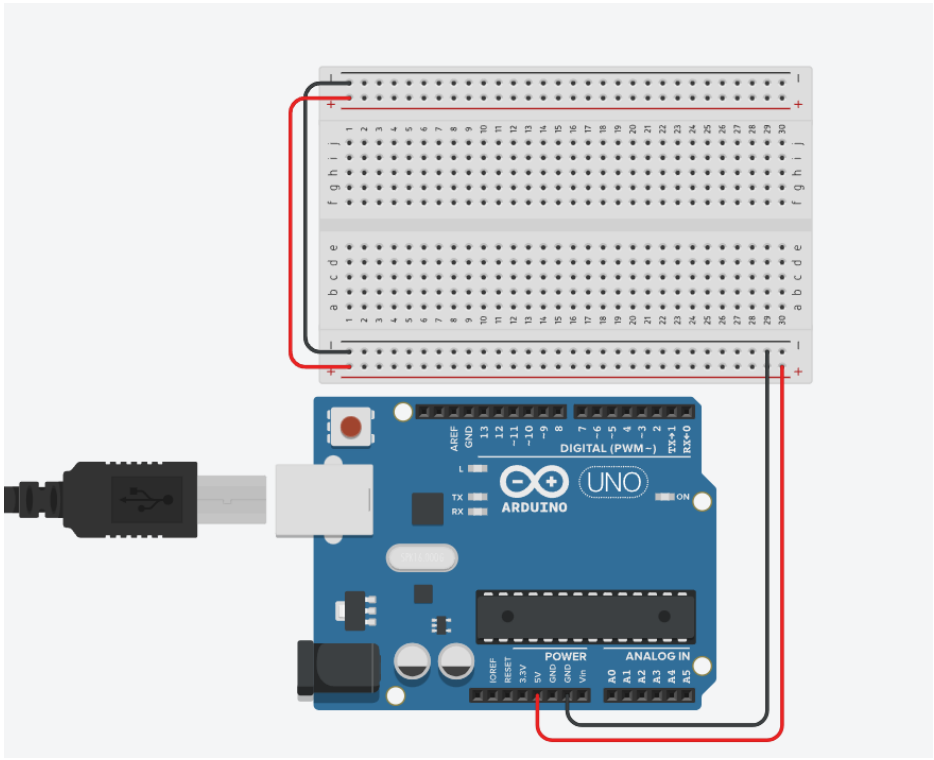
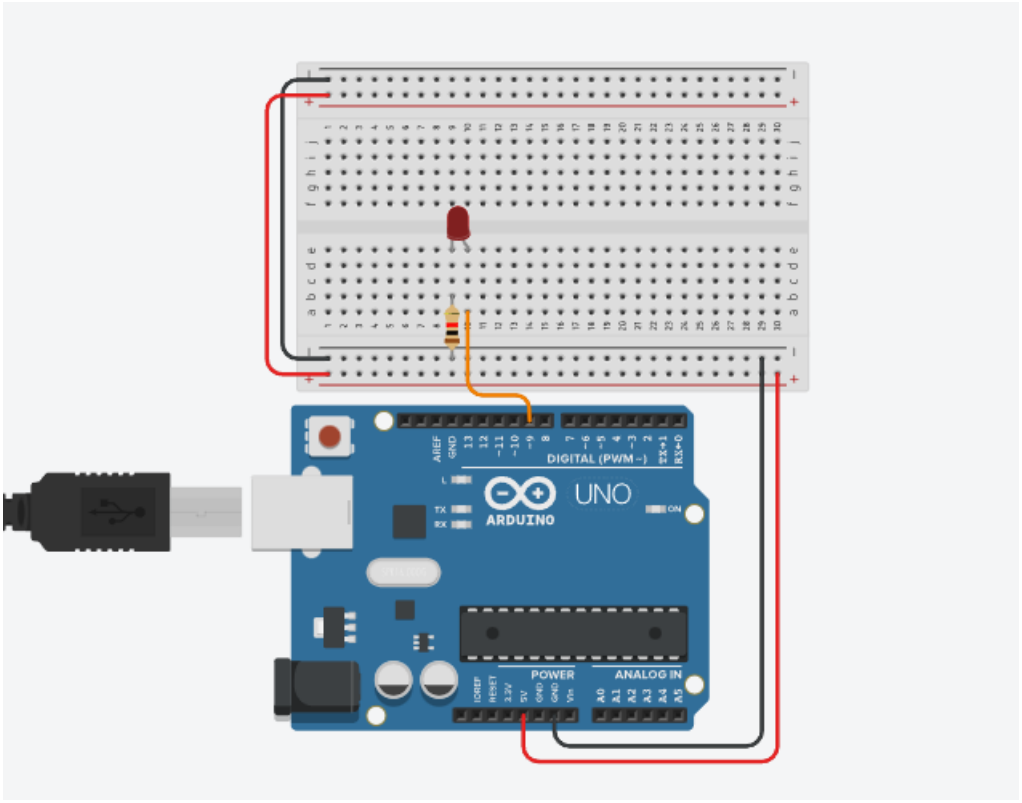
Passive Infrared Sensor (PIR)

A Passive Infrared (PIR) sensor is an electronic device used to detect motion by sensing changes in infrared radiation emitted by objects in its field of view. It is a common component in security systems, motion-activated lighting, and other applications where motion detection is required. When triggered, the sensor sends a signal to activate an alarm or switch on lights, making it a valuable tool for automating and enhancing various systems' security and energy efficiency.

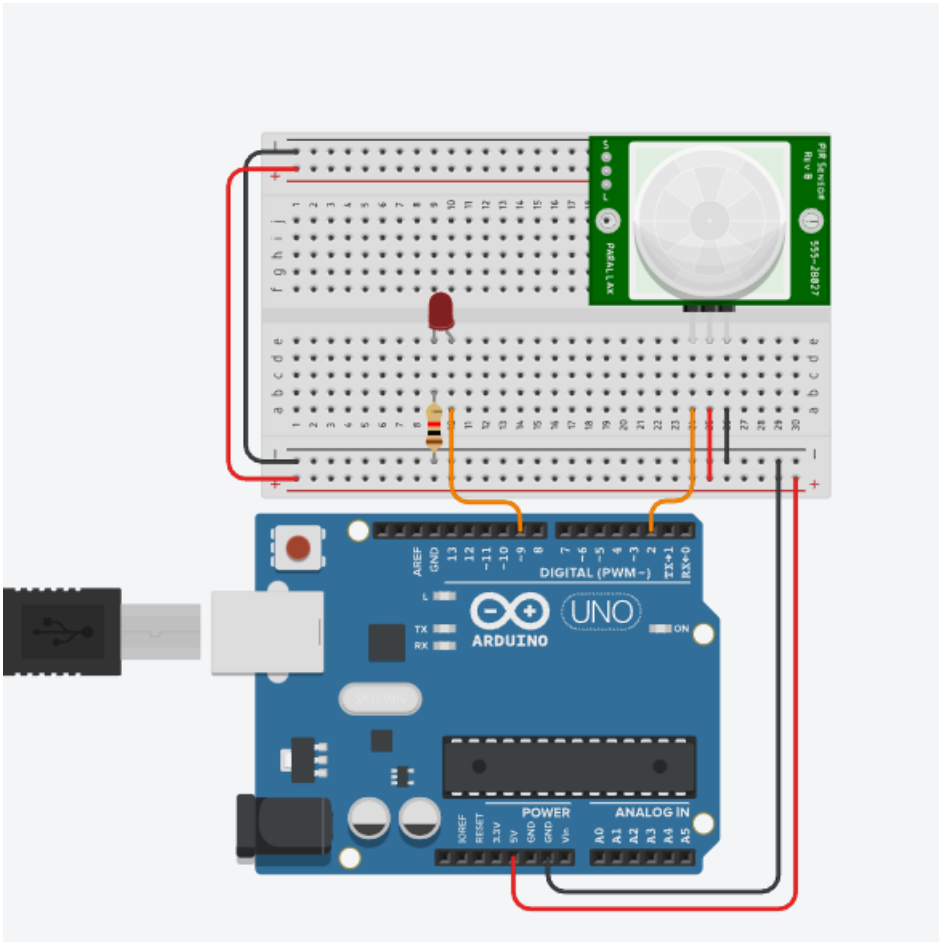
1. Open a new project on Tinkercad and begin developing the following circuit:



2. Connect the breadboard to power and ground from the Arduino. Use the 5v power.



3. Add an LED to the breadboard. Connect the cathode to ground via a 220Ω resistor and connect the anode to pin 9 on the Arduino Uno.



4. Add a PIR component to the breadboard.
    a. The right pin connect to ground.
    b. The middle pin connects to power
    c. The left pin is a signal pin which connects to the Arduino to transfer analogue data.

5. Next, add your code!

Code:

```
// C++ code
//
const int led = 9;
const int pirpin = 2;
//Setting constant variables of components with pin numbers
int pirstate = LOW;
int pirread;
//Setting the pirread variable for reading the value of pir and state variable
void setup()
{
  pinMode(led,OUTPUT);
  pinMode(pirpin,INPUT);
  //Setting led as output and pir as input
  Serial.begin(9600);
}

void loop()
{
  pirread = digitalRead(pirpin);
  //Read pir value and store it in pirread variable
  if (pirread == HIGH){
    digitalWrite(led,HIGH);
    //If pir detects something turn on led
    if (pirstate == LOW){
      Serial.println("Motion Detected");
      pirstate = HIGH;
      //If pir detects something and state is low, print motion, will only print once as we set state to high during detection
    }
  }
  else{
    digitalWrite(led, LOW);
    //If pir isn't detecting anything turn off led
    if (pirstate == HIGH)
    {
      Serial.println("Motion ended!");  // print on output change
      pirstate = LOW;
      //If pir isn't detecting anything and state is high, toggle state back to low
  }
}
}
```

Code Breakdown:

```
// C++ code
//
const int led = 9;
const int pirpin = 2;
//Setting constant variables of components with pin numbers
int pirstate = LOW;
int pirread;
//Setting the pirread variable for reading the value of pir and state variable
void setup()
{
  pinMode(led,OUTPUT);
  pinMode(pirpin,INPUT);
  //Setting led as output and pir as input
  Serial.begin(9600);
}
```

- Create constant variables for the LED and PIR using their Arduino pin numbers.
- Create two more variables, one for the current PIR state and one for the PIR read value.
- Set the LED as an output and the PIR as an input.

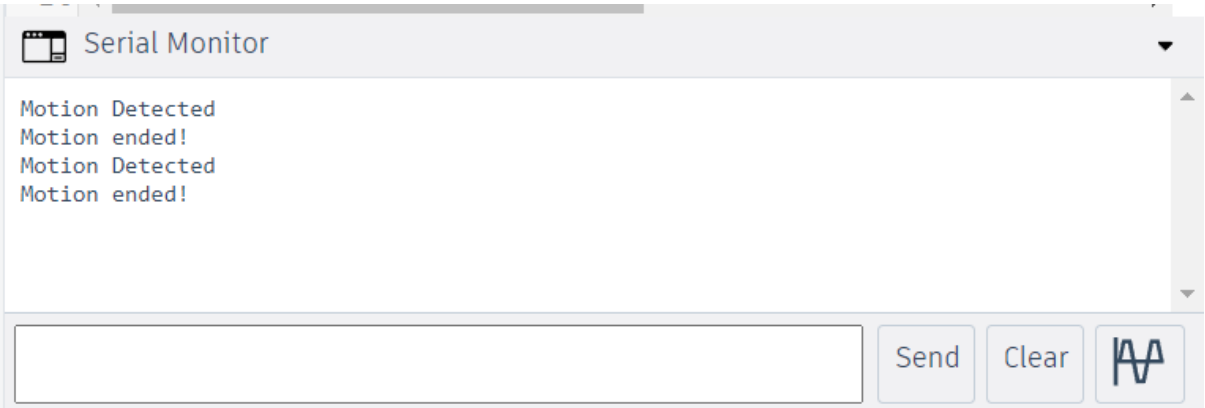
- Initialise serial communication with a baud rate of 9600

```
void loop()
{
  pirread = digitalRead(pirpin);
  //Read pir value and store it in pirread variable
  if (pirread == HIGH){
    digitalWrite(led,HIGH);
    //If pir detects something turn on led
    if (pirstate == LOW){
      Serial.println("Motion Detected");
      pirstate = HIGH;
      //If pir detects something and state is low, print motion,
    }
  }
```

- For the loop section, we start by using a digital read to store the value of the PIR, we use digital to indicate whether something is moving or not (i.e. 0 or 1).
- We create an IF statement to determine movements (if pirread == HIGH, the LED turns on)
- The following IF statement prints to the serial monitor that motion is detected, pirstate is then set to HIGH so it can't go back into the if statement multiple times and repeatedly print on the serial monitor. Hence, this method is used to only print once on the monitor.

```
  else{
    digitalWrite(led, LOW);
    //If pir isn't detecting anything turn off led
    if (pirstate == HIGH)
    {
      Serial.println("Motion ended!");  // print on output change
      pirstate = LOW;
      //If pir isn't detecting anything and state is high, toggle state back to low
  }
}
}
```

- If the PIR does not detect any movement, the LED will turn off with the digital write low.
- The conditional statement above prints 'Motion ended!' on the serial monitor and sets the state back to low. This method also prints once as shown below.



6. Now that you have gained experience to operate the FSR, LDR and PIR sensors. Integrate all 3 sensors into one circuit and modify the code to monitor their readings on the serial monitor and through the LEDs.