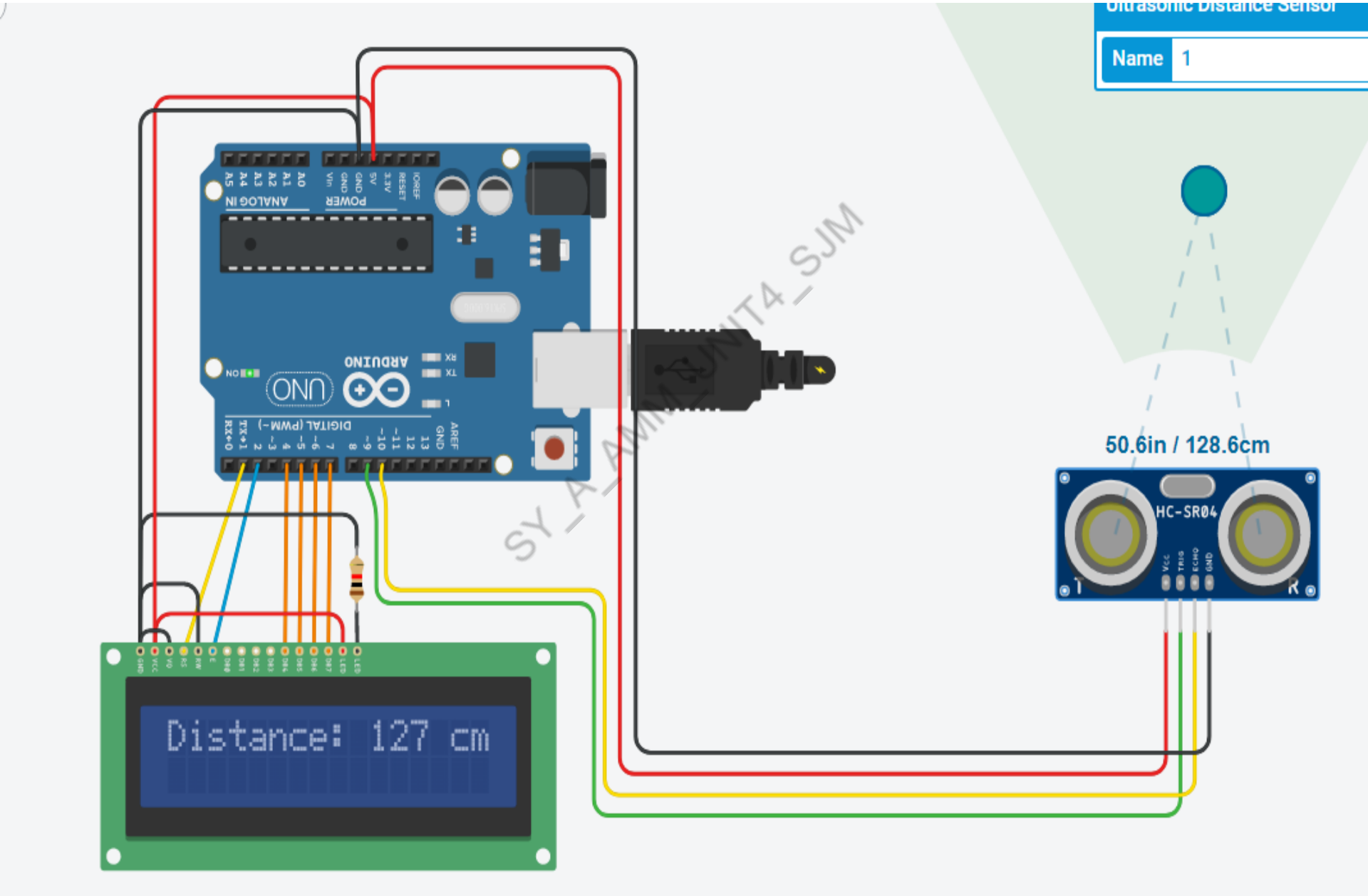
SY_E_AMM_UNIT1_SJM

speed of sound is =340 m/s = 0.034 cm/µs



https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

Ultrasonic Distance Sensor

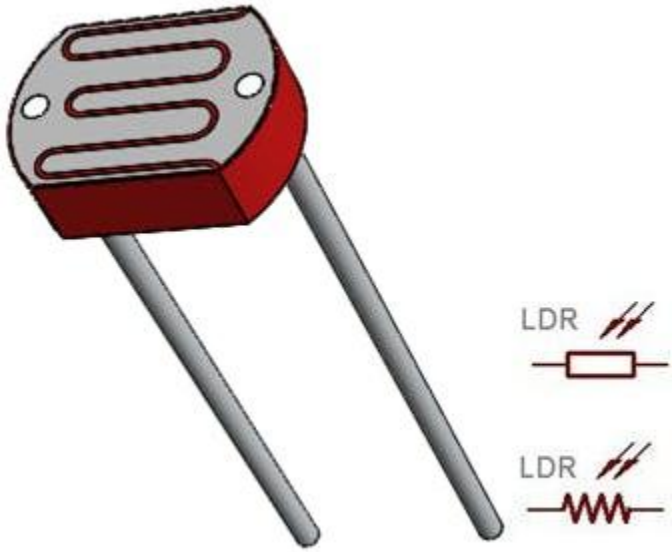


```
#include <LiquidCrystal.h>
LiquidCrystal lcd(1, 2, 4, 5, 6, 7);

const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm;


void setup()
{
lcd.begin(16,2);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}

void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;

lcd.setCursor(0,0);
lcd.print("Distance: ");
lcd.print(distanceCm);
lcd.print(" cm");
delay(10);

}
```

S7_A_AMI_UNIT4_SJM

Detecting Light using LDR

- LDR (light dependent resistor) also called photo resistors are responsive to light.
- Photo resistors are used to indicate the intensity of light.
- When there is **darkness** the resistance of photo resistor **increases** and when there is sufficient light it dramatically decreases.
- LDR (light dependent resistor) which has two terminals. Terminal one is the signal pin which should be connected according to the code. Another terminal is considered as the ground pin which should be connected to the ground of the system.
- Connections of LDR sensor : First terminal should be connected to **analog pin 0 (A0**) of Arduino. Second terminal should be connected any one led of the resistor. Another leg of resistor should be connected to Gnd of Arduino.
- Led connections : Positive pin should be connected to digital pin 5 of Arduino. Another leg of resistor should be connected to Gnd of Arduino.



SY_A_MON_UNIT4_18M

Detecting Light using LDR

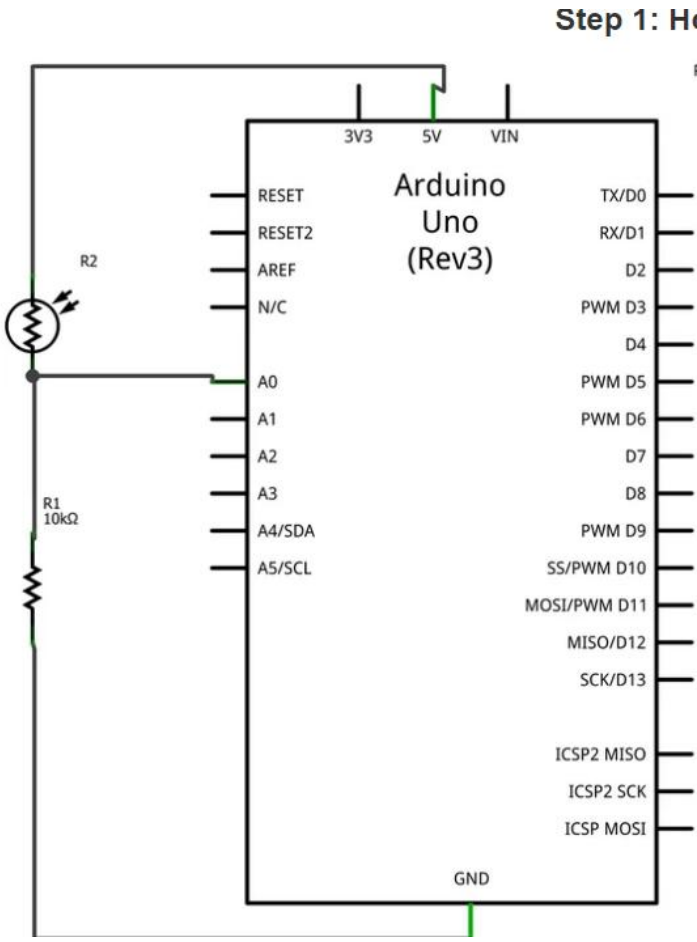
As both resistors are in series the same current must flow through them both.

The LDR resistance drops with light, which causes the current in both resistors to increase (I=V/R), and therefore the voltage across the other (non-LDR) increases.

Va0 = 5 * R1/(R1+R2)

where Va0 is the voltage at A0 pin, R2 is the top resistor value, R1 is the bottom resistor value;

SY_A_AMM_UNIT4_SJM



Detecting Light using LDR

- When there is **darkness** the resistance of photo resistor **increases**
- Example 1 :

R1 = 10k, R2 = 5k =>

Va0 = 5 * 10000/(10000 + 5000)

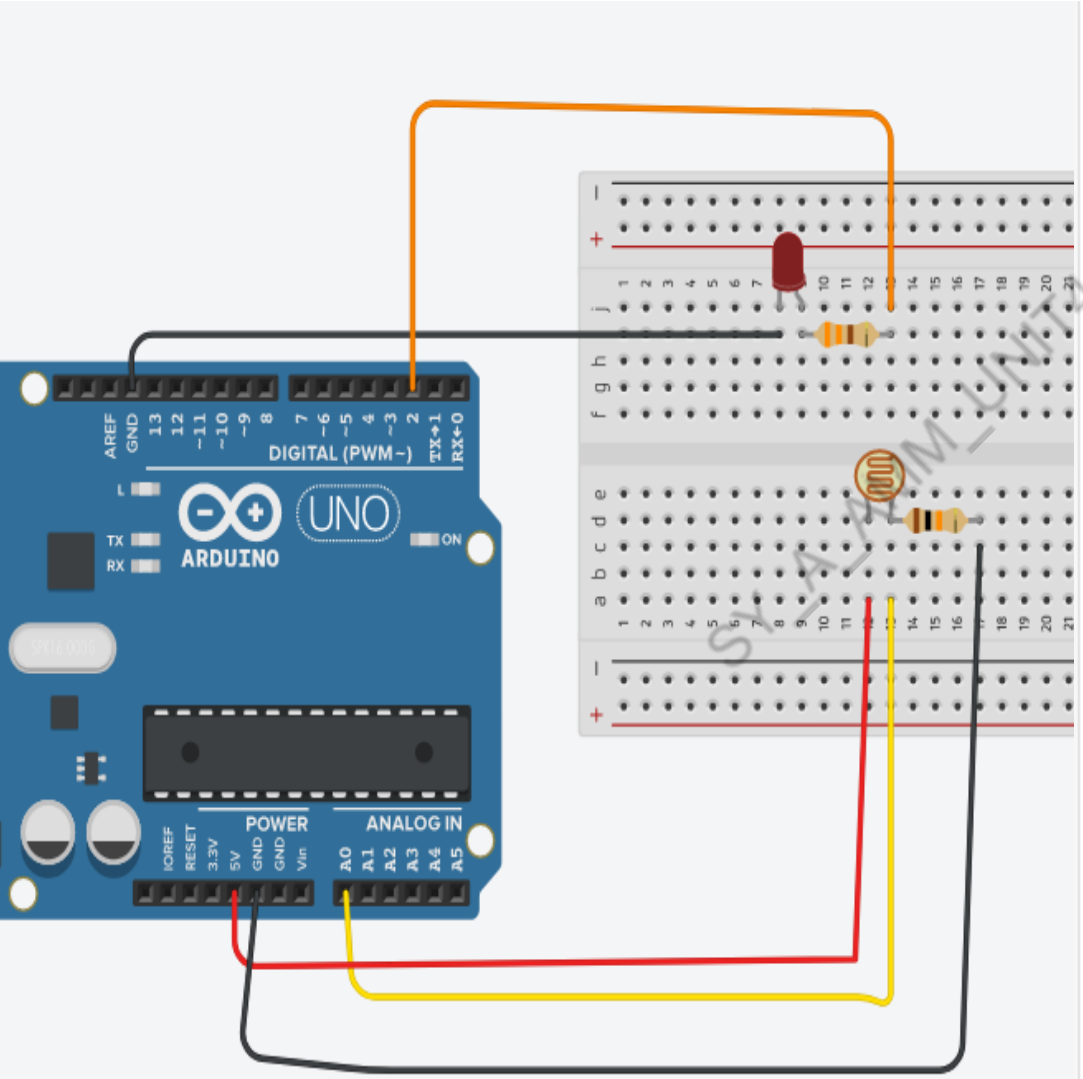
= 5 * 10/15 = 3.33V

- when there is sufficient light it LDR R2 decreases from 5k ohm to 1 k ohm.
- Example 2 :

- e.g. R1 = 10k, R2 = 1k =>
- Va0 = 5 * 10000/(10000 + 1000)
- = 5 * 10/11 = 4.55V

SY_A_MM_UNIT4_SJM

Detecting Light using LDR




```
int photocellPin = 0;
int ledPin = 2;
int photocellReading;
const float limit =900;

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

void loop(void)
{
photocellReading = analogRead(photocellPin);
Serial.print("Analog reading = ");
Serial.println(photocellReading);
if (photocellReading < limit)
{
digitalWrite(ledPin, HIGH);
}
else
{
digitalWrite(ledPin, LOW);
}
delay(1000);
}
```

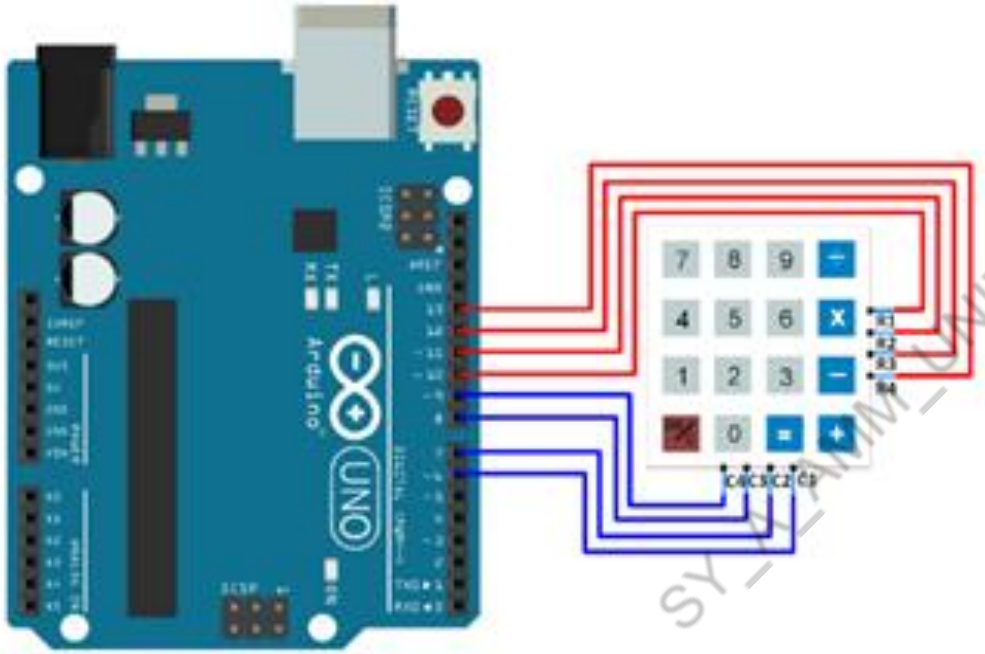
Interfacing 4*4 keypad with Arduino

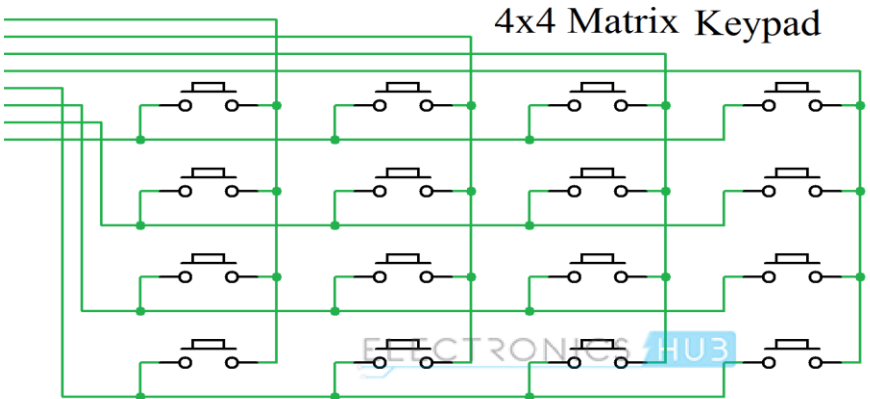
- Keypad is used as an input device to read the key pressed by the user and to process it.
- 4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns.
- A key press establishes a connection betweer the corresponding row and column, between which the switch is placed.

SY_A_AMM_UNIT1_SJM

Interfacing 4*4 keypad with Arduino

Interfacing Diagram





Reading the key pressed on 4x4 keypad and displaying it on the serial terminal of Arduino.

The Arduino UNO board has digital pins 0 and 1 connected to the Tx and Rx pins which are used for serial communication.

Since the sketch uses serial communication to display the key pressed on serial terminal, this will cause erroneous behavior.

To avoid this, use pins other than pins 0 and 1.

SY_EMM_UNIT4_SJM

Interfacing 4*4 keypad with Arduino

- **Functions Used**
  - makeKeymap(keys):

    - This function is used to initialize the internal keymap to be equal to the user defined key map
    - It uses the makeKeymap() function of the Keypad library using they previously defined keys array

Keypad kpd = Keypad(makeKeymap(keys), rowPins, colPins, rows, cols)

- This defines an object kpd of the class Keypad and initializes it.

- rowPins and colPins are the pins on Arduino to which the rows and columns of the keypad are connected to

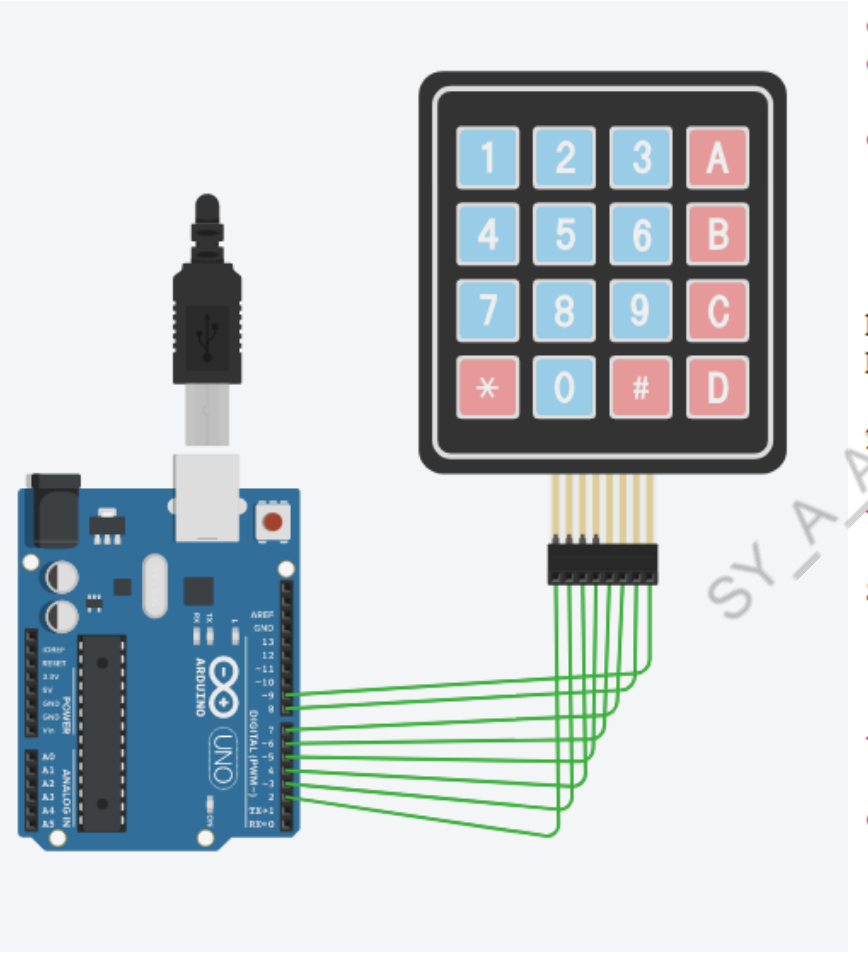
- *rows* and *cols* are the number of rows and columns the keypad has.

kpd.getKey()

- This function is used to identify which key is pressed on the keypad.

SY_A_AMM_UNIT 4_SJM

Interfacing 4*4 keypad with Arduino

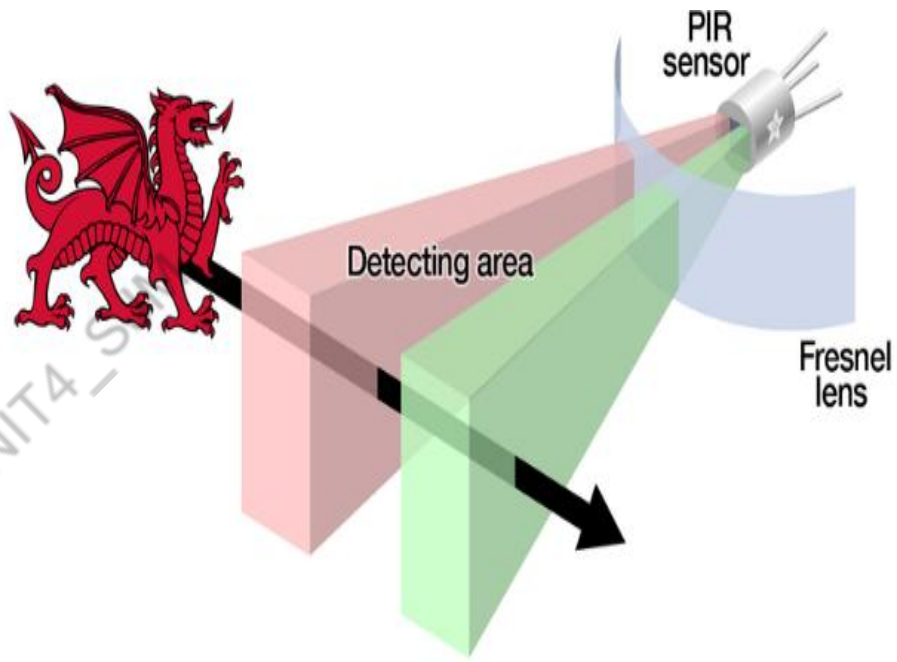


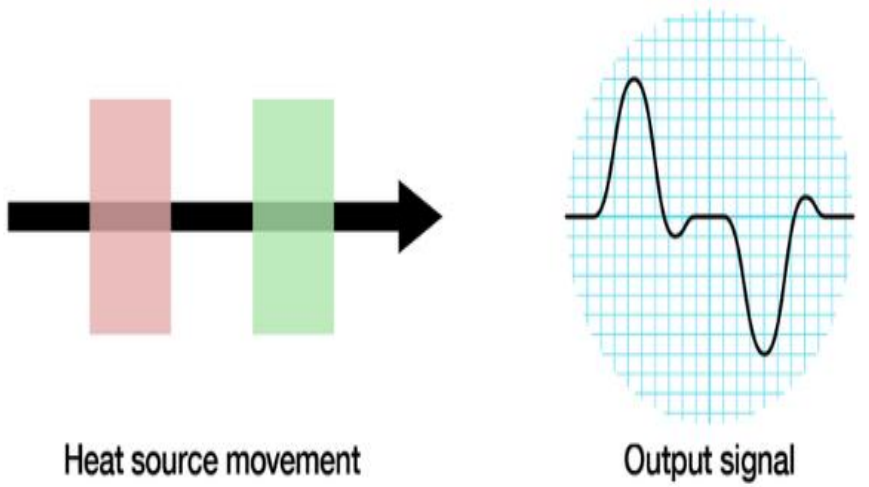

```
#include <Keypad.h>
const byte rows = 4;
const byte cols = 4;

char hex[rows][cols] = {{'1','2','3','A'},
                        {'4','5','6','B'},
                        {'7','8','9','C'},
                        {'*','0','#','D'}
                       };
byte rowpins[rows] = { 2,3,4,5} ;
byte colpins[cols] = {6,7,8,9};

Keypad kpd = Keypad(makeKeymap(hex),rowpins,colpins,rows,cols);

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

}

void loop()
{
char keypressed = kpd.getKey();
  if(keypressed)
  {
  Serial.print(" Key pressed is");
    Serial.println(keypressed);
  }
}
```

SY_A_AMM_UNIT4_SJM

PIR sensor

- It stands for Pyroelectric ("Passive") InfraRed Sensors.
- Pyroelectricity can be described as **the ability of certain materials to generate a temporary voltage when they are heated or cooled**.
- The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR.
- When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room.
- When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a *positive differential* change between the two halves.
- When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.





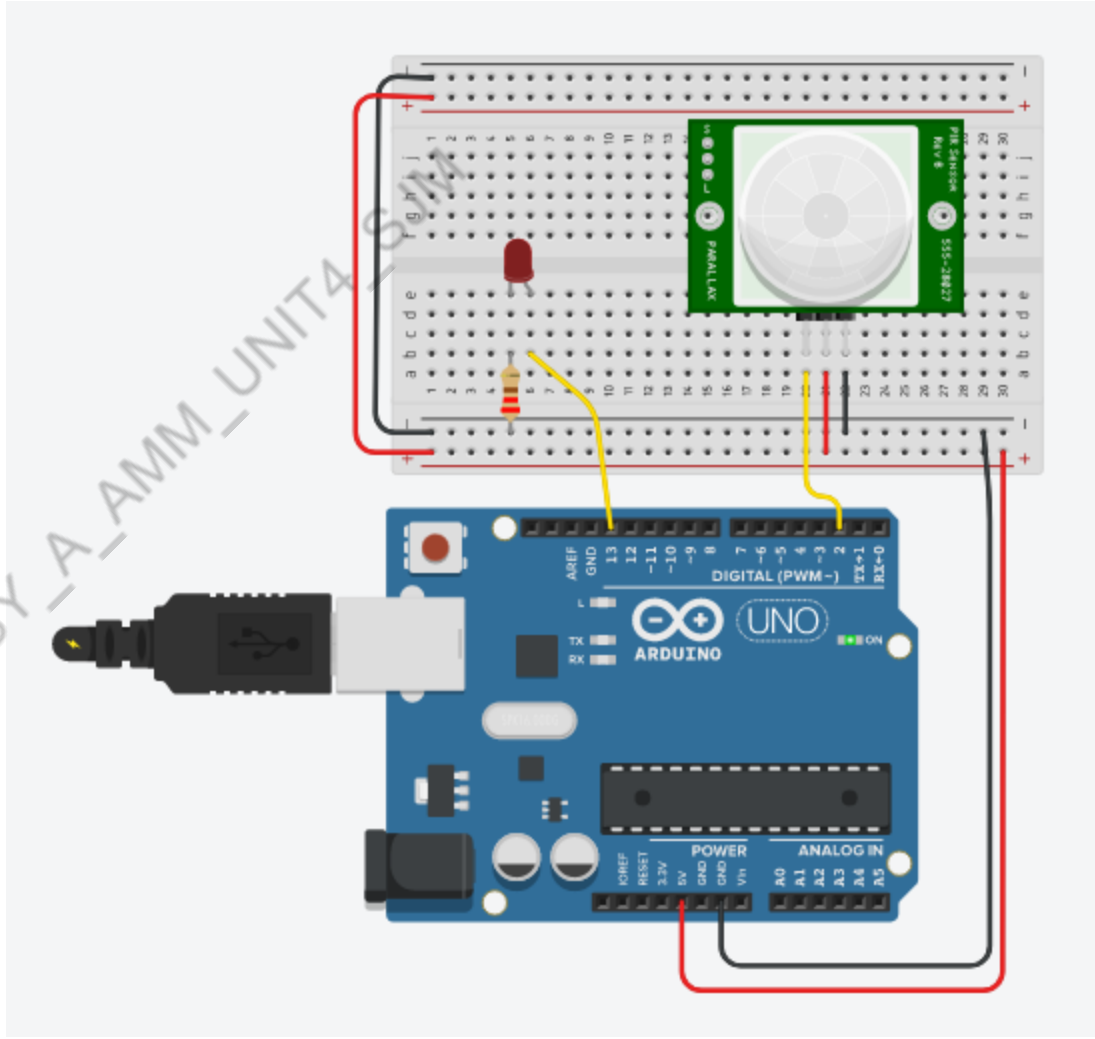
Interfacing PIR sensor with Arduino

- Passive Infra Red sensors can detect movement of objects that radiate IR light (like human bodies).
- The output of PIR motion detection sensor can be connected directly to one of the Arduino (or any microcontroller) digital pins.
- If any motion is detected by the sensor, this pin value will be set to “1”.

SY_ _AIM_UNIT4_SJM

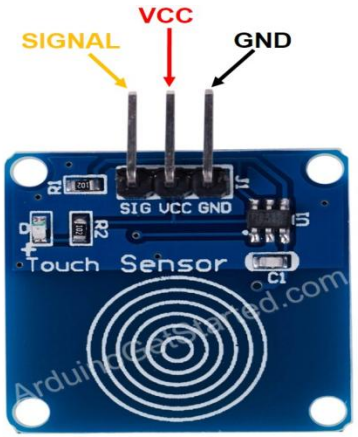
Interfacing PIR sensor with Arduino

```
int sensorState = 0;

void setup()
{
  pinMode(2, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);

}

void loop()
{
sensorState = digitalRead(2);

if (sensorState == HIGH)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Sensor activated!");
  }

 else
    {
    digitalWrite(LED_BUILTIN, LOW);
   }
  delay(10);
}
```

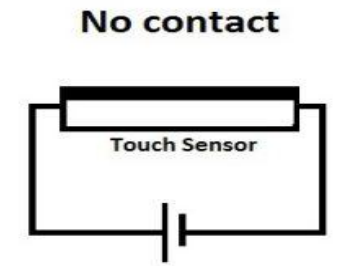


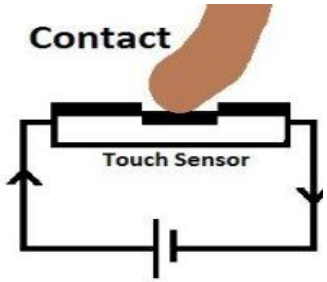
Capacitive Touch Sensor

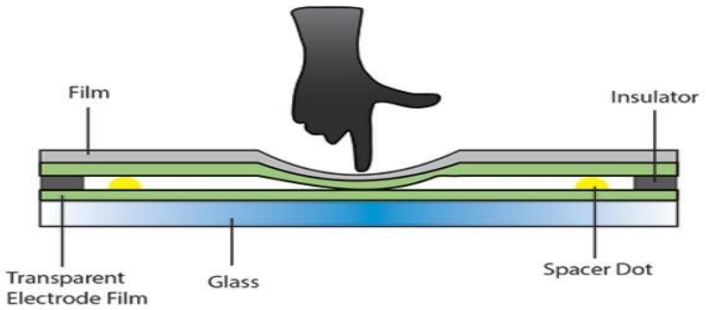
- The sensor plate and your body forms a capacitor. We know that a capacitor stores charge. The more its capacitance, the more charge it can store.(Logic 1)
- The capacitance of this capacitive touch sensor depends on how close your hand is to the plate.
- The **touch sensor's SIGNAL pin** is connected to an Arduino's input **pin.**



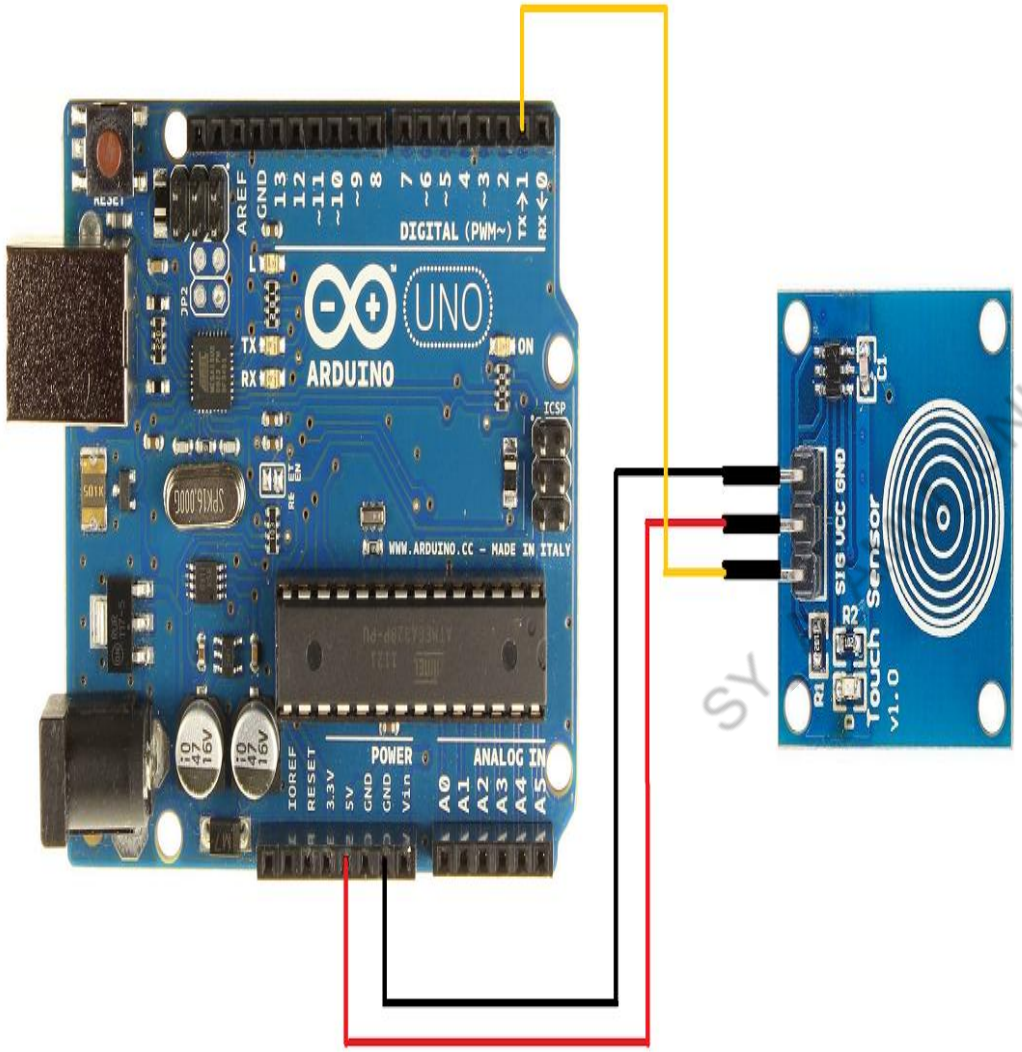
SY_A_AMM_UNIT4_SJM







Touch sensor interfacing with Arduino




```
int sensorpin = 1;
void setup()
{
pinMode(sensorpin , INPUT);
pinMode(ledpin , OUTPUT);
 Serial.begin(9600);
}
void loop()
 {
         int senseValue = digitalRead(sensorPin);

        if (senseValue == HIGH)

       {    digitalWrite(ledPin ,HIGH);
             Serial.println("TOUCHED");
       }
       Else
       {   digitalWrite(ledPin,LOW);
            Serial.println("not touched");
        }

delay(500);
  }
```

Unit 4 Completed

SY_A_AMM_UNIT4_SJM