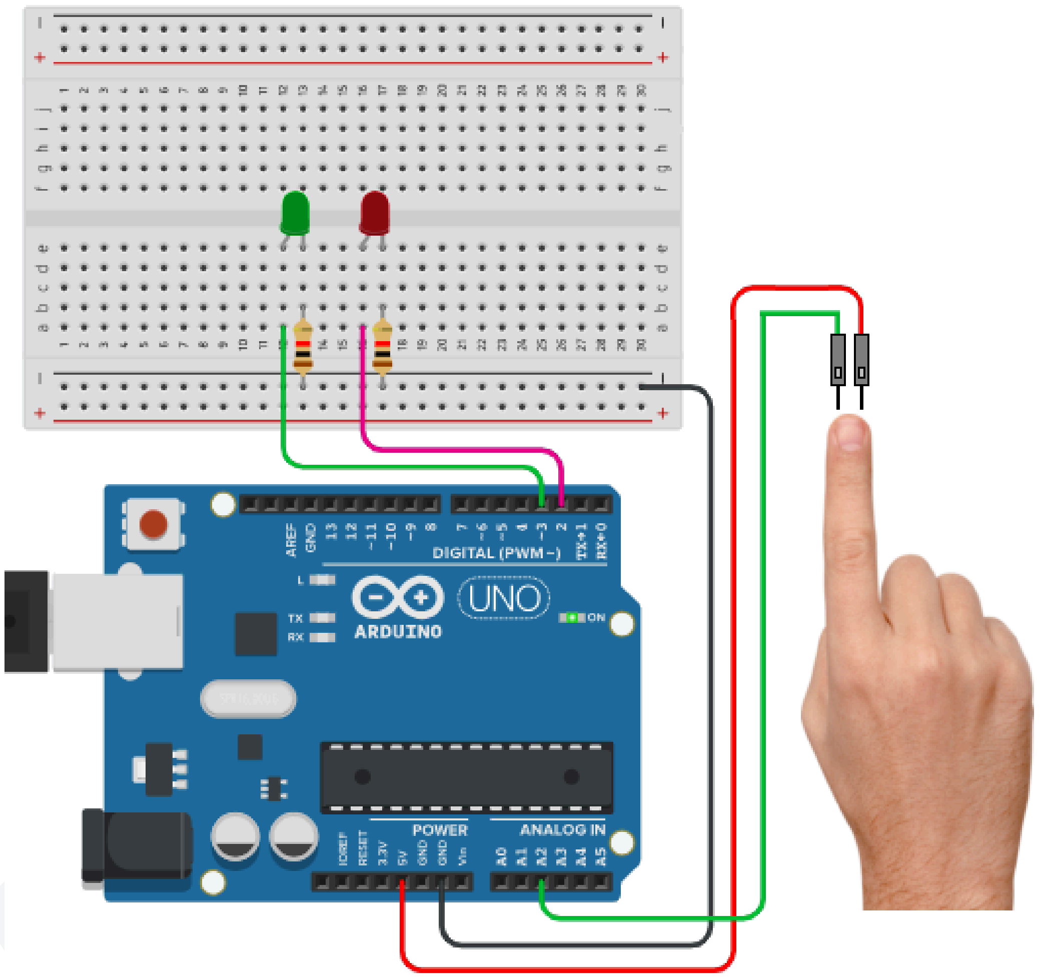
Activity: Understanding Analog Inputs for Arduino

Objective:



This activity is very different from all the Arduino based activities we have done so far. Each of the previous activities made use of "digitalRead();" or "digitalWrite()" to make a given pin HIGH and LOW or to check whether the pin has gone HIGH or LOW. But not all types of inputs have just 2 possible states.

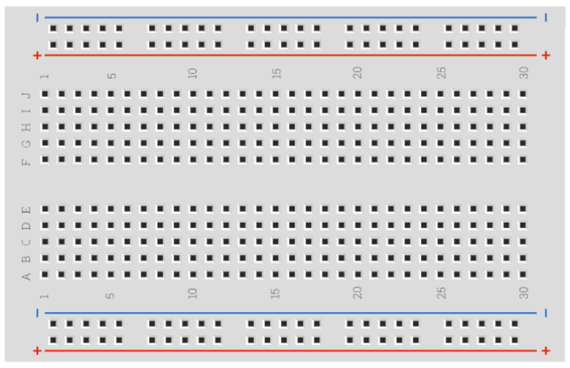
Consider physical entities such as Temperature, Light, Voltage, Soil Moisture, Touch etc... .they all have a continuously varying value and not just 0 and 1.

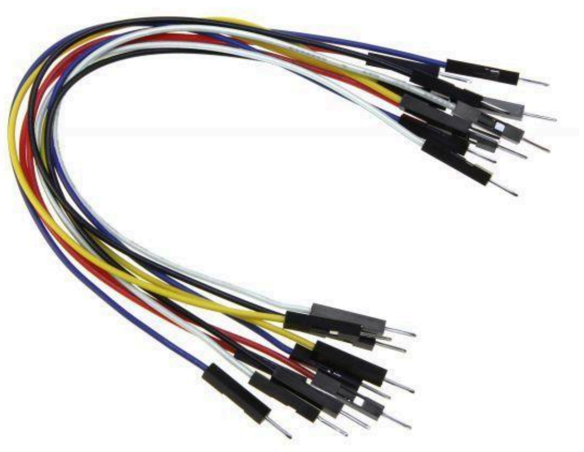
In other words these entities are not Digital but Analog. And so these cannot be read by "digitalRead();". Just as we have 0 and 1 for a digitalRead();, we have values ranging from 0 to 1023 for an "analogRead();"

So, in today's activity we will:

1. Understand Digital and Analog inputs.
2. How Arduino receives and processes analog inputs.
3. Mathematics and physics involved.
4. Simple program to read an analog input pin and send it to the serial monitor.

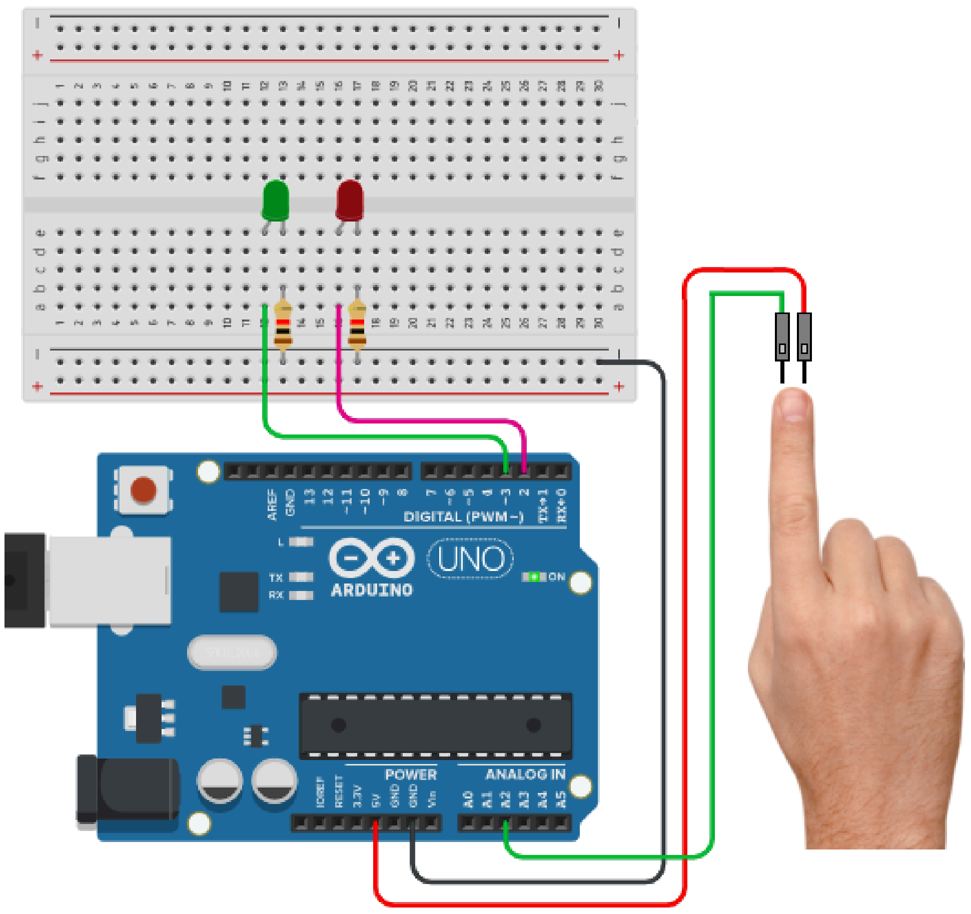
Materials Required:

S.no.	Part	Qty	Image
1	Arduino (UNO / Nano)	1	
2	Breadboard		
3	USB cable for Arduino		

4	M-M Connection wires	10	
5	1k resistor (Brown Black Red Golden)	2	
6	5mm Red LED	1	
7	5mm Green LED	1	

Connection Diagram:

The connection diagram shown here is designed around the Arduino UNO.



Explanation:

This is a rather simple activity to set up as it uses less parts and there is no electronic sensing component specifically used to detect touch. It is just 2 wires, whose metal tips are used as Touch sensitive inputs.

The Touch sensitive terminals are such that one of the wires (Green) is connected directly to Arduino Analog pin 2 (A2). The other touch sensitive terminal is directly connected to +ve of the breadboard.

The Red LED is connected such that its +ve terminal is connected to the Arduino's digital Pin 2 (D2) and the -ve terminal is connected to the 1k resistor. The other side of the 1k resistor is connected to the -ve of the Breadboard.

The Green LED is connected such that its +ve terminal is connected to the Arduino's digital Pin 3 (D3) and the -ve terminal is connected to the 1k resistor. The other side of the 1k resistor is connected to the -ve of the Breadboard.

Lastly, the Arduino's 5v pin is connected to the +ve of the breadboard and the Gnd pin is connected to the -ve of the breadboard.

Arduino Code:

The Arduino code is simple but needs to be done in 2 parts. Let's see why. We have 1 analog input whose value can range from some minimum value to some maximum value. And we have 2 outputs, one of which we want to turn On (and the other remains Off) when the present value of the sensor is below a given **threshold** (just a fancy word for limit). We want the second Output to turn On (and the first remains Off) when the present value of the sensor is above the given threshold.

If we don't know where to place this **threshold**, we won’t be able to set when the Red LED goes On or Off and when the Green LED goes On or Off. So first, we need to find out the value of this **threshold**. For this, we simply send the value of the sensor continuously to the Computer using ”Serial.println();”.

Here is that initial code:

```
#define TOUCH   A2                  //pin A2 named as LDR

int touch_value;                    // variable for touch sensitivity

void setup()
{
  pinMode(TOUCH,INPUT);             // TOUCH pin as Input

  Serial.begin(9600);               // begin serial comm at 9600 bps
}

void loop()
{
  touch_value = analogRead(TOUCH);  //read TOUCH pin and store

  Serial.println(touch_value);      // send stored value to computer
  delay(200);                       // wait for 200ms
}
```

Explanation:

```
#define TOUCH   A2                   //pin A2 named as LDR

int touch_value;                     // variable for touch sensitivity
```

Here, we are telling the Arduino that from now on, we will call pin **A2** as **TOUCH** and then take an Integer type variable “**touch_value**” for storing the analog value.

```
void setup()
{
  pinMode(TOUCH,INPUT);              // TOUCH pin as Input

  Serial.begin(9600);                // begin serial comm at 9600 bps
}
```

Here, inside the setup function, we are declaring that we are going to use TOUCH (pin A2) as Input and that we are beginning a serial communication with the Computer at a speed of 9600 bits per second (bps).

```
void loop()
{
  touch_value = analogRead(TOUCH);  //read TOUCH pin and store

  Serial.println(touch_value);       // send stored value to computer
  delay(200);                        // wait for 200ms
}
```

Here, inside the loop function, we are continuously:

1. Reading the TOUCH pin and storing its value in the “touch_value” variable.
2. Printing the value of this variable on the serial monitor (computer’s USB).
3. Waiting for a short time to make sure the sent value is printed for sure.

Once this code is compiled and uploaded, we will see a continuous stream of live values printed on the serial monitor terminal. These values will change suddenly when we touch the metal tips of the connection wires with our fingertip. These values will again change suddenly when we move our finger away from the touch points.

We are looking for this approximate value on one side of which we will call it touch and on the other side of it we will call it no touch. This value need not necessarily be the same for everyone. It will vary from person to person and place to place, depending upon how moist or dry your hand is.

Let’s say we found this threshold value to be 900. So now, we will turn On Red on one side (greater than) of this value and turn On Green on the other side (less than) of this value.

So, we will use two sets of “if” conditions to compare the present value to the threshold. Then, depending on whether the value is less than or greater than 900 we will “digitalWrite();” the Red and the Green LEDs accordingly.

Given below is the complete Arduino code for doing the process that was just described.

```
#define TOUCH A2    //pin A2 named as TOUCH
#define Red   2     //pin 2 named as Red
#define Green 3     //pin 3 named as Green


int touch_value;    // variable for touch sensitivity



void setup()
{
  pinMode(TOUCH,INPUT);  // TOUCH pin as Input
  pinMode(Red,OUTPUT);   // Red pin as Output
  pinMode(Green,OUTPUT); // Green pin as Output

  Serial.begin(9600);    // begin serial comm at 9600 bps
}



void loop()
{
  touch_value=analogRead(TOUCH);  //read TOUCH pin and store in variable

  Serial.println(touch_value);    // send stored value to computer
  delay(200);                     // wait for 200ms
  if(touch_value > 900)           // if value more than threshold
    {
      digitalWrite(Red,HIGH);     // turn Red LED On
      digitalWrite(Green,LOW);    // turn Green LED Off
    }

  if(touch_value < 900)           // if value less than threshold
    {
      digitalWrite(Red,LOW);      // turn Red LED Off
      digitalWrite(Green,HIGH);   // turn Green LED On
    }
}
```

Here, we added the comparison part using the “if”. One “if” compares if the value is Greater than 900 while the other “if” compares if the value is Less than 900.

Consequently, the code does the following:

1. If the present measured analog value is greater than 900, the Red LED is turned On and the Green LED is turned Off.

2. If the present measured analog value is less than 900, the Red LED is turned Off and the Green LED is turned On.

Outcome and Observations:

1. Analog inputs or analog entities are those which have more than 2 possible values such as Temperature, Light, Distance, Touch, Soil moisture etc. Such inputs are processed by the Arduino using the “analogRead()” function.

2. In an Arduino Nano or UNO, the values of an analogRead() can range from 0 - 1023. Thus we have a total of 1024 possible values for any analog input received by the Arduino.

3. Since Arduino can receive a maximum of 5v as analog input, the resolution comes out to be (5/1024) volt/division. This means if the analogRead(); returns a value of 100 then the voltage on the analog input pin is:

   100 x (5/1024) = 0.488 Volts

   Thus 0 = 0v, 1023 = 5v and 100 = 0.488v and so on.

4. Just like the touch points, any other component can also be placed such as an LDR light sensor, an IR sensor, a water overflow sensor etc. They will work in the same way but with different threshold values.

5. All that is happening here is a change in resistance between the two points. So, anything that causes a change in resistance can be electrically measured using this technique.