160120749022

Experiment 1

Aim: Introduction to IoT and the Arduino Programming Language.

Description: Arduino is a programming language made by a company named the same. It is compatible with many hardware providers such as NodeMCU. Arduino's software is all free software, the hardware is also open hardware.

The language is very similar to C++, thus everything such as Classes, Objects, control statements, loops, datatypes are all same as C++.

A microcontroller is a computer that is designed for doing limited tasks such as reading data from sensors, controlling Actuators and reporting back to another device through a network.

Sensors: A sensor is a device that converts any physical value into electric signals.

Ex: Temperature Sensor, Infrared Sensor.

Actuators: An actuator is a device that performs actions based on the electric signals it obtains from a controller.

Ex: Motor, LED.

Interfacing peripherals to a microcontroller:

A microcontroller has two types of pins:

Analog: These pins are mostly used to capture data from sensors. They have the ability to read continuous voltage levels.

Digital: These pins are mostly used to output data to a sensor. They have the ability to send fixed values only (HIGH or LOW). Some digital pins can output a PWM signal as well.

160120749022

Here are some Arduino specific functions for common operations:

`pinMode( pinNum, INPUT|OUTPUT):`

Used to specify whether a pin is used for input or output.

`digitalWrite( pinNum, HIGH|LOW):`

Used to output discrete values to an output device.

`analogWrite( pinNum, Value):`

Used to output continuous values to an output device.

`digitalRead(pinNum):`

Used to gather discrete input values from an input device.

`analogRead(pinNum):`

Used to gather analog input values from an input device.

`void setup():`

This block contains code that performs setup.

`void loop():`

The code inside this block is executed in a loop. It contains the main logic of the application.

`delay():`

Used to insert delays between operations inside the program.

160120749022

Experiment 2

Aim: Write a program to interface a PIR sensor with Arduino and turn on LED when motion is detected.

Description:

PIR stands for Passive Infrared. This sensor can detect Infrared radiation. Thus, it can be used to detect the presence of hot objects and living things.

The sensor features three pins, two of which are for power and the third one being a data pin.

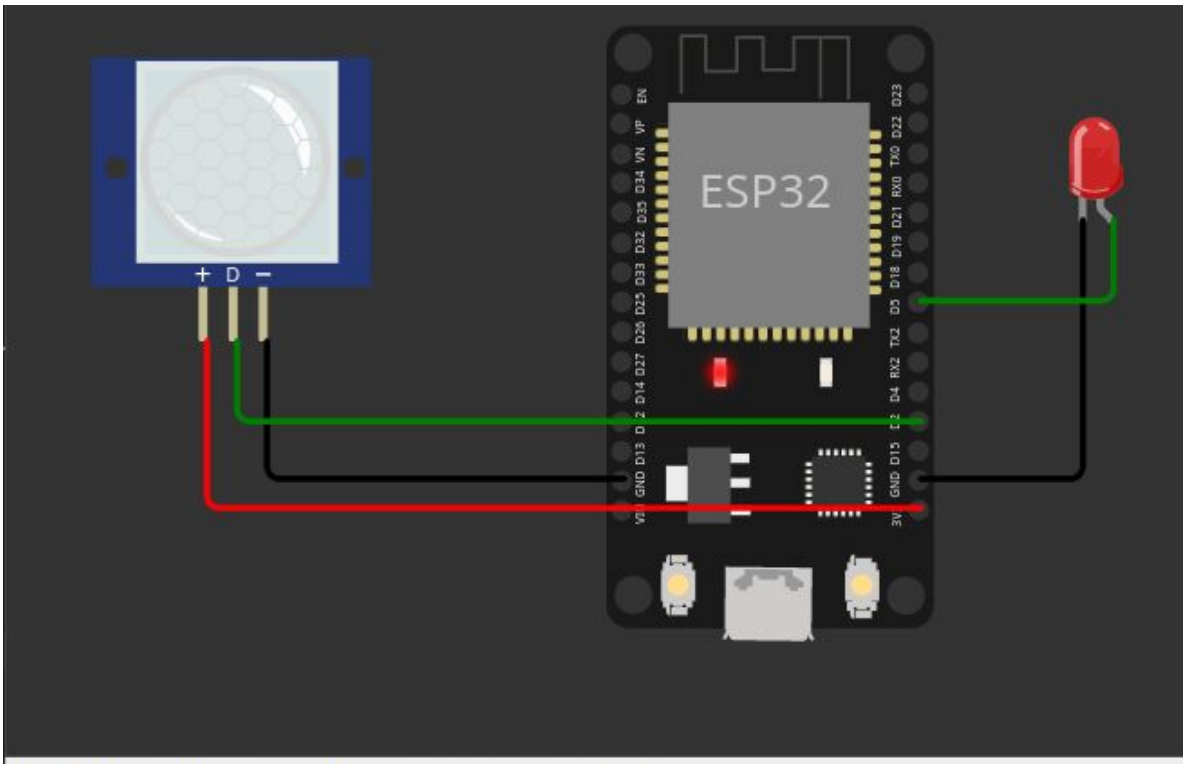
Program:

```
int Redled = 5;
int sensor = 2;

void setup() {
pinMode(sensor, INPUT);
pinMode(Redled, OUTPUT);
Serial.begin(115200);
}

void loop() {
long state = digitalRead(sensor);
if(state==HIGH){
digitalWrite(Redled,HIGH);
```

160120749022

```
14  Serial.println("Motion detected!..LED will blink");
15  delay(1000);
16  }
17  else{
18  digitalWrite(Redled,LOW);
19  Serial.println("No motion detected..LED will not glow");
20  delay(1000);
21  }
22  }
```

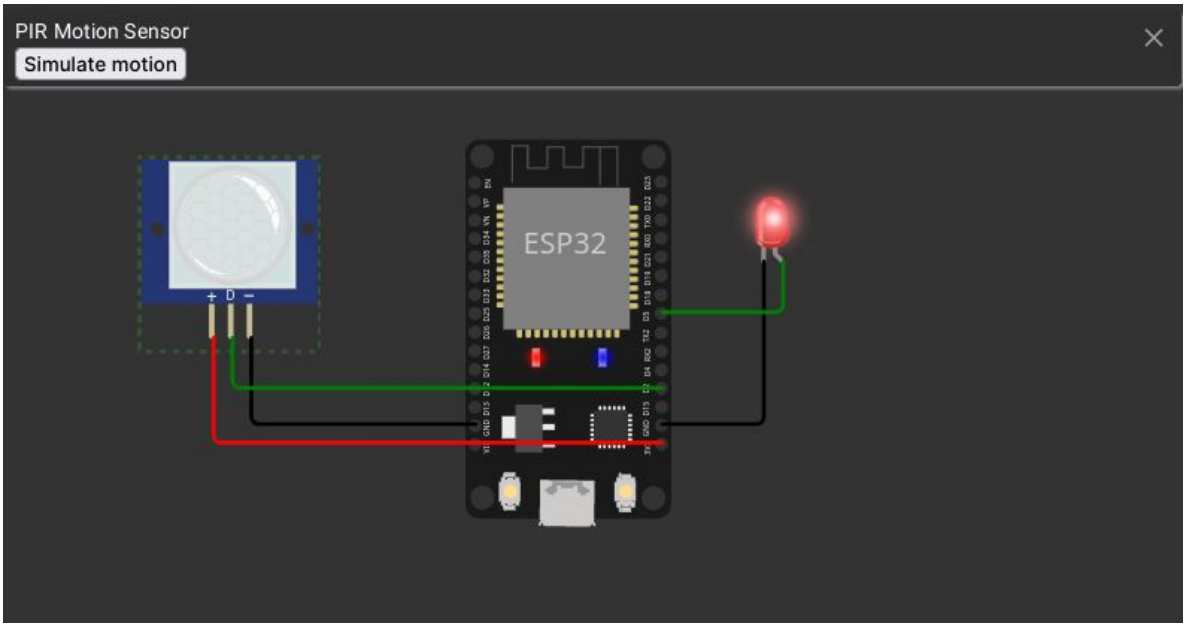
Outputs:




```
No motion detected..LED will not glow
No motion detected..LED will not glow
No motion detected..LED will not glow
```

160120749022

Upon detection of motion:



Conclusions:

The sensor outputs only two possible values, HIGH or LOW.

It is best suited for performing motion detection.

160120749022

EXPERIMENT-3

Aim: Write a program to interface DHT22 sensor with Arduino and display temperature and humidity readings.

Description:

The DHT22 sensor consists of a humidity sensing component along with an NTC thermistor.

The humidity sensing component has two electrodes connected by a moisture holding substrate. Thus with a change in humidity, the conductivity of the path changes.

The NTC thermistor has a resistance that varies with temperature. Thus the resistance readings can be used to detect temperature.

The DHT22 sensor has four pins, two of which are for power, one being a no-connect and the final one being a data pin.

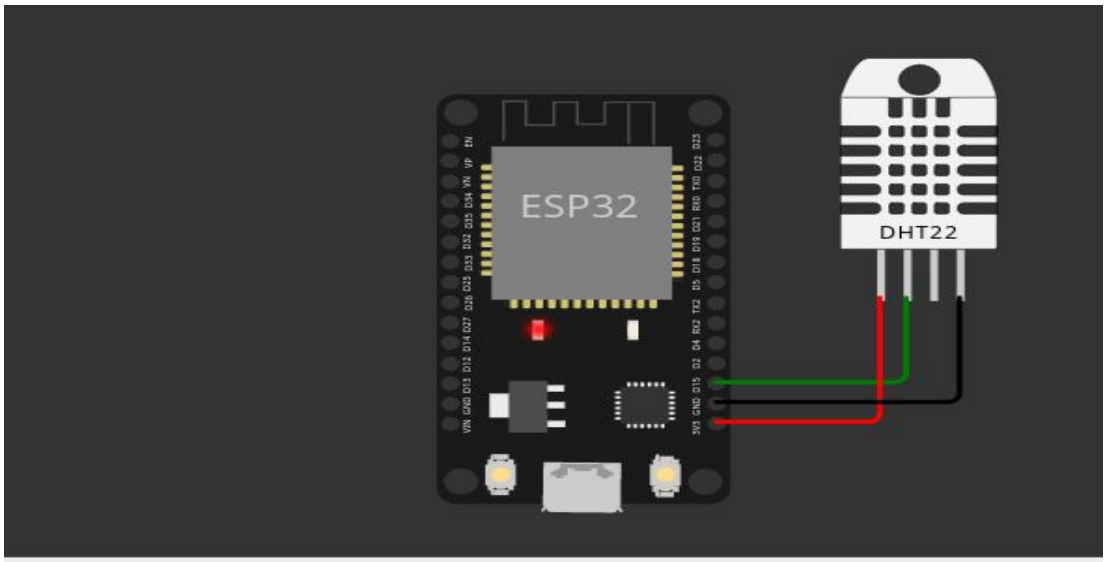
Program:

```
#include "DHTesp.h"
const byte sensor=15;
DHTesp dht;

void setup(){
Serial.begin(9600);
pinMode(sensor, INPUT);
dht.setup(sensor, DHTesp::DHT22);
```

160120749022

```
 9  }
10
11  void loop(){
12  TempAndHumidity data = dht.getTempAndHumidity();
13  Serial.println("Temp: " + String(data.temperature, 2) +
"'C");
14  Serial.println("Humidity: " + String(data.humidity, 1) +
"%");
15  Serial.println();
16  delay(1000);
17  }
```

Output:




```
---
Temp: 24.00°C
Humidity: 40.0%
---
Temp: 24.00°C
Humidity: 40.0%
---
```

Conclusion:

The sensor outputs a voltage corresponding to the temperature and humidity. The DHTesp library takes care of interpreting the voltage values and providing direct readings in Celsius and Percentage.

160120749022

Experiment 4

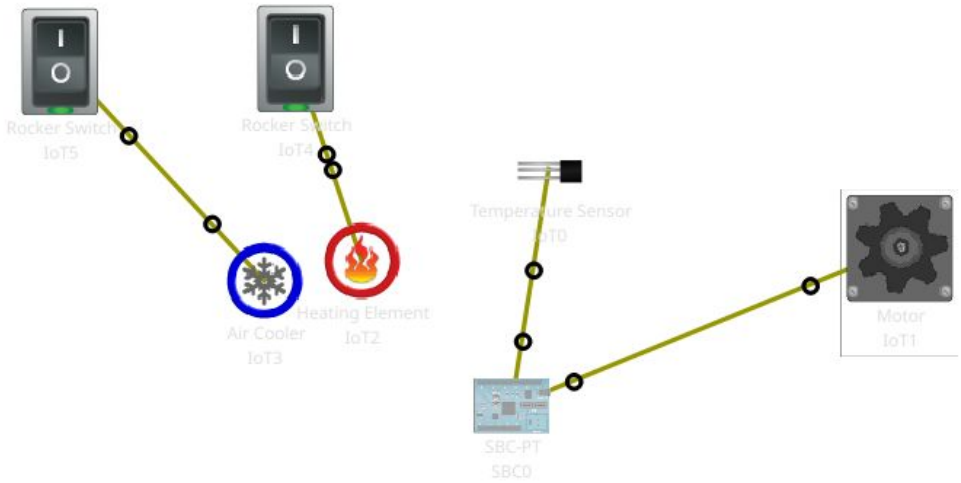
Aim: To interface a motor with a Raspberry Pi. Turn ON the motor when temperature is high.

Description:

This experiment has been performed on the proprietary network simulation software Packet Tracer.
An SBC is connected to a motor and an NTC Thermistor.
An NTC Thermistor changes its resistance based on the temperature of the room, thus acting as a temperature sensor.

The SBC takes the readings from the Thermistor and turns the motor when temperature reaches a certain threshold.

The SBC has been programmed in the Python Programming Language.



The heaters and coolers have been used to adjust the temperature of the simulated environment.

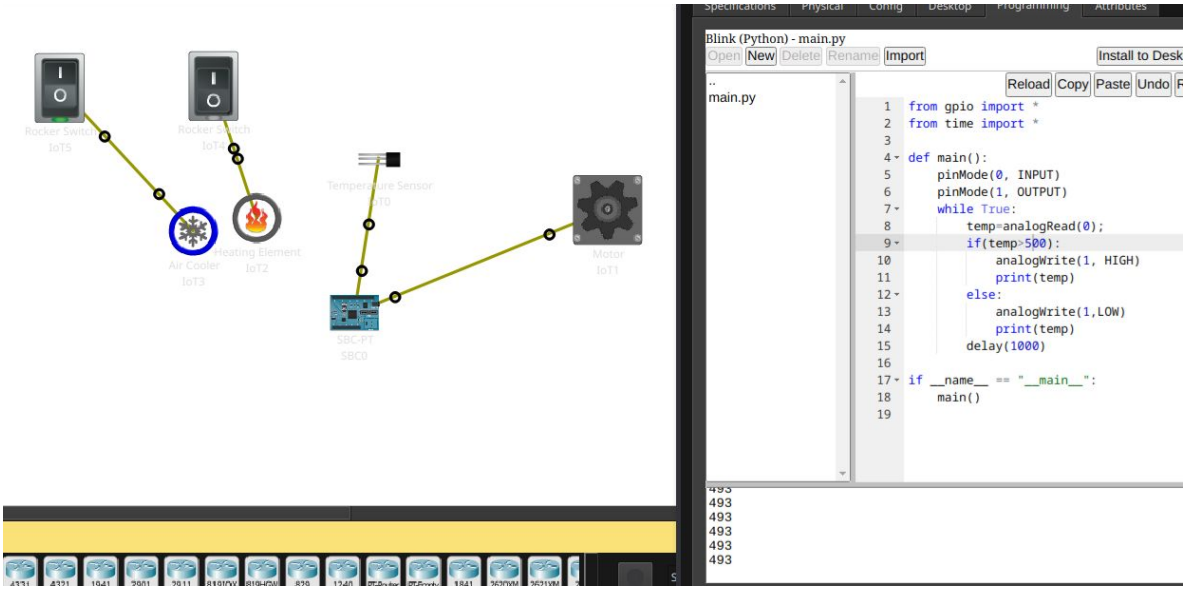
160120749022

Program:

```
1  from gpio import *
2  from time import *
3
4  def main():
5      pinMode(0, INPUT)
6      pinMode(1, OUTPUT)
7      while True:
8          temp=analogRead(0);
9          if(temp>500):
10             analogWrite(1, HIGH)
11             print(temp)
12         else:
13             analogWrite(1,LOW)
14             print(temp)
15         delay(1000)
16
17 if __name__ == "__main__":
18     main()
```

160120749022

Output:



Once the heating element is turned on and the temperature crosses 500 motor starts rotating.

Conclusions:

The motor draws a high amount of current, thus in real scenarios it must be powered by an external source, as Single Board Computers cannot supply high currents through their Pins.

This can be achieved by adding a Transistor or a MOSFET in a switch configuration.

160120749022

Experiment 5

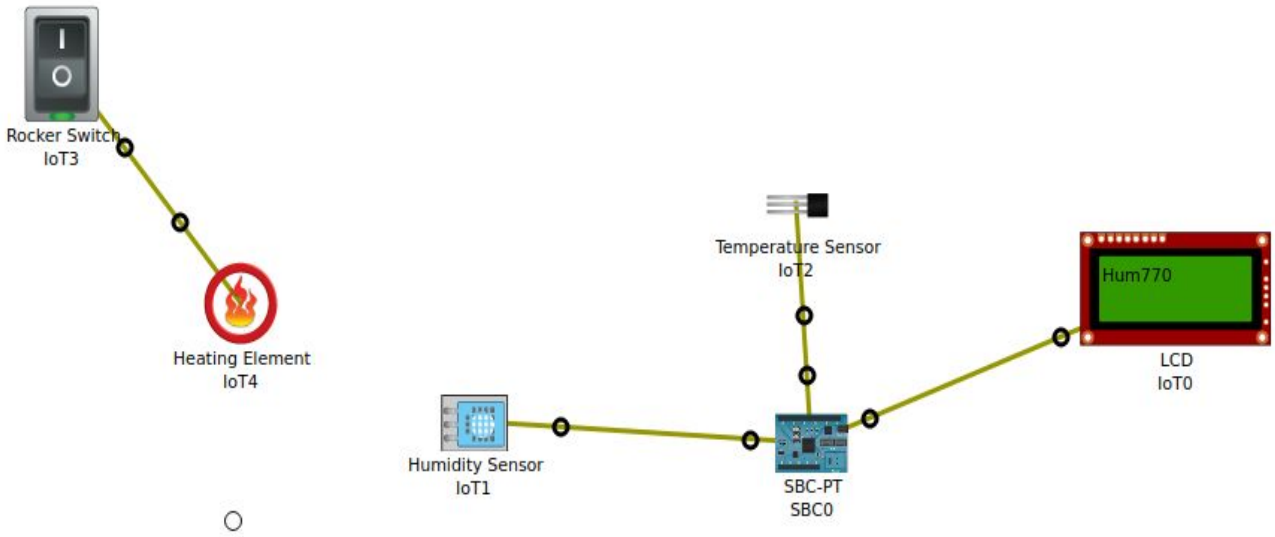
Aim: To interface an LCD with a Raspberry Pi inorder to display the Temperature and Humidity readings from a sensor.

Description:

An LCD stands for Liquid Crystal Display. They are cheap and common in electronic displays.

This experiment is performed using the proprietary, non-free network simulator that is Cisco Packet Tracer.

An SBC is connected to a Temperature sensor and a Humidity sensor. The readings are interpreted and displayed onto the connected LCD.



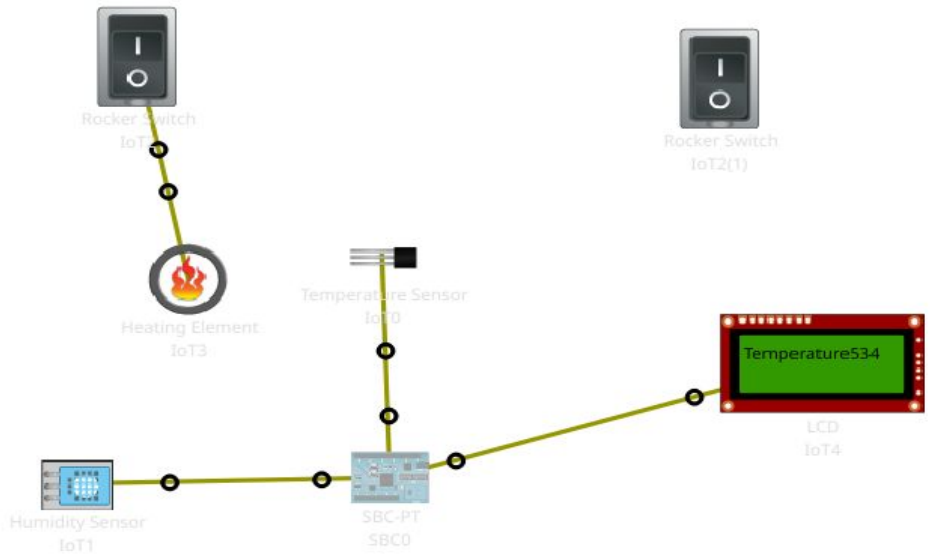
A heating element is added to control the temperature for demonstration purposes.

160120749022

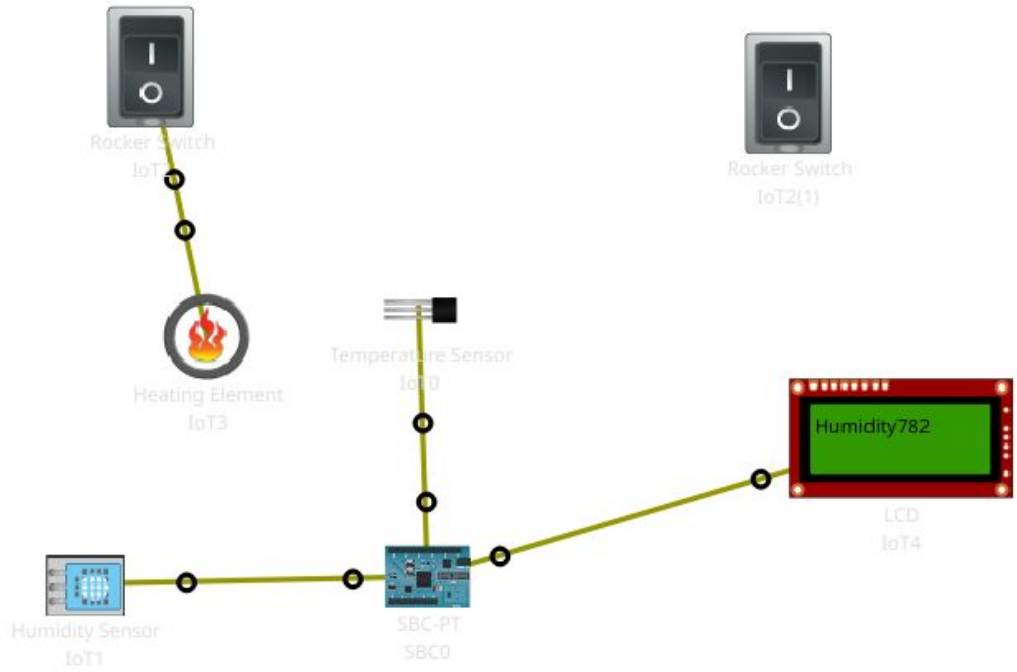
Program:

```
tempSensor = 0;
humSensor = 1;
LCDPin = 2;
from gpio import *
from time import *
def main():
pinMode(tempSensor,INPUT)
pinMode(humSensor, INPUT)
pinMode(LCD, OUT)
while True:
 hum=analogRead(humSensor);
      temp=analogRead(tempSensor);
delay(1000)
customWrite(LCDPin,"Humidity "+str(hum)+" F")
      delay(2000)
      customWrite(LCDPin,"Temperature "+str(temp)+" C")
      delay(1000)
 if __name__ == "__main__":
main()
```

Output:



160120749022



Conclusions:

The LCD displays the readings provided by the SBC.

160120749022