# FINAL REPORT OPEN ENDED LAB

## Contents

# SYSTEM ARCHITECTURE:

## OBJECTIVE:

**The objective of this project is to design and implement an IoT-based temperature control system that can maintain a desired temperature range in a given environment.**

## SCOPE:

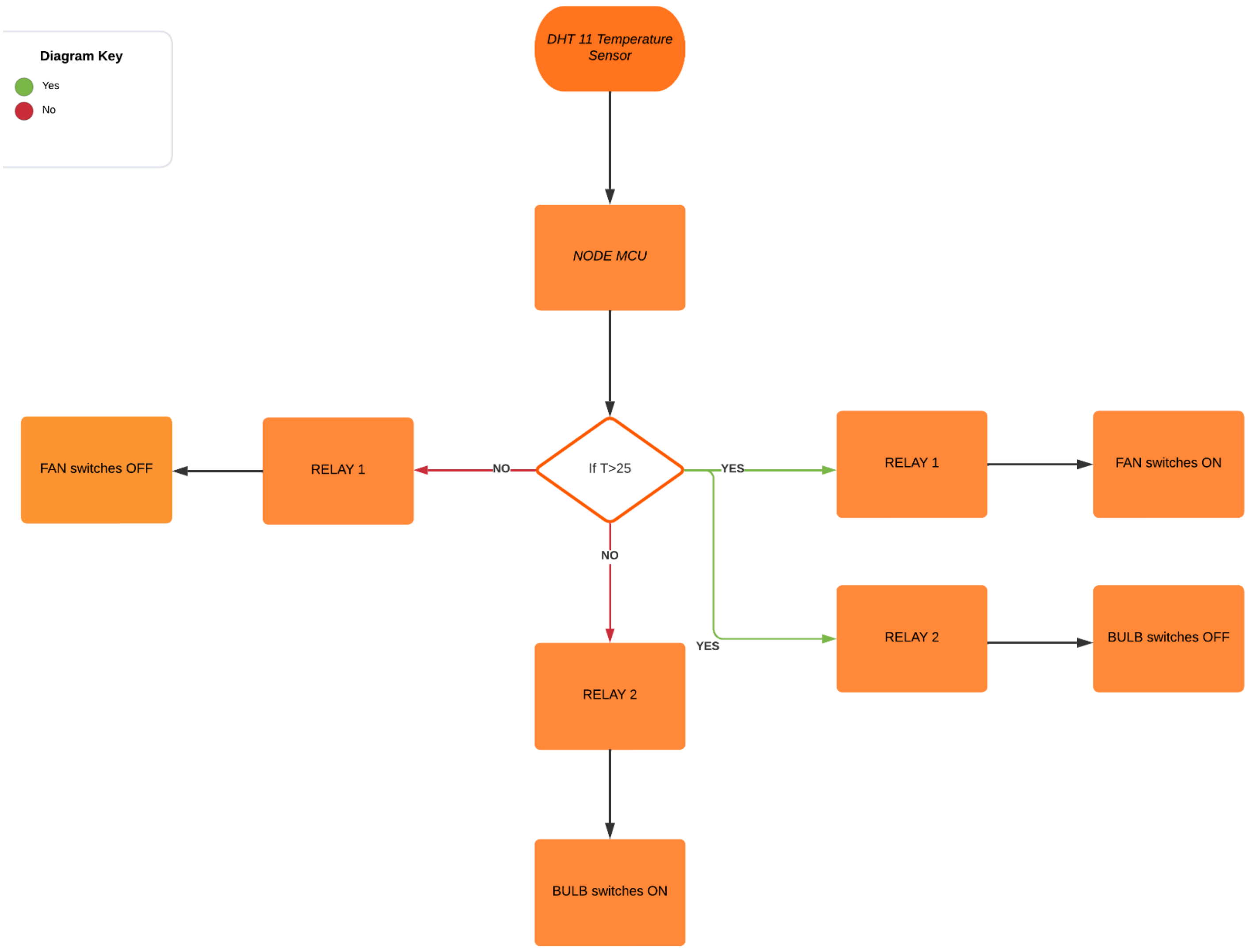
**Temperature regulation is critical in a variety of modern industries. Storage facilities, greenhouses, chemical factories, are some examples of places that demand a temperature monitoring system that is precise and dynamic. Automated IoT temperature monitoring systems can do this job efficiently. The system can also be connected to arduino iot cloud to monitor and control the temperature remotely through a web or mobile application. This will enable the user to check the temperature of the space from anywhere and make adjustments as needed. The system is simple and low cost and can be easily scaled to control multiple devices and rooms, making it a suitable solution for both residential and commercial buildings.**

## SECTOR:

**Since the system is designed for temperature control, which is a common requirement in many sectors, hence it can be use in multiple sectors like;**

1. **Residential Sectors: For homes, etc**
2. **Commercial Sectors: It can be used for servers' rooms or it can be used for the comfort of employees**
3. **Industrial Sectors: Certain products during manufacturing requires a define range of temperature and humidity like in pharmaceutical industry.**

## DIRECTION OF DATA FLOW:



**The data flow direction of the system is as follows:**

- **Temperature sensor measures the ambient temperature and humidity and sends the data to the Node MCU.**
- **Node MCU receives the temperature data and compares it to the set threshold of 25 degree Celsius.**
- **If the temperature is above 25 degrees Celsius, the Node MCU sends a command to the relay to turn ofn the fan and turn off the heating element.**
- **If the temperature is below 25 degrees Celsius, the Node MCU sends a command to the relay to turn off the fan and turn on the heating element.**

## CONNECTION OF WIRES:

**ThingProperties.h is used for updating information on dashboard**

**Pin D1 of nodemcu is connected to sensor DHT11. The ground and voltage pin of dht11 is connected with nodemcu ground and 3v pin.**

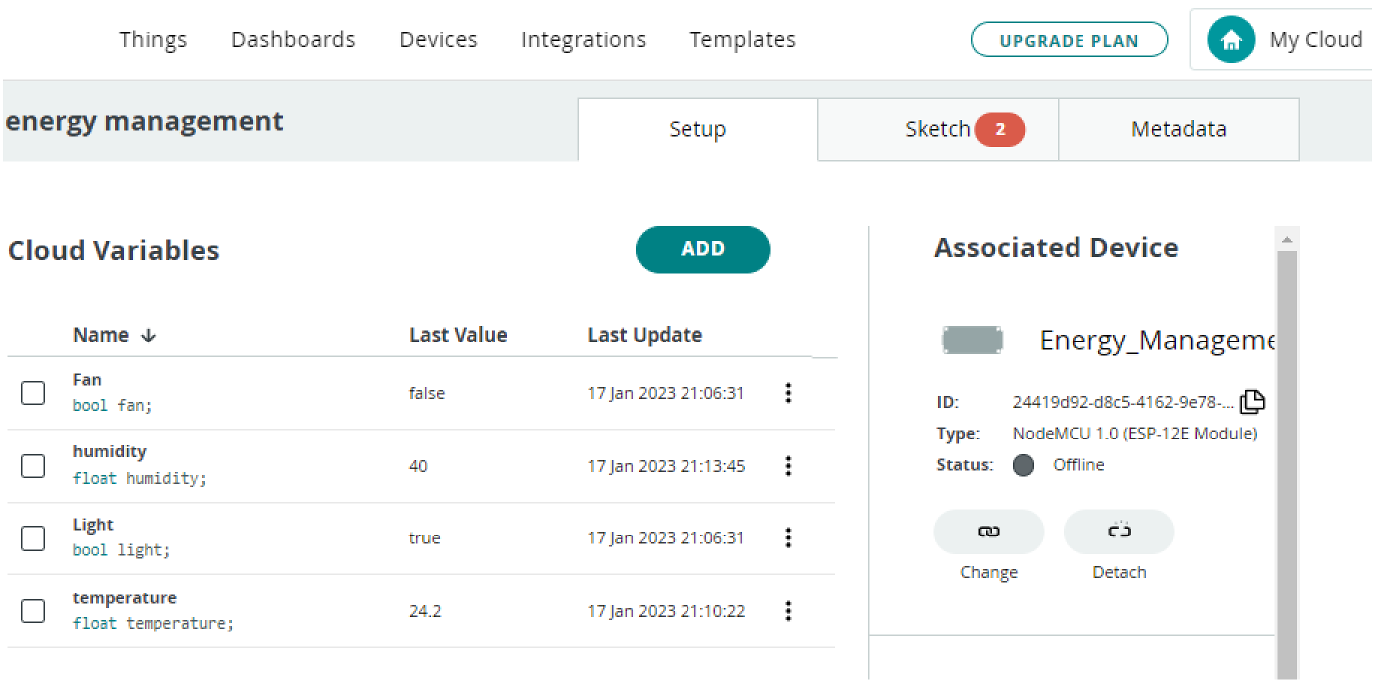
**Pin D2 OF nodemcu is connected to relay for fan. Similarly, the ground and voltage pin of relay is connected with nodemcu ground and 3v pin. The main power for fan is connected with normally open of relay**

**Pin D3 OF nodemcu is connected to relay for bulb. Similarly, the ground and voltage pin of relay is connected with nodemcu ground and 3v pin. The main power for fan is connected with normally open of relay.**

## DATA STRUCTURE:

**The data structure for this IoT temperature control system can be as simple as two single variables to store the current temperature and humidity reading from the sensor. This variable can be stored in the memory of the NodeMCU and can be accessed and updated by the program running on the device. Additionally, the system is connected to a remote server or cloud service, the data structure may also include fields for storing information such as device ID, Boolean variables for cooling and heating to store the value and to show it on the dashboard.**

## VARIABLES:



| Name | Last Value | Last Update |
|---|---|---|
| Fan `bool fan;` | false | 17 Jan 2023 21:06:31 |
| humidity `float humidity;` | 40 | 17 Jan 2023 21:13:45 |
| Light `bool light;` | true | 17 Jan 2023 21:06:31 |
| temperature `float temperature;` | 24.2 | 17 Jan 2023 21:10:22 |

Energy_Manageme

ID: 24419d92-d8c5-4162-9e78-...
Type: NodeMCU 1.0 (ESP-12E Module)
Status: ● Offline

Change
Detach

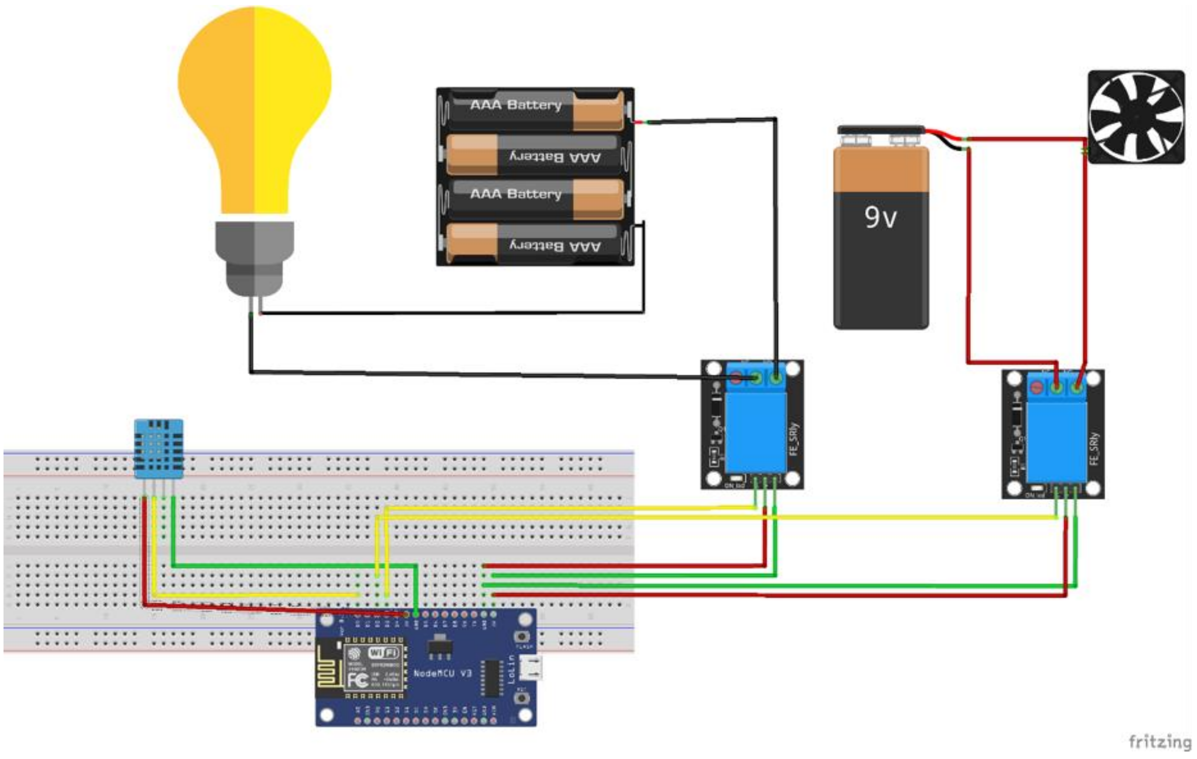
## THE THINGS:

Designing an IoT device, also known as "the things" in IoT terminology, involves several key components:

1. **Sensors:** These are the devices that collect data from the physical environment. In this case, DHT11 sensor would be the sensor used to collect temperature and humidity data.
2. **Microcontroller:** This is the brain of the IoT device, responsible for processing the sensor data and controlling the actuators. In this case, the NodeMCU microcontroller would be used to process the temperature and humidity data and control the relays for the fan and heating element.
3. **Actuators:** These are the devices that perform a physical action based on the sensor data and microcontroller commands. In this case, the relays would be the actuators that switch the fan and heating element on and off based on the temperature data.
4. **Communication interface:** This is the component that allows the IoT device to connect to the internet or a local network. This can be accomplished using WiFi communication technologies.
5. **Power supply:** This is the component that provides power to the IoT device. This can be achieved through a battery or a connection to a power source such as an adapter.
6. **User Interface:** this component is used to display the status of the device. It can be a simple LED or a more complex display such as a LCD.

All of these components are integrated into a single device, which can be programmed to perform specific tasks and communicate with other devices or servers.

## CIRCUIT DIAGRAM:



## NETWORKING CONFIGURATION:

The network configuration for our IoT system involve connecting the NodeMCU to a local network and configuring it to communicate with other devices or servers. To connect to the NodeMCU, IoT client will need to know the IP address of the NodeMCU on the local network, and also require a username and password for authentication.

## IoT CLIENTS:

Iot Clients are typically software applications that run on devices such as smartphones, tablets or laptops that allows users to interact with and control the Iot devices in system. Arduino Iot Cloud was used by us for our project. But different cloud services like AWS IoT, Google IoT core, Microsoft Azure IoT central, IBM Watson IoT platform, etc can also be used to connect IoT devices and access them remotely.

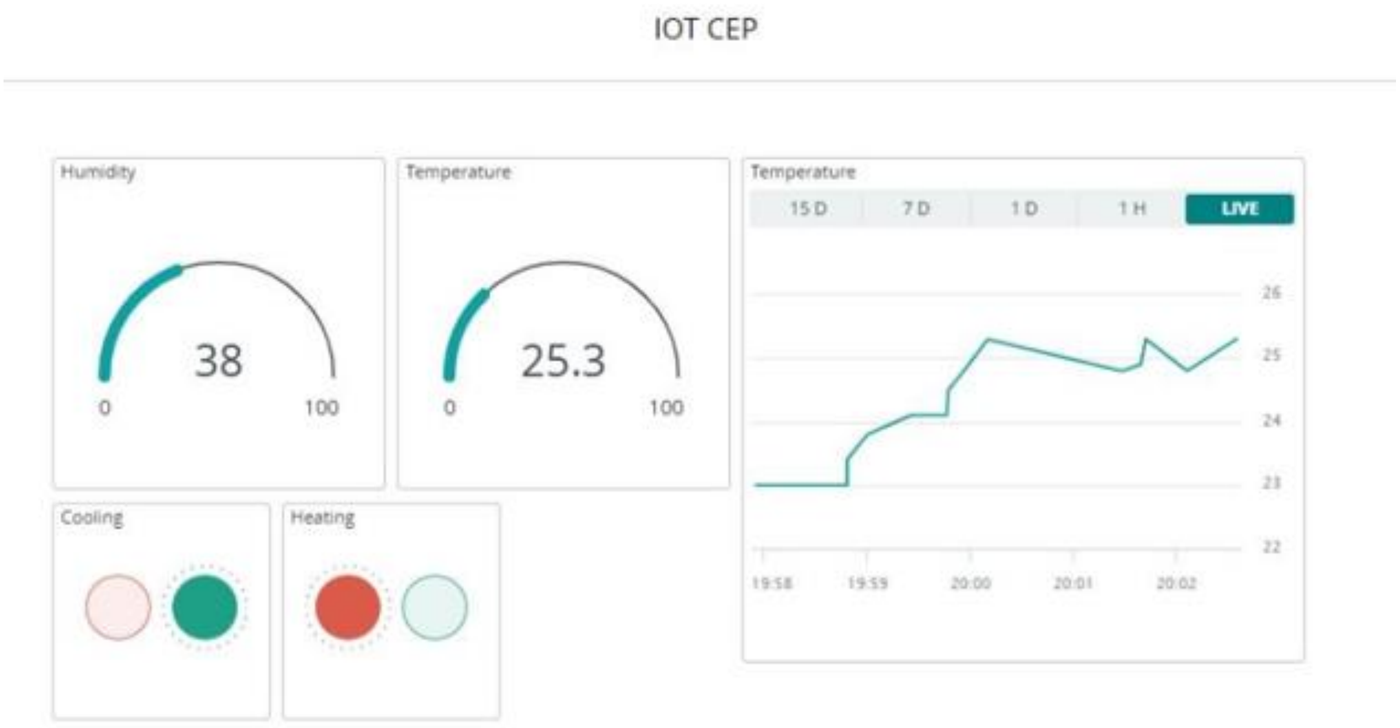
## ARDUINO CODING:

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/6be09864-dfbc-4521-910e-
c24ad694aa16

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when
changes are made to the Thing

  float humidity;
  float temperature;
  bool fan;
  bool light;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this
sketch.
*/

#include "thingProperties.h"
#include "DHT.h"
#define DHTPIN D1
#define DHTTYPE DHT11
#define relay1 D2 //fan
#define relay2 D3 //bulb

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if
none is found
  delay(1500);
   Serial.println("Humidity and Temperature\n\n");
  dht.begin();
  pinMode(relay1,OUTPUT);
  pinMode(relay2,OUTPUT);

  // This delay gives the chance to wait for a Serial Monitor without blocking if
none is found
```

```
  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  temperature = t;
  humidity = h;
  delay(800);
  if(t>25)
  {
     digitalWrite(relay1,LOW);
     digitalWrite(relay2,HIGH);
     fan=HIGH;
     light=LOW;
    //Serial.println("FAN ON");
  }
  else
  {
    digitalWrite(relay1,HIGH);
    digitalWrite(relay2,LOW);
    fan=LOW;
    light=HIGH;
    // Serial.println("FAN OFF");
  //  digitalWrite(relay2,LOW);
  // Serial.println("BULB OFF");
  }
}
//

/*
```

```
  Since Temp is READ_WRITE variable, onTempChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onTempChange()  {
  // Add your code here to act upon Temp change
}

/*
  Since Humidity is READ_WRITE variable, onHumidityChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onHumidityChange()  {
  // Add your code here to act upon Humidity change
}

/*
  Since Temperature is READ_WRITE variable, onTemperatureChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onTemperatureChange()  {
  // Add your code here to act upon Temperature change
}

/*
  Since Fan is READ_WRITE variable, onFanChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onFanChange()  {
  // Add your code here to act upon Fan change
}

/*
  Since Light is READ_WRITE variable, onLightChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLightChange()  {
  // Add your code here to act upon Light change
}
```

## DASHBOARD:



### Temperature:
A gauge widget is used show the real time environment temperature.

### Humidity:
A gauge widget is used show the real time environment humidity.

### Graph:
A graph widget is used to study the temperature variation for past 15 days. It can be used to check defect if the temperature variation doesn't lie in our defined range. It indicates that the sensor can be subjected to many possible errors.

### Cooling and heating indicators:
It enables the remote checking facility of the system, if the heating or cooling is occurring in the environment.