PRACTICAL 1

Introduction to basic electronic components used in IOT using a Simulator (Tinkercad).

- **Basic Components:**

1. **Battery =** A battery is an electronic component that pushes electric current through a circuit. Batteries are available in different voltages, such as 1.5V and 9V, and are labeled with their voltage. The voltage indicates the pressure of the power source in the electronic circuit.

Fig. - Battery

2. **Resistor =** A resistor is an electrical component that controls the flow of electrical current in a circuit. It can also provide a specific voltage for an active device.

Fig. - Resistor

3. **LED =** The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes: a positive electrode and a negative one. It lights up only when a forward current passes, and it can flash red, blue, green, yellow, etc.

Fig. – LED

4. **Capacitor =** A capacitor in Tinkercad is an electrical component that stores and releases current in a circuit. Capacitors smooth out the current supplied to components, and gradually reduce the current supplied to a component. They are similar to batteries, but store less current and don't need to be replaced.

Fig. - Capacitor

PRACTICAL 2

Introduction to Arduino its pin diagram and its components.

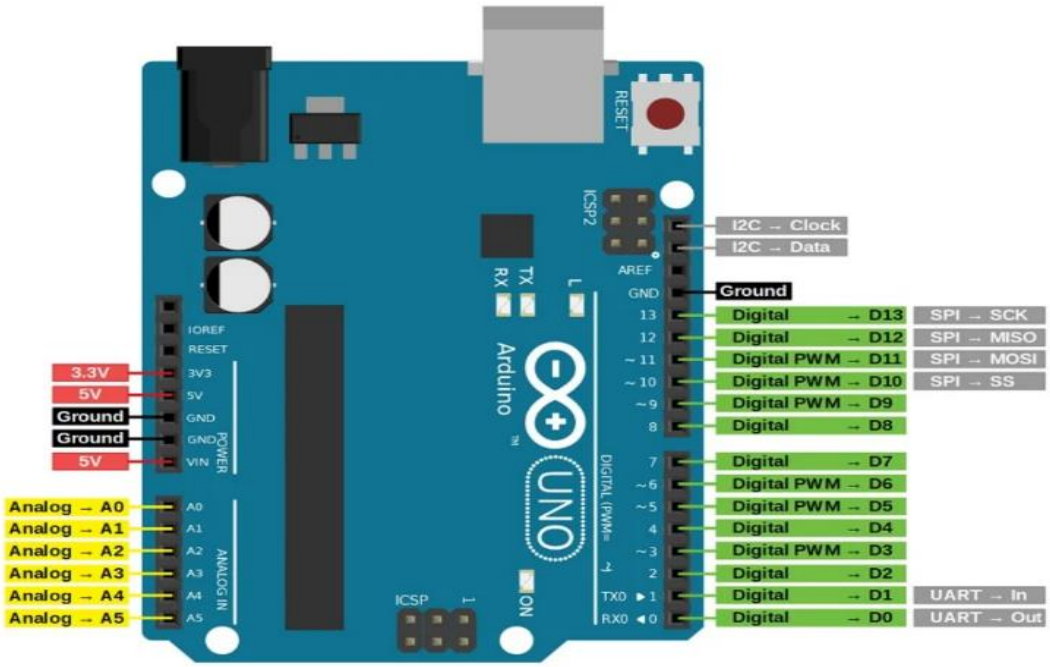


Fig. - Arduino

Arduino is a software as well as hardware platform that helps in making electronic projects. It is an open-source platform and has a variety of controllers and microprocessors.

There are various types of Arduino boards used for various purposes. The Arduino is a single circuit board, which consists of different interfaces or parts. The board consists of the set of digital and analog pins that are used to connect various devices and components, which we want to use for the functioning of the electronic devices.

- **Arduino Pin Diagram Components:**

1. Digital Pins (D0 - D13)

- **Function**: Used for digital I/O (input/output).
- **Usage**: Can read digital states (HIGH or LOW) or output 5V signals (HIGH) or 0V (LOW).

- **PWM Pins**: Pins with a "~" symbol (usually D3, D5, D6, D9, D10, D11) support Pulse Width Modulation (PWM) to simulate analog output.

2. Analog Pins (A0 - A5)

- **Function**: Used to read analog signals, such as those from sensors.
- **Range**: Converts analog signals to a 10-bit digital number (0-1023).

3. Power Pins

- **Vin**: The input voltage to the Arduino when using an external power source.
- **5V**: Powers most Arduino components, provided by USB or external adapter.
- **3.3V**: Some sensors and modules need 3.3V instead of 5V.
- **GND (Ground)**: The common ground for circuits.

4. Reset Pin

- **Function**: Resets the board when brought LOW.
- **Usage**: Connect to a button or switch to reset your program.

5. AREF (Analog Reference)

- **Function**: Provides a reference voltage for analog inputs.
- **Usage**: Used to set a custom reference voltage for analogRead().

Key Components on Arduino Board:

1. **Microcontroller (ATmega328P)**:

- The "brain" of the Arduino that executes code.

2. **USB Port**:

- **Purpose**: Used for programming the board and providing power from a computer.

3. **Power Jack**:

- Connects an external power source (typically 7-12V).

4. **Voltage Regulator**:

- Controls and stabilizes voltage coming into the board.

5. **Crystal Oscillator**:

- Ensures the microcontroller runs at a stable speed (typically 16 MHz).

6. **LED Indicators**:

- **ON**: Indicates power to the board.
- **L (Pin 13)**: Linked to pin 13, can be used for testing.
- **TX/RX**: Blinks during data transmission/reception (Serial communication).

7. **Reset Button**:

- Used to reset the board manually.

PRACTICAL 3

Introduction to Raspberry pi its pin diagram and its components.

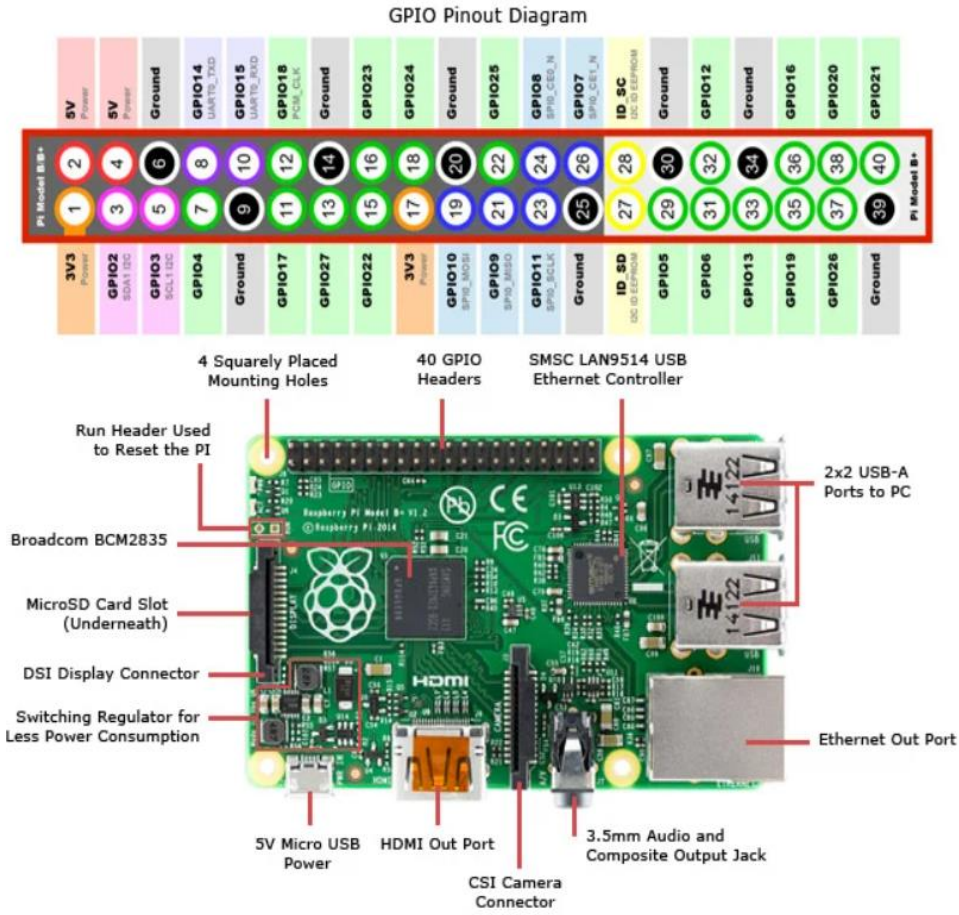


Fig. – Raspberry Pi

Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer. Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications. Raspberry Pi is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at a low power consumption.

Raspberry Pi Pin Diagram Components

1. 40-Pin GPIO Header

- **Function**: Used to connect various sensors, modules, and circuits.
- **Pins**:

- **Power Pins**: 5V and 3.3V power output pins.
- **Ground (GND)**: Several GND pins for common ground connections.
- **GPIO Pins**: Configurable for digital input or output. Many are programmable to support I2C, SPI, and UART protocols.

2. MicroSD Card Slot

- **Purpose**: Stores the operating system and all files. This is where you insert the microSD card with the Raspberry Pi OS.

3. USB Ports (USB 3.0 and USB 2.0)

- **Purpose**: Connect USB devices like keyboards, mice, external storage, and Wi-Fi dongles.
- **Types**:
  - **USB 3.0 Ports (blue)**: Faster transfer speeds, ideal for high-speed devices.
  - **USB 2.0 Ports (black)**: Standard USB ports for lower-speed devices.

4. HDMI Ports (Micro-HDMI)

- **Function**: Connects displays, like monitors or TVs.
- **Dual Ports**: Raspberry Pi 4 has two micro-HDMI ports, supporting dual displays up to 4K resolution.

5. Ethernet Port

- **Purpose**: Provides a wired network connection.
- **Speed**: Up to 1 Gbps, suitable for high-speed internet access and networking.

6. CSI and DSI Ports

- **CSI (Camera Serial Interface)**: Connects Raspberry Pi Camera Module.

- **DSI (Display Serial Interface)**: Connects the official Raspberry Pi touchscreen display.

7. Audio and Composite Video Jack (3.5mm)

- **Function**: Outputs analog audio and composite video signals, useful for audio output and connecting to older displays.

8. USB-C Power Input

- **Function**: Supplies power to the board.

- **Voltage**: Requires a 5V, 3A USB-C power supply.

9. SoC (System on Chip) – Broadcom Processor

- **Purpose**: Acts as the CPU, containing ARM cores, GPU, and other essential components.

- **Model**: Raspberry Pi 4 uses a Broadcom BCM2711 chip, a quad-core ARM Cortex-A72 processor.

10. RAM (Random Access Memory)

- **Purpose**: Provides memory for running applications and OS.

- **Options**: Available in 2GB, 4GB, and 8GB configurations in the Raspberry Pi 4.

11. Wireless Communication Chips

- **Wi-Fi and Bluetooth**: Built-in Wi-Fi (2.4 GHz and 5 GHz) and Bluetooth 5.0 support.

12. LED Indicators

- **PWR**: Indicates if the Pi is powered on.

- **ACT**: Shows SD card activity, blinking when data is read/written.

PRACTICAL 4

Introduction to tinkercad and make basic(non-programmable) connections on tinkercad circuit simulation.

Tinkercad is a free-of-charge, online 3D modeling program that runs in a web browser. Since it became available in 2011 it has become a popular platform for creating models for 3D printing as well as an entry-level introduction to constructive solid geometry in schools.

- **Steps of Connections:**

1. **Place the Battery:**
    - In Tinkercad, drag a 9V battery onto the workspace which have positive (+) and a negative (−) terminal.
2. **Connect the Resistor to the LED:**
    - Place a resistor on the breadboard or connect it directly to the LED.
    - Connect one end of the resistor to the anode (longer leg) of the LED. The resistor limits the current to prevent the LED from burning out.
3. **Complete the LED Circuit:**
    - Connect the cathode (shorter leg) of the LED directly to the negative terminal of the battery. This creates a path for the current to return to the battery.
4. **Connect the Battery:**
    - Connect the positive terminal of the battery to the free end of the resistor.
    - Connect the negative terminal of the battery to the cathode of the LED (completing the circuit).

OUTPUT: -

Before Simulation **After Simulation**

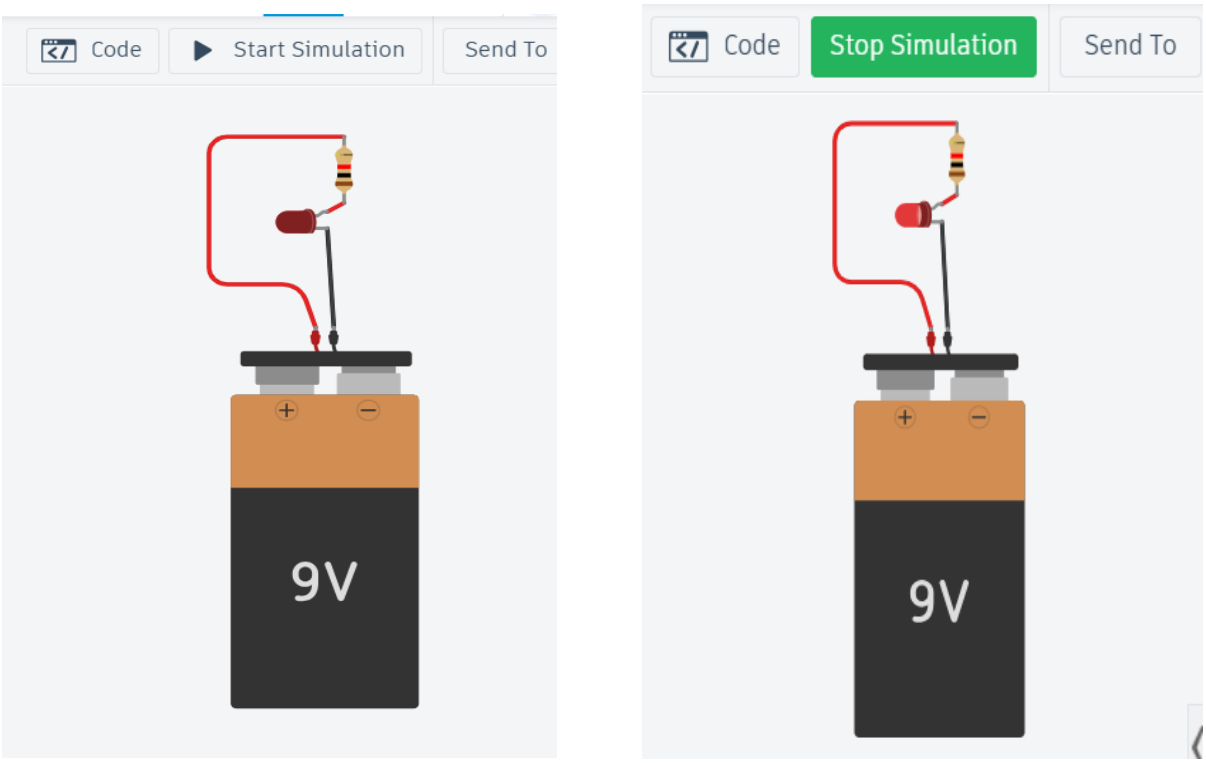


Fig. – Basic connections and simulation using Tinkercad

PRACTICAL 5

Write a program using Arduino for Blink LED's.

- **Steps for Connections:**

1. **Identify the LED Legs:**

    - The longer leg is the positive (anode), which connects to the digital output pin of the Arduino.

    - The shorter leg is the negative (cathode), which connects to the ground (GND) pin.

2. **Place the LED on the Breadboard (optional):**

    - Insert the two legs of the LED into separate rows on the breadboard.

3. **Connect the Resistor:**

    - Connect a 220-ohm resistor in series with the positive leg of the LED (anode).

    - One end of the resistor should connect to the positive leg, and the other end will go to the Arduino pin.

4. **Connect the LED to the Arduino:**

    - Connect the positive leg of the LED (with the resistor) to digital pin 13 (or any other digital pin you plan to use).

    - Connect the negative leg of the LED to the GND pin on the Arduino.

- **Text Code:**

```
// C++ code
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
```

```
}
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100000);
}
```

OUTPUT: -

Before Simulation **After Simulation**

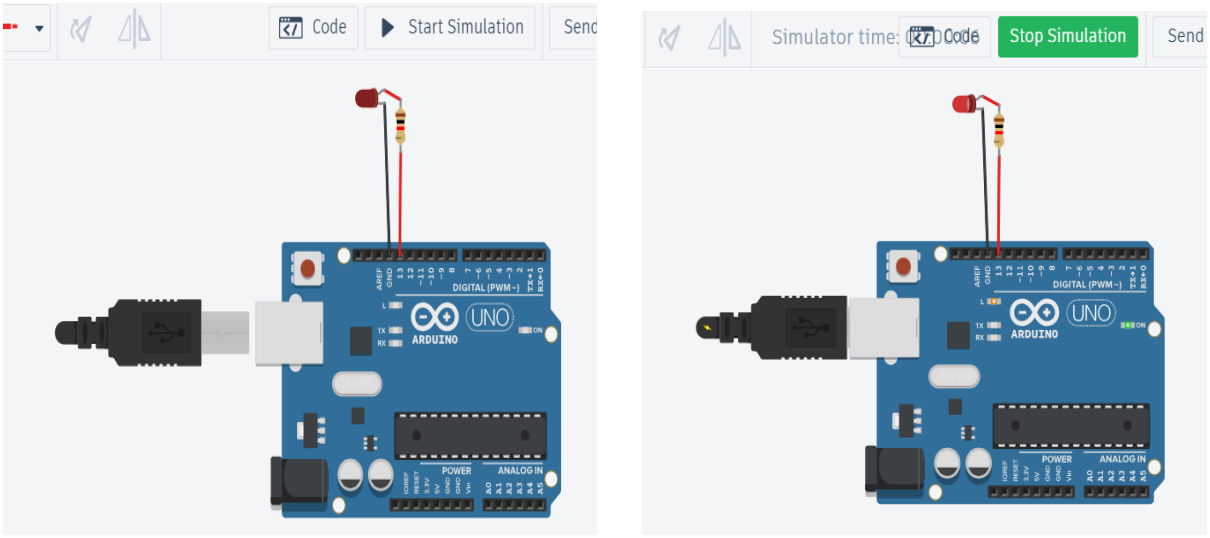


Fig. – Connection & Simulation of blink LED

PRACTICAL 6

Write a program to show how to fade an LED on PWM pin of Arduino using analogWrite() function and also attach oscilloscope for graphical representation.

- **Steps for Connection:**

1. **Prepare the LED:**

    - Identify the positive (anode) and negative (cathode) legs of the LED.

    - The longer leg is the anode (positive), and the shorter leg is the cathode (negative).

2. **Place the LED and Resistor on the Breadboard:**

    - Insert the LED into the breadboard (if using one).

    - Connect a 220-ohm resistor in series with the positive leg (anode) of the LED to limit current.

3. **Connect the LED to the Arduino:**

    - Connect one end of the resistor to digital pin 13 (or another digital pin) on the Arduino.

    - Connect the negative leg (cathode) of the LED to the GND pin on the Arduino.

4. **Connect the Oscilloscope Probes:**

    - Positive (probe): Connect the oscilloscope's positive probe to the digital pin 13 (or whichever pin the LED is connected to) on the Arduino. This will allow you to see the signal sent from the Arduino to the LED.

    - Ground (negative): Connect the oscilloscope's ground clip to the GND pin on the Arduino to complete the circuit for measurement.

- **Text Code:**

```
int i = 0;
int brightness = 0;
```

```
void setup()
{
  pinMode(9, OUTPUT);
}
void loop()
{
  for (brightness = 0; brightness <= 255; brightness += 5) {
    analogWrite(9, brightness);
    delay(10); // Wait for 30 millisecond(s)
  }
  for (brightness = 255; brightness >= 0; brightness -= 5) {
    analogWrite(9, brightness);
    delay(10); // Wait for 30 millisecond(s)
  }
}
```

OUTPUT: -

Before Simulation **After Simulation**

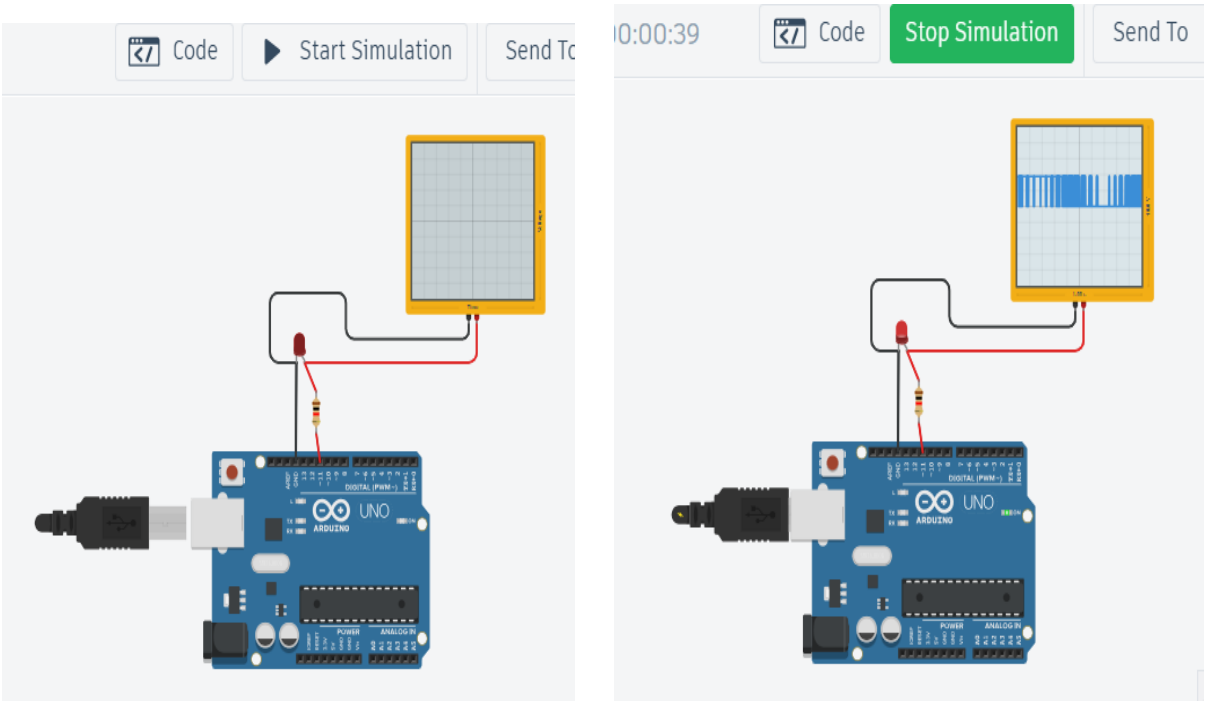


Fig. – Connection of Arduino, Oscilloscope & LED

PRACTICAL 7

Control directions (clockwise/anticlock) and speed of hobbygearmotor using L293D IC with Arduino board.

- **Steps for Connections**

1. **Set Up the L293D on the Breadboard (if using one):**

    - Place the L293D IC on the breadboard, making sure each pin is accessible. The IC has 16 pins, with a small notch at the top for orientation.

2. **Connect the Motor to the L293D:**

    - Pin 3 (OUTPUT 1): Connect to one terminal of the motor.

    - Pin 6 (OUTPUT 2): Connect to the other terminal of the motor.

    - This setup will allow the L293D to control the motor connected across these two pins.

3. **Connect the L293D to Power:**

    - Pin 1 (ENABLE 1-2): Connect to 5V on the Arduino. This enables the first H-bridge that controls your motor. You could also connect it to a PWM pin to control the motor speed.

    - Pin 4 and Pin 5 (GND): Connect to GND on the Arduino.

    - Pin 8 (VCC2): Connect to the positive terminal of the external power source (e.g., 5-12V) to power the motor. This voltage should match the motor's operating voltage.

    - Pin 16 (VCC1): Connect to 5V on the Arduino. This pin powers the internal logic of the L293D.

4. **Connect the Arduino Control Pins:**

    - Pin 2 (INPUT 1): Connect to digital pin 8 on the Arduino (to control motor direction).

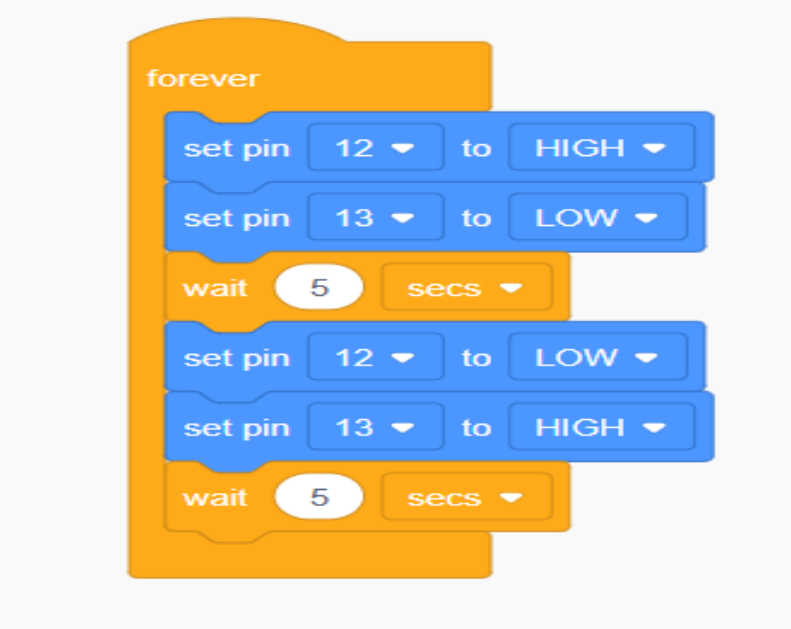
- Pin 7 (INPUT 2): Connect to digital pin 9 on the Arduino (to control motor direction).

- These input pins will control the direction of the motor by controlling the high and low signals.

5. **Connect Ground:**

- Connect the GND of the external power source to the GND on the Arduino and GND pins (4 and 5) on the L293D to ensure a common ground for the circuit.

- **Block Code:**



- **Text Code:**

```
void setup()
{
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop()
{
  digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
```

```
  delay(3000);
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  delay(3000);
}
```

OUTPUT: -

Before Simulation **After Simulation**

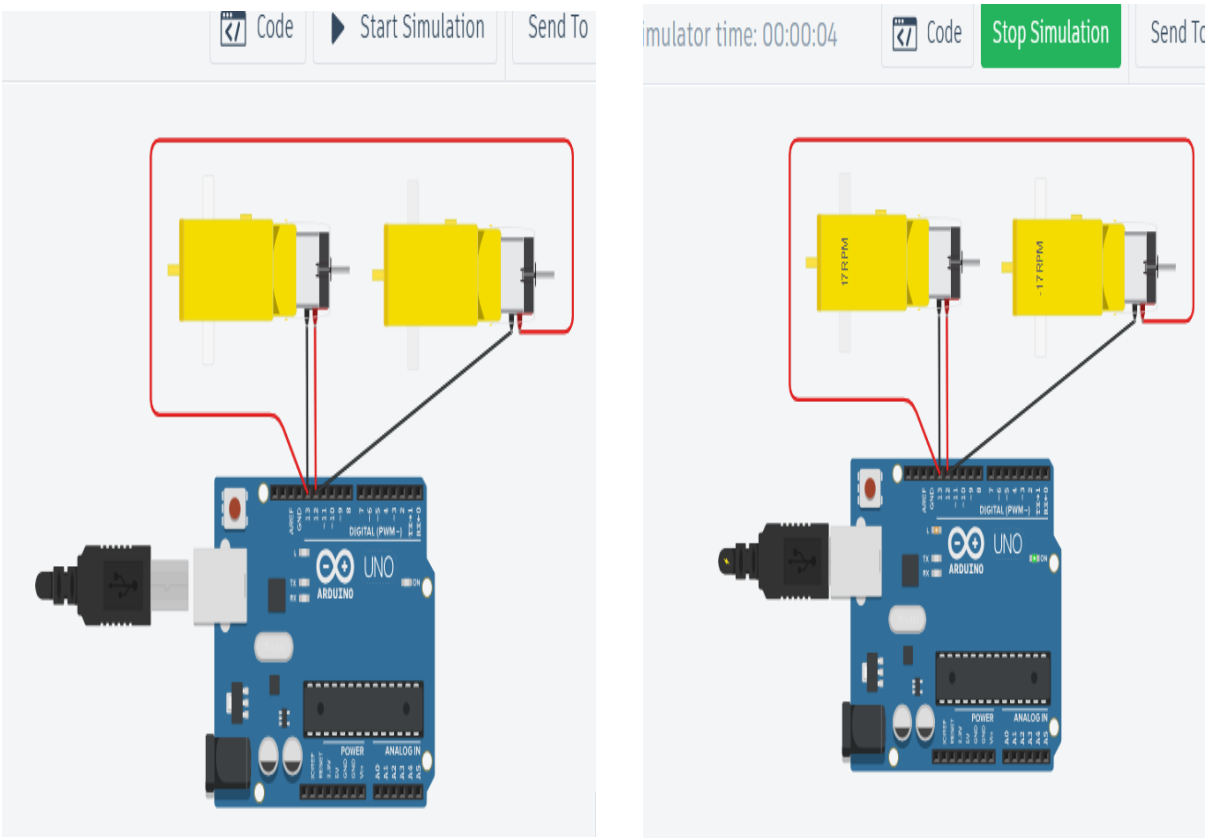


Fig. – Connections of Arduino and Hobbygearmotor using L293D IC

PRACTICAL 8

Design a circuit to vary the light intensity of LED using Photoresistor sensor with Arduino.

- **Steps for connections:**

1. **Set Up the Photoresistor Circuit (Voltage Divider):**

    - Place the photoresistor and the 10k-ohm resistor on the breadboard in series.
    - One end of the photoresistor will connect to 5V on the Arduino.
    - The junction between the photoresistor and the 10k-ohm resistor will connect to an analog input pin on the Arduino (e.g., A0).
    - The other end of the 10k-ohm resistor connects to GND on the Arduino.

This voltage divider circuit will output a variable voltage at the junction depending on the light level detected by the photoresistor, which the Arduino will read on the analog pin.

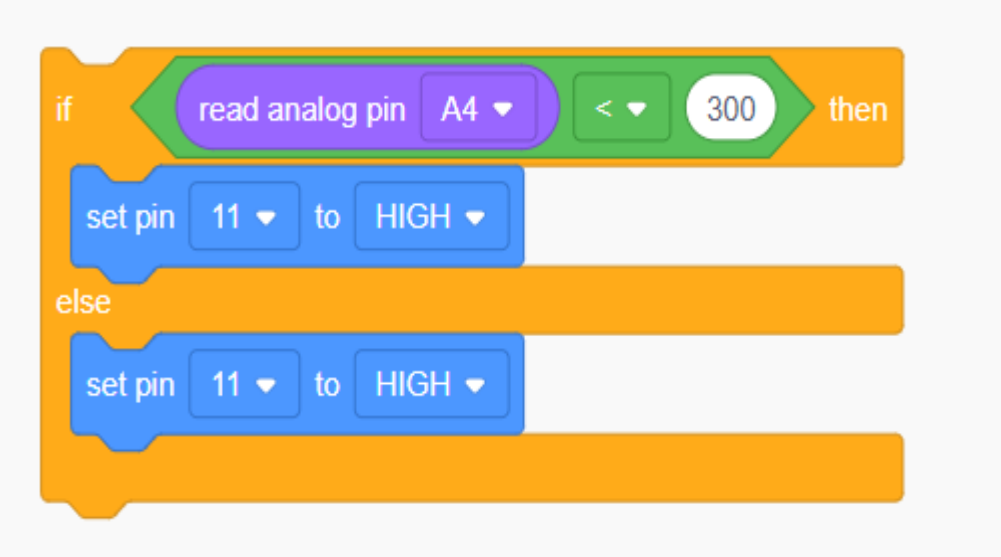
2. **Set Up the LED Circuit:**

    - Place the LED on the breadboard.
    - Connect a 220-ohm resistor in series with the anode (positive leg) of the LED to limit the current.
    - Connect the other end of the resistor to a digital pin on the Arduino (e.g., pin 9).
    - Connect the cathode (negative leg) of the LED to GND on the Arduino.

3. **Connect Power and Ground:**

    - Ensure the 5V and GND pins from the Arduino are connected to the respective power and ground rails on the breadboard, providing power to the circuit.

- **Block Code:**



- **Text Code:**

```
// C++ code

void setup()

{

  pinMode(A4, INPUT);

  pinMode(11, OUTPUT);

}

void loop()

{

  if (analogRead(A4) < 300) {

    digitalWrite(11, HIGH);
```

```
  } else {
    digitalWrite(11, LOW);
  }
  delay(10);
}
```

OUTPUT: -

Before Simulation **After Simulation**

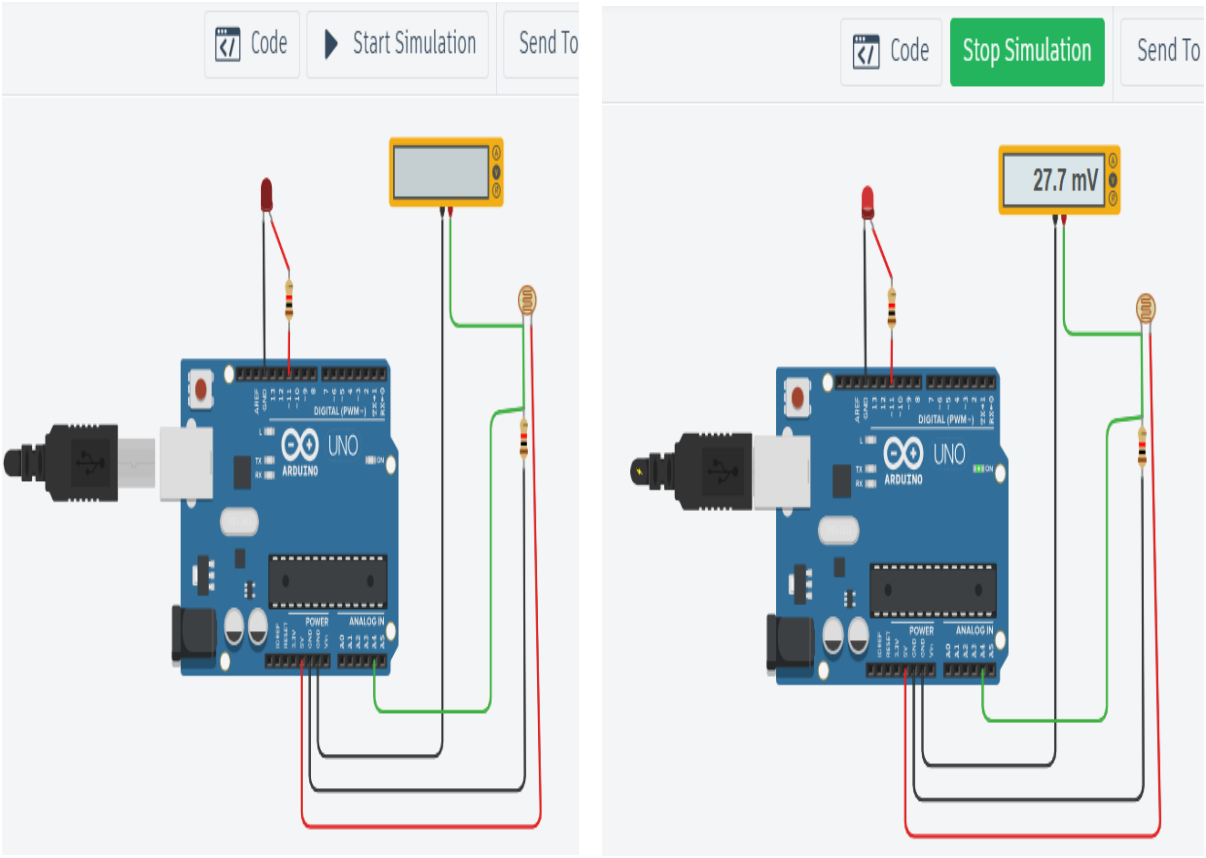


Fig. - Connections of photoresistor sensor, LED bulb using Arduino

PRACTICAL 9

Integrating temperature sensor through Arduino board and its application.

- **Connections Explained:**

1. Temperature Sensor (TMP)

- TMP Pin 1: Connect to the 5V pin on the Arduino to power the sensor.
- TMP Pin 2 (Output): Connect to A5 (Analog Pin 0) on the Arduino. This is where the analog temperature data is read.
- TMP Pin 3: Connect to GND on the Arduino for grounding.

The TMP36 sensor outputs a voltage that changes based on temperature, which the Arduino reads as an analog value on pin A5.

2. DC Motor

- Red Wire (Power): Connect to digital pin 10 on the Arduino. This is the pin that will control the motor's power.
- Green Wire (Ground): Connect to GND on the Arduino.

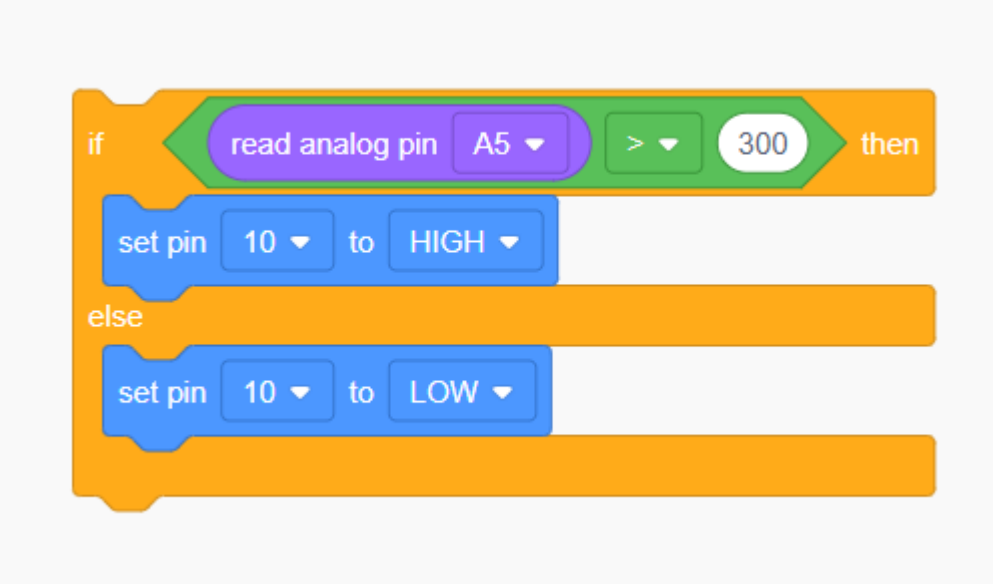
This setup uses the Arduino to power the motor directly from a digital pin, which is suitable for low-power applications. Note that a motor driver or transistor is usually recommended for higher power motors to protect the Arduino.

3. LCD Display

- The LCD display in this diagram appears to be connected to 5V and GND from the Arduino.
- Black Wire (GND): Connect to GND on the Arduino.
- Red Wire (Power): Connect to 5V on the Arduino.

The LCD display will show values based on the readings from the temperature sensor or other output values, though the specific data displayed would be controlled by code uploaded to the Arduino.

- **Block Code:**



- **Text Code:**

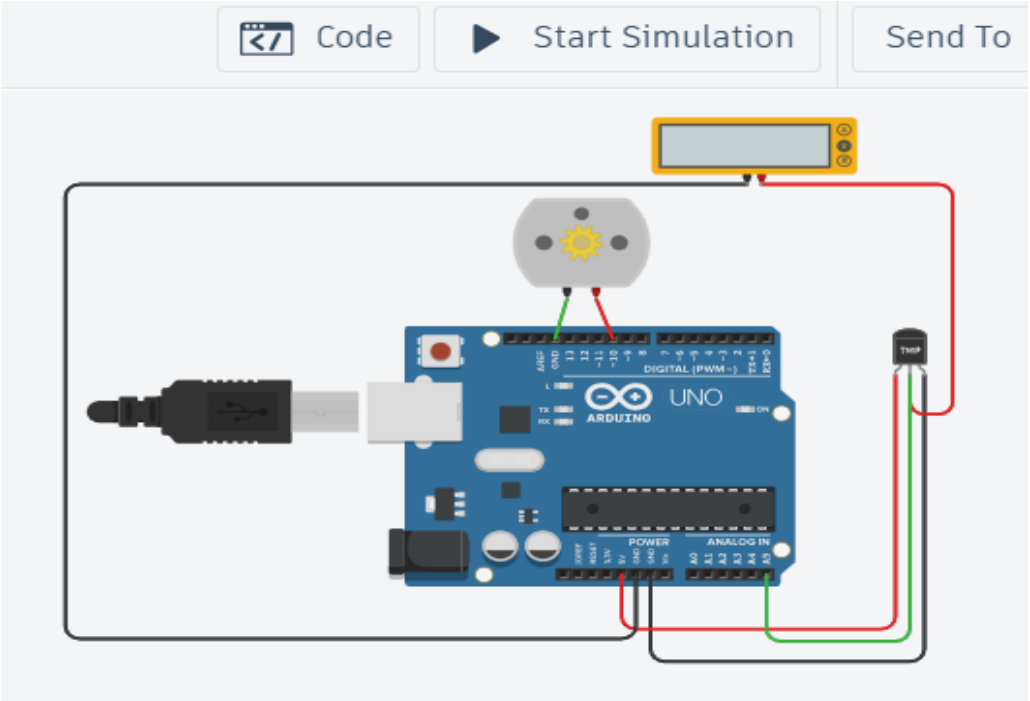
```
// C++ code

void setup()
{
  pinMode(A5, INPUT);
  pinMode(10, OUTPUT);
}


void loop()
{
  if (analogRead(A5) > 300) {
    digitalWrite(10, HIGH);
  } else {
    digitalWrite(10, LOW);
  }
  delay(10);
      }
```

OUTPUT: -

- **Before Simulation**



- **After Simulation**

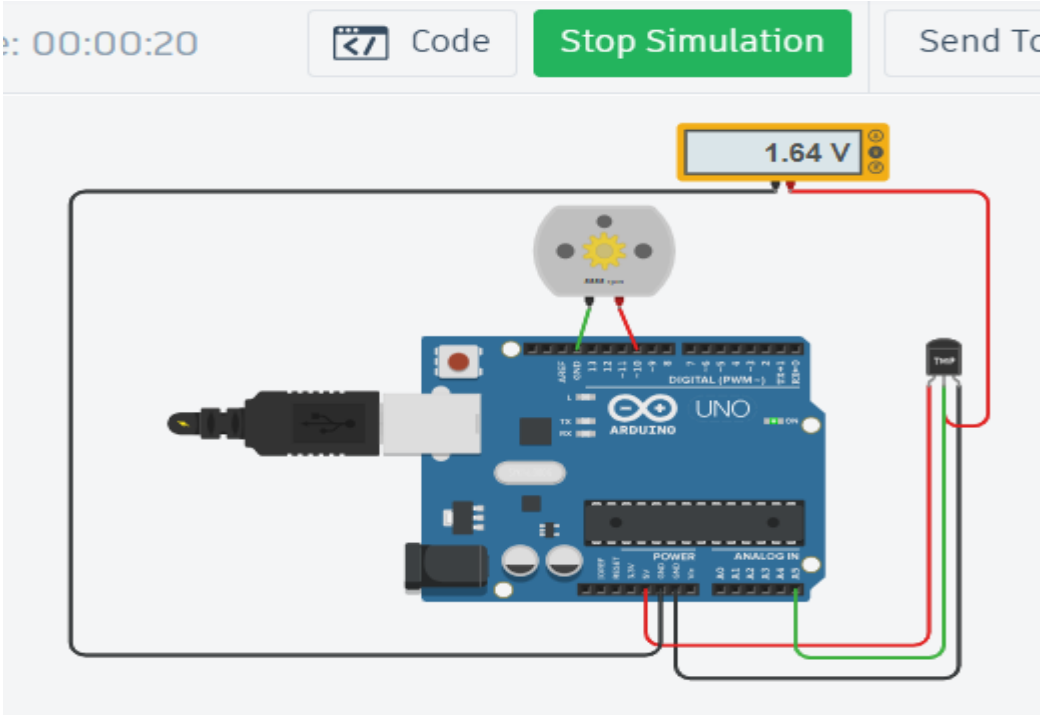


Fig. - Connections of Temperature Sensor, Dc Motor using Arduino

PRACTICAL 10

Integrating gas sensor in Arduino board designing a smoke detector.

- **Connection Explained:**

1. Connect the Gas Sensor (MQ-2/MQ-135)

- VCC (Power): Connect to the 5V pin on the Arduino to power the sensor.
- GND (Ground): Connect to the GND pin on the Arduino.
- AO (Analog Output): Connect to analog pin A0 on the Arduino. This pin outputs an analog voltage proportional to the concentration of gas in the air.

2. Connect the Buzzer or LED for Alert

- **If using a Buzzer:**
  - Connect the positive terminal of the buzzer to digital pin 8 on the Arduino.
  - Connect the negative terminal of the buzzer to GND on the Arduino.
- **If using an LED:**
  - Connect the anode (long leg) of the LED to digital pin 8 through a 220-ohm resistor.
  - Connect the cathode (short leg) of the LED to GND on the Arduino.

3. Connect Power and Ground

- Ensure the 5V and GND pins from the Arduino are connected to the respective power and ground rails on the breadboard to power the sensor.

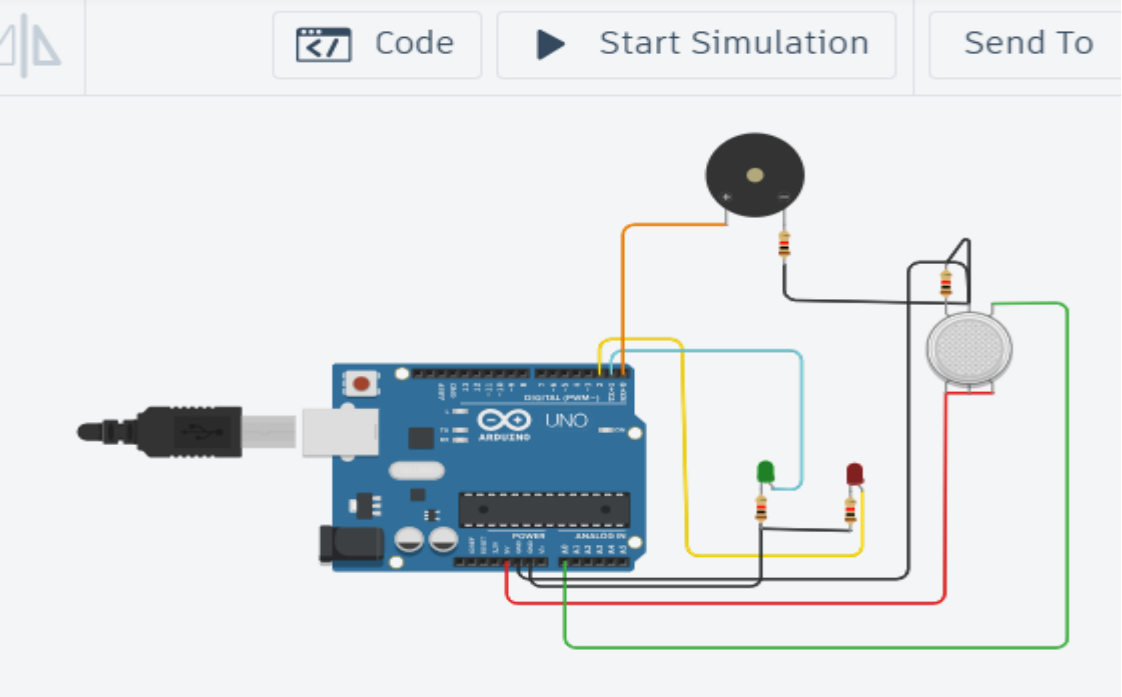
- **Text Code:**

```
#define gasSensor 1
#define buzzer 0
#define ledGreen 1
#define ledRed 2
#define HIGH 600
void setup() {
  //Initialising all pins
  pinMode(gasSensor, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledRed, OUTPUT);
 }
void loop() {
 //Read data from the sensor
int gas_value = analogRead(gasSensor);
//check data from sensor if there is smoke, if will execute otherwise else will execute
if(gas_value > HIGH)
{
 tone(buzzer,1000,500);
 digitalWrite(ledRed, HIGH);
 digitalWrite(ledGreen,LOW);

 }
else{
 noTone(buzzer);
 digitalWrite(ledGreen,HIGH);
 digitalWrite(ledRed, LOW);
}
delay(200);
}
```

OUTPUT: -

- **Before Simulation**



- **After Simulation**

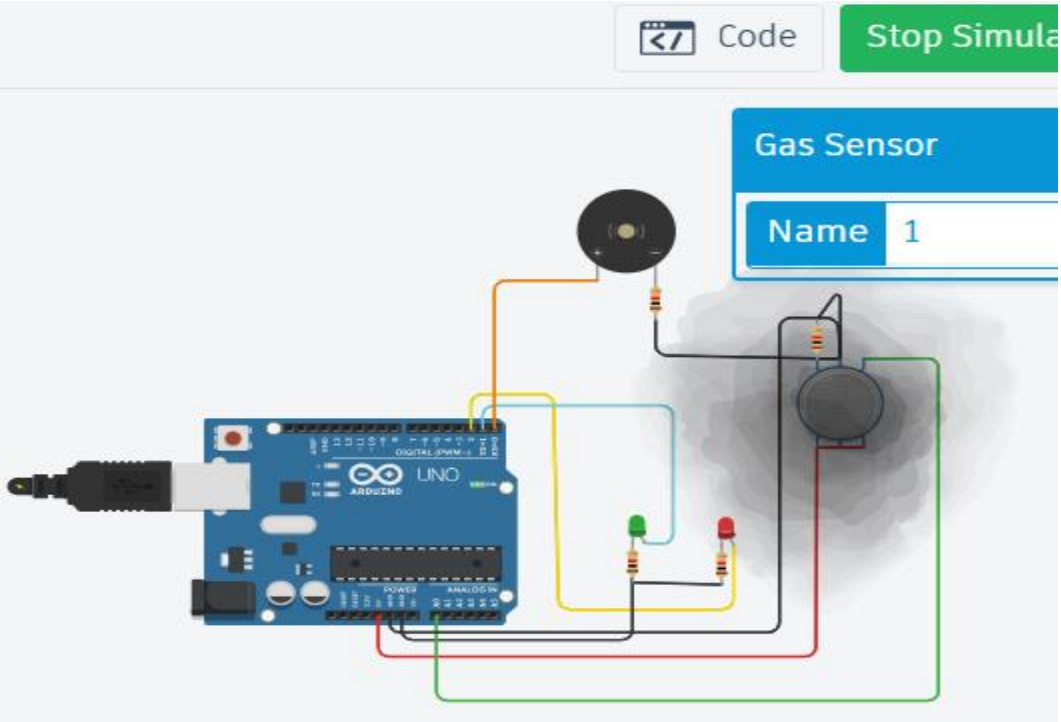


Fig. – Connection of smoke detection

PRACTICAL 11

Integrating PIR sensor with Arduino using tinkercad simulator and print its value on serial monitor.

- **Connection Explanation:**

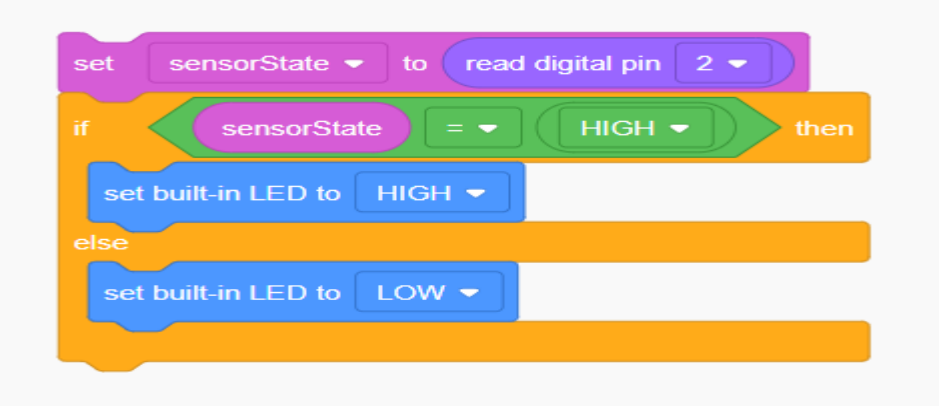
1. **Place Components on the Workbench:**

   - Drag an Arduino Uno onto the workbench.
   - Add a PIR sensor (it's usually labeled as "PIR Motion Sensor" in Tinkercad).

2. **Wiring the PIR Sensor:**

   - The PIR sensor has three pins: VCC, GND, and OUT.
   - VCC connects to the 5V pin on the Arduino.
   - GND connects to a GND pin on the Arduino.
   - OUT connects to digital pin 2 on the Arduino.

- **Block code:**



- **Text code:**

```
int sensorState = 0;
void setup()
{
```

```
  pinMode(2, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600); // Initialize serial communication at 9600 baud
}
void loop()
{
  sensorState = digitalRead(2); // Read the sensor state
  if (sensorState == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Motion detected!");
  }
else {
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("No motion.");
  delay(10);
}
```

OUTPUT: -

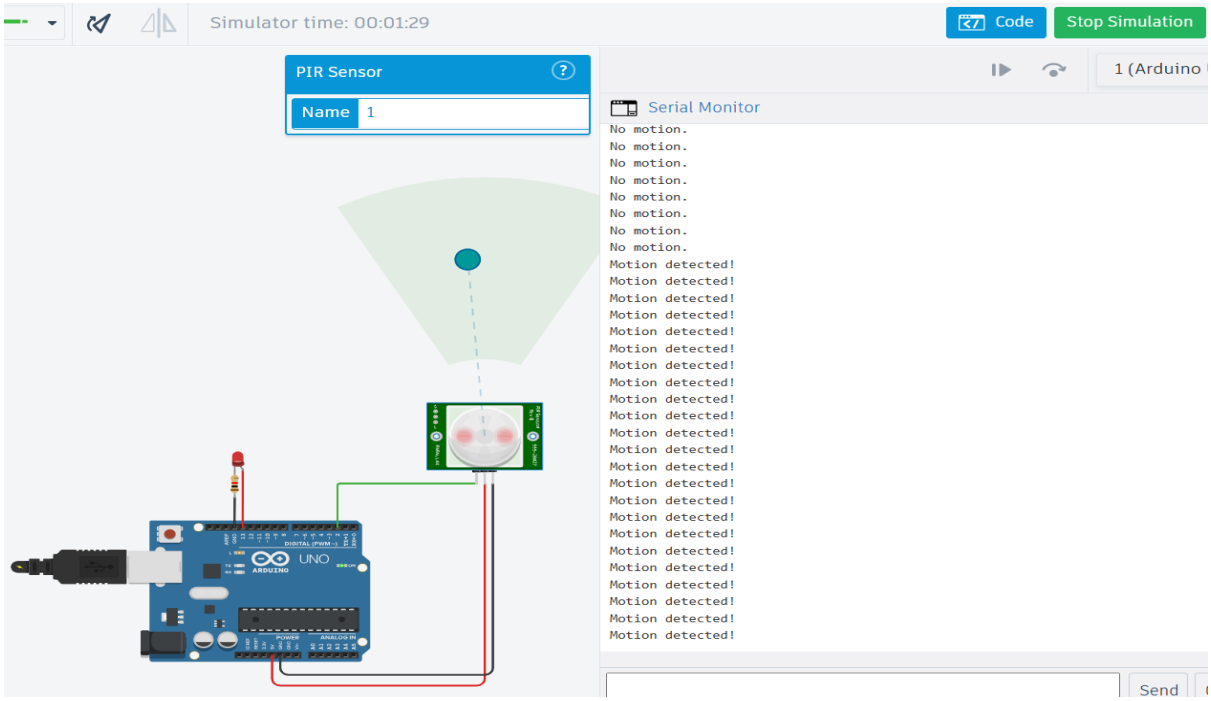


Fig. – Connection of PIR sensor using Arduino

PRACTICAL 12

Measuring the distance using ultrasonic sensors and make LEDs blink using arduino. (Car Parking system)

- **Steps for Connections**

1. LED Connections

- Connect the anode (long leg) of the first LED to digital pin 10 on the Arduino through a resistor.
- Connect the anode of the second LED to digital pin 11 on the Arduino through a resistor.
- Connect the anode of the third LED to digital pin 12 on the Arduino through a resistor.
- Connect the cathode (short leg) of each LED to GND on the Arduino.

2. Buzzer Connection

- Connect the positive terminal of the buzzer to digital pin 8 on the Arduino.
- Connect the negative terminal of the buzzer to GND on the Arduino.

3. Ultrasonic Sensor (HC-SR04) Connections

- VCC Pin: Connect the VCC pin of the ultrasonic sensor to the 5V pin on the Arduino.
- GND Pin: Connect the GND pin of the ultrasonic sensor to GND on the Arduino.
- Trig Pin: Connect the Trig pin of the sensor to digital pin 7 on the Arduino.
- Echo Pin: Connect the Echo pin of the sensor to digital pin 6 on the Arduino

- **Text code:**

```
const int trigPin = 7;
const int echoPin = 6;
const int led1 = 10;
const int led2 = 11;
```

```
const int led3 = 12;
const int buzzer = 8;

long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Set the trigPin high for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distance = duration * 0.034 / 2;

  // Print the distance on the Serial Monitor
  Serial.print("Distance: ");
```

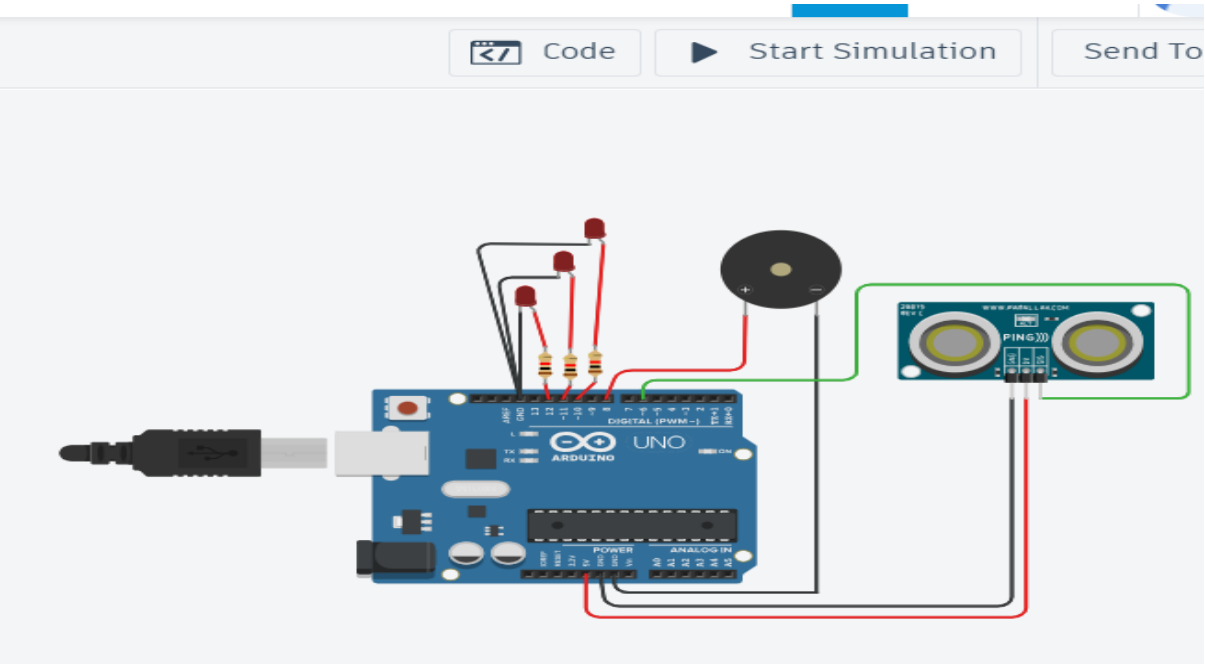
```
 Serial.print(distance);
 Serial.println(" cm");

 // Turn LEDs and buzzer on or off based on distance
if (distance <= 10) {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  tone(buzzer, 1000); // Buzzer at 1000 Hz
 }
 else if (distance <= 20) {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, LOW);
  tone(buzzer, 500); // Buzzer at 500 Hz
 }
 else if (distance <= 30) {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  noTone(buzzer); // Buzzer off
 }
 else {
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  noTone(buzzer); // Buzzer off
 }

 delay(100);
}
```

OUTPUT: -

- **Before Simulation**



- **After Simulation**

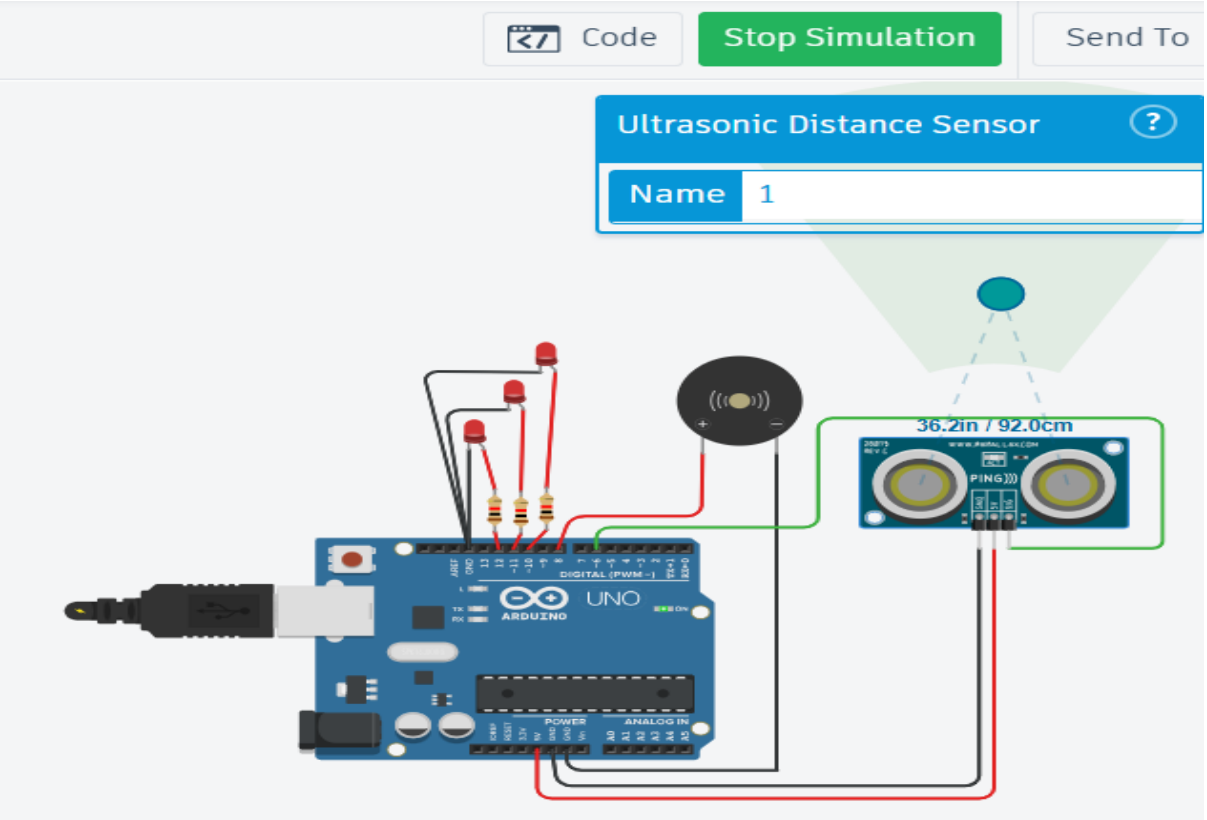


Fig. – Connection of car parking system

PRACTICAL 13

Home automation system (Sensors: Ultrasonic, PIR, Temperature sensor Actuators: Servomotor, DC-motor, LED's)

- **Connection Explanation:**

1. LEDs with Resistors

- Anode (long leg) of the first LED connects to digital pin 10 on the Arduino, with a resistor in series.
- Anode of the second LED connects to digital pin 11 on the Arduino, with a resistor in series.
- Cathodes of both LEDs connect to GND on the Arduino.

2. Buzzer

- Positive terminal of the buzzer connects to digital pin 9 on the Arduino.
- Negative terminal of the buzzer connects to GND.

3. Ultrasonic Sensor (HC-SR04)

- VCC Pin connects to the 5V pin on the Arduino.
- GND Pin connects to GND on the Arduino.
- Trig Pin connects to digital pin 7 on the Arduino.
- Echo Pin connects to digital pin 6 on the Arduino.

4. PIR Sensor

- VCC Pin connects to the 5V pin on the Arduino.
- GND Pin connects to GND.
- Output Pin connects to digital pin 8 on the Arduino.

5. Temperature Sensor (TMP36)

- VCC Pin (left pin) connects to 5V on the Arduino.
- GND Pin (right pin) connects to GND.
- Analog Output Pin (middle pin) connects to analog pin A0 on the Arduino.

6. Photoresistor (LDR)

- Connect one side of the photoresistor to 5V.
- Connect the other side of the photoresistor to analog pin A1 on the Arduino.
- Place a resistor (typically 10kΩ) between the other side of the LDR and GND to create a voltage divider.

7. LCD Display (Assuming it's I2C)

- SDA Pin connects to A4 on the Arduino.
- SCL Pin connects to A5 on the Arduino.
- VCC Pin connects to 5V.
- GND Pin connects to GND.

- **Text Code:**

```
// Smart Home
int ldrvalue=0;
void setup()
{
  pinMode(4,INPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,INPUT);
  pinMode(A0,INPUT);
  pinMode(A5, INPUT);
  pinMode(10, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  //burglar alarm
  int p=digitalRead(4);
  if(p)
  {
    tone(12,30);
    delay(100);

  }
  noTone(12);
  delay(100);

  //ultrasonic sensor
  digitalWrite(2,LOW);
  digitalWrite(2,HIGH);
```

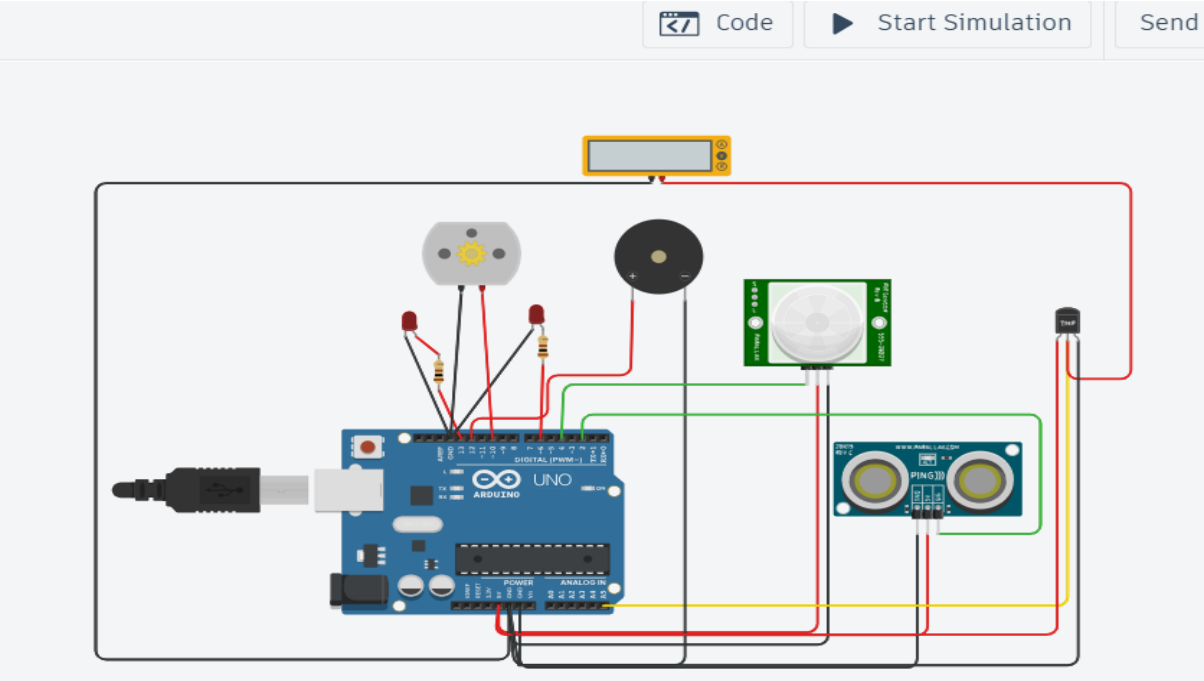
```
  delayMicroseconds(10);
  digitalWrite(2,LOW);
  float dur=pulseIn(3,HIGH);
  float dis=(dur*0.034)/2;
  Serial.print("Ultrasonic dis=");
  Serial.println(dis);
  if(dis<50)
  {
    digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,LOW);
    delay(100);

  }
  //LDR
  ldrvalue=analogRead(A0);
  Serial.print("LDR dis=");
  Serial.println(ldrvalue);
  analogWrite(6,map(ldrvalue,0,1023,0,255));
  delay(100);

  //motor
  if (analogRead(A5) > 300) {
    digitalWrite(10, HIGH);
  } else {
    digitalWrite(10, LOW);
  }
  delay(10);
}
```

OUTPUT: -

- **Before Simulation**



- **After Simulation**

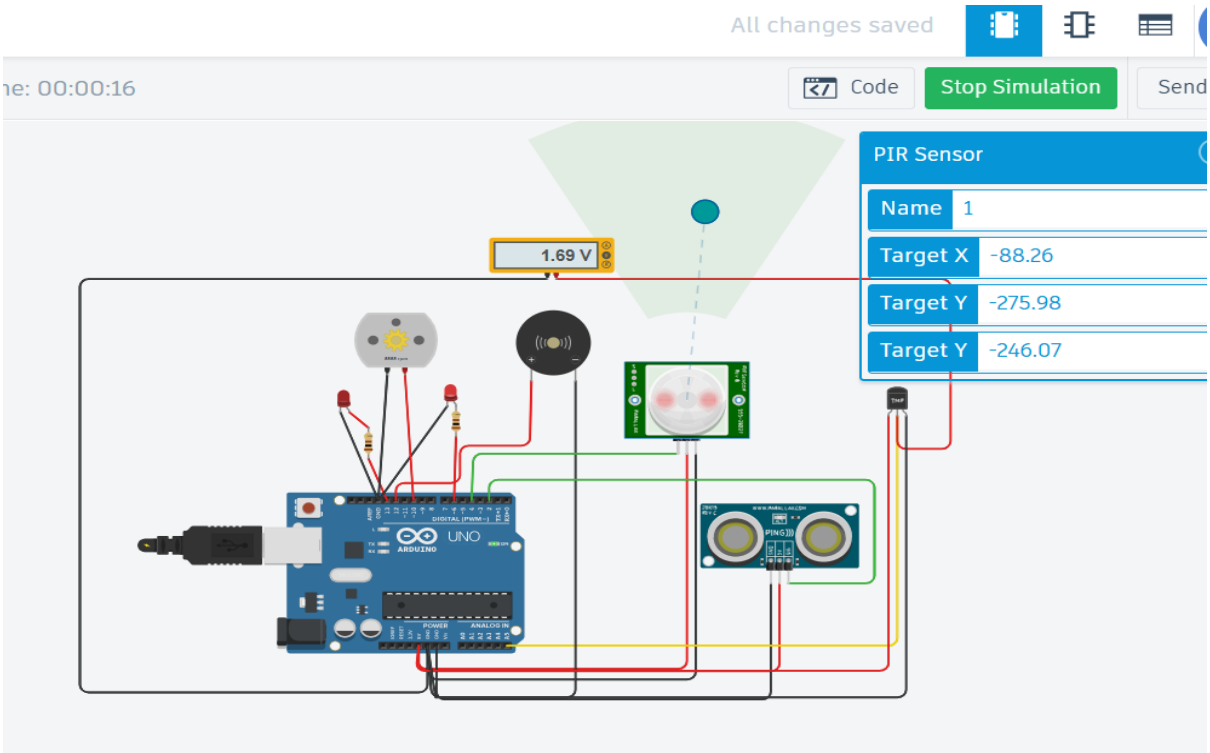


Fig. – Smart Home Connection