A PROJECT REPORT

ON

REMOTE MONITORING OF PERISHABLE GOODS USING IOT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF

B. Tech

BY

K. SATYA SHIVA SAI RAM	187Y5A0532
CH. ASHWITHA	187Y1A04C5
T. MANISHA	187Y1A0486

UNDER THE ESTEEMED GUIDANCE OF

Marri Laxman Reddy Institute of Technology and Management

2019

ACKNOWLEDGEMENT

A dissertation work, which integrated skill full and hard work need a great deal of guidance and helping hand. I would like to add a few heartfelt words for the people who have directly or indirectly in preparing in this dissertation work in numerous ways.

My heartfelt thanks to **Marri Laxman Reddy**, chairman of our college for providing me an opportunity to care out this project work form this college.

My sincere thanks to **Dr. K Venkateswara Reddy**, the principal of our college for providing me the required facilities to complete my project.

With profound sense of gratitude and regards, I acknowledge with great pleasure the Guidance and support extended by, **Mr. K Abdul Basith & Mr. K Nagabhushanam**, heads of the Computer Science & Electronics and Communication Engineering departments and internal guide, **Mr. B Prasad** for his timely suggestion, logical thought that have greatly contributed in bringing out the project in a systematic way. He has been inculcating innovative skills and technicalities at all stages at performing this project.

My sincere thanks to all the teaching and non- teaching staff of our department for their constant co-operation extended to me to complete the project successfully. My thanks to all my classmates and friends for their constant help and support during my project work.

My heartfelt gratitude to my parents and my family for their constant encouragement throughout the work carried out to prepare this project report.

SUBMITTED BY

BATCH NO- 04

K. SATYA SHIVA SAI RAM	187Y5A0532
CH. ASHWITHA	187Y1A04C5
T. MANISHA	187Y1A0486

ABSTRACT

Food safety is imperative to avoid food borne diseases and to ensure the public health. Monitoring of perishable food products and early detection of degradation will avoid loss due to food wastage and also ensures the freshness of food. In this scenario, remote monitoring of fruits during transportation from field to shelf can ensure the quality of fruit. In this work, a wireless sensor network was designed for monitoring of fruits during transportation and even after storage. Internet of things was also used for facilitating online monitoring of fruits from any remote location. NodeMCU was used as sensor node and gateway node. It has performed the fusion of sensor data such as temperature, humidity and moisture to avoid redundant data storage and increase the efficiency of decision making.

Our project is not only used in during transportation, it is used in foods cold storage, medical cold storage etc. We can monitor form any place and can be controlled, because it was wireless communication.

Index

Introduction 1

Block Diagram 2

Hardware 3

NodeMCU 3

NRF Module 4

DHT11 Sensor 6

Software 8

Arduino IDE 8

IBM Cloud 9

IOT Platform 9

Node-Red 10

MIT app inventor-2 11

Procedure 12

Software implementation 12

Arduino IDE 12

Node-RED 16

Mit app inventor 2 21

Hardware implementation 26

NodeMCU 26

Project Description 29

Conclusion 30

INTRODUCTION

Nowadays, markets and consumers are more demanding of quality services in all areas, including transportation or merchandise. Transporting goods is a complicated endeavor and more so when it comes to products that are sensitive, such as perishable goods.

Keeping fish or fruit from other countries in excellent condition for when they arrive to the consumer's table is not an easy task. It requires a complex, quality system throughout the entire logistics process, from origin to the final point of distribution.

A perishable good is any **product in which quality deteriorates due to environmental conditions through time**, such as **meat and meat by-products, fish and seafood, dairy products, fruit and vegetables, flowers, pharmaceutical products, and chemicals**.

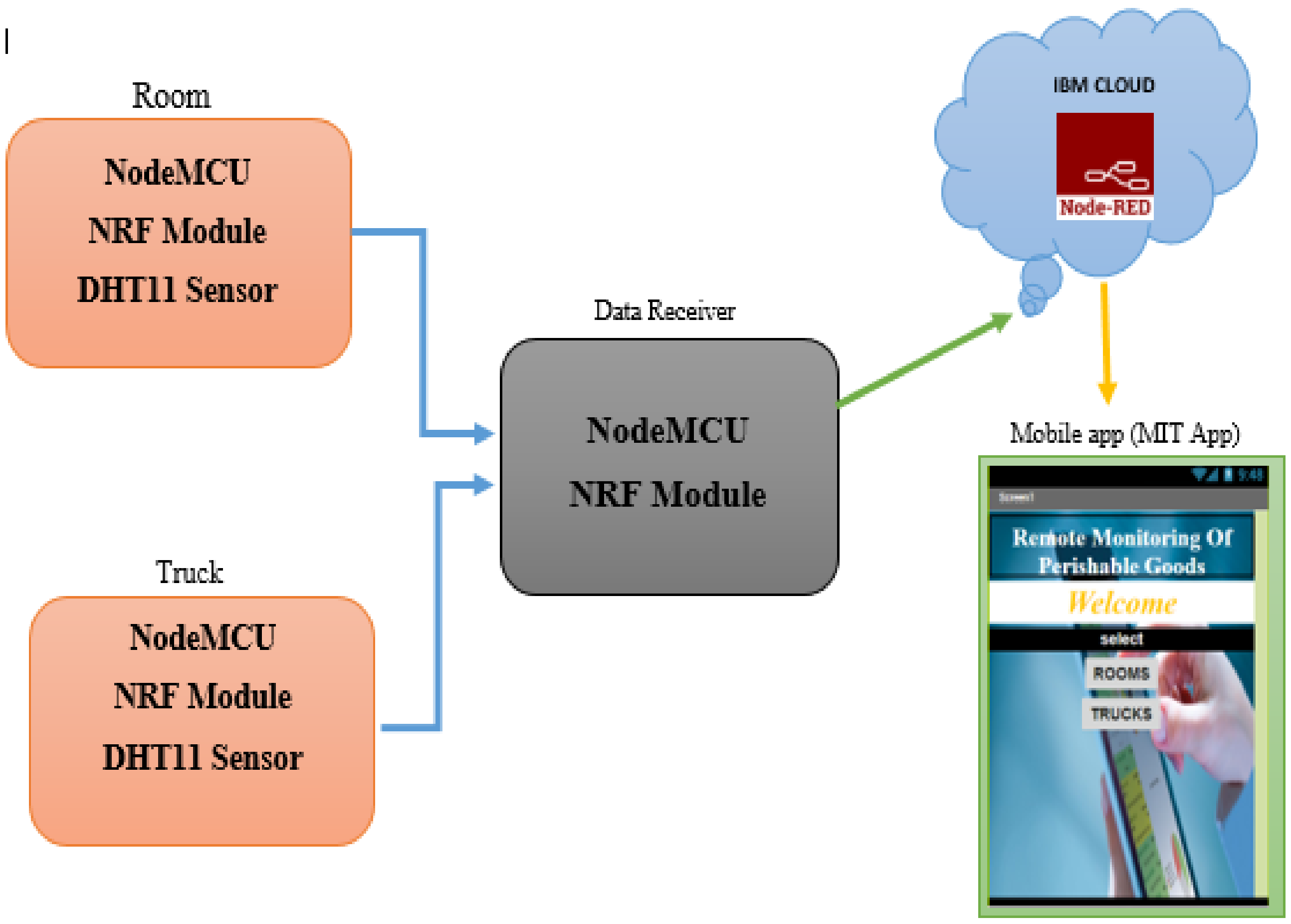
Due to their chemical and/or physiological characteristics, these products have short lifespans; they are more susceptible to severe and irreparable damage during transport, especially if temperature is not kept consistent. These products must be handled with the utmost caution and efficiency in order to preserve them and keep them in excellent condition when they reach the final consumer.

In order for this to happen, the key factors to keep in mind are **time, isolation, and holding temperature.**

Most losses occur between post-harvest and product distribution. These damages affect the final consumer and create major losses for the businesses that sell them.

As previously mentioned, the main issue when transporting perishable goods is to respect the cold chain as much as possible, since it guarantees that the properties of your products are kept intact at all times.

Block Diagram:



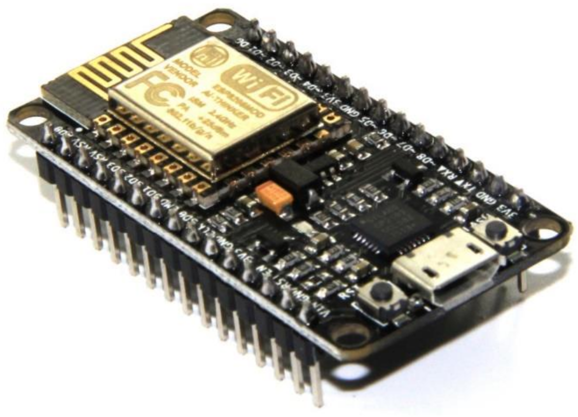
Hardware

Hardware used in this project

- NodeMCU-2
- NRF Module-2
- DHT11 sensor-1

NodeMCU:

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.



NodeMCU provides access to the GPIO (General Purpose Input/Output) and a pin mapping table is part of the API documentation.

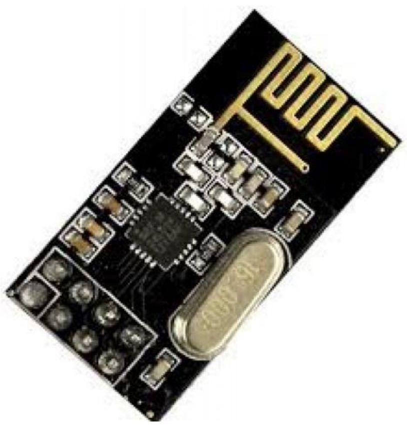
[*] D0 (GPIO16) can only be used for GPIO read/write. It does not support open-drain/interrupt/PWM/I²C or 1-Wire.

I/O index	ESP8266 pin	I/O index	ESP8266 pin
0 [*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3
3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12	-	-

NRF Module:

The n**RF24L01 is a wireless transceiver module**, meaning each module can both send as well as receive data. They operate in the frequency of 2.4GHz, which falls under the ISM band and hence it is legal to use in almost all countries for engineering applications. The modules when operated efficiently can cover a distance of 100 meters (200 feet) which makes it a great choice for all wireless remote controlled projects.

The module operates at 3.3V hence can be easily used with 3.2V systems or 5V systems. Each module has an address range of 125 and each module can communicate with 6 other modules hence it is possible to have multiple wireless units communicating with each other in a particular area. Hence mesh networks or other types of networks are possible using this module. So if you are looking for a wireless module with the above properties then this module would be an ideal choice for you.



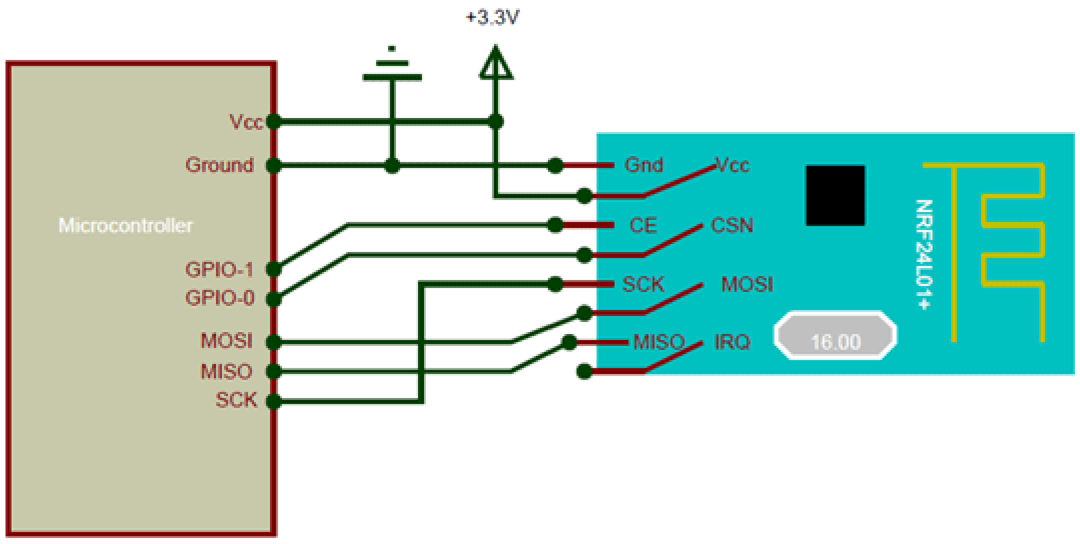
NRF Module Features

- 2.4GHz RF transceiver Module
- Operating Voltage: 3.3V
- Nominal current: 50mA
- Range : 50 – 200 feet
- Operating current: 250mA (maximum)
- Communication Protocol: SPI
- Baud Rate: 250 kbps - 2 Mbps.
- Channel Range: 125
- Maximum Pipelines/node : 6
- Low cost wireless solution

NRF Module Pin Configuration:

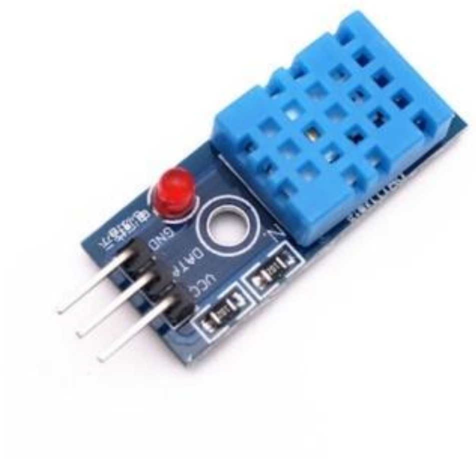
Pin Number	**Pin Name**	**Abbreviation**	**Function**
1	Ground	Ground	Connected to the Ground of the system
2	Vcc	Power	Powers the module using 3.3V
3	CE	Chip Enable	Used to enable SPI communication
4	CSN	Ship Select Not	This pin has to be kept high always, else it will disable the SPI
5	SCK	Serial Clock	Provides the clock pulse using which the SPI communication works
6	MOSI	Master Out Slave In	Connected to MOSI pin of MCU, for the module to receive data from the MCU
7	MISO	Master In Slave Out	Connected to MISO pin of MCU, for the module to send data from the MCU

8	IRQ	Interrupt	It is an active low pin and is used only if interrupt is required



DHT11 Sensor:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.



TECHNICAL DETAILS:

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

Connecting Wires:

These wires are used to connect the NodeMCU and NRF module. And this wires are connected to DHT11 sensor also. The wires will transmit data among the devices.

Software

Software used to implement the project.

- Arduino IDE

IBM Cloud

IOT Platform

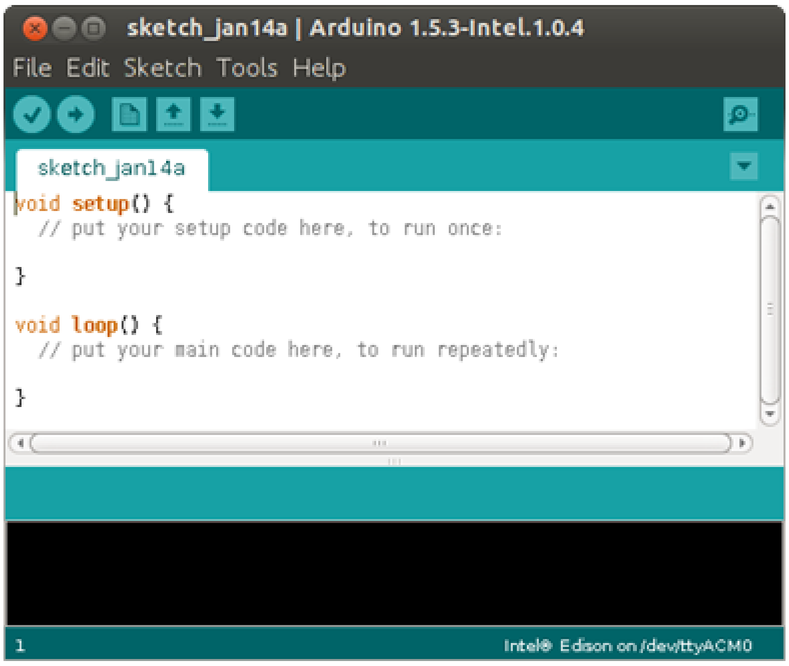
Node-Red

- MIT app inventor-2

Arduino IDE:

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiringproject, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.


sketch_jan14a | Arduino 1.5.3-Intel.1.0.4
File Edit Sketch Tools Help
sketch_jan14a
1
Intel® Edison on /dev/ttyACM0

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

IBM Cloud

1. IOT Platform:

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. It basically connects your hardware, however diverse, to the cloud by using flexible connectivity options, enterprise-grade security mechanisms, and broad data processing powers. For developers, an IoT platform provides a set of ready-to-use features that greatly speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility.

Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

Internet of Things Platform

2. Node-Red:

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14 MQTT nodes can make properly configured TLS connections.

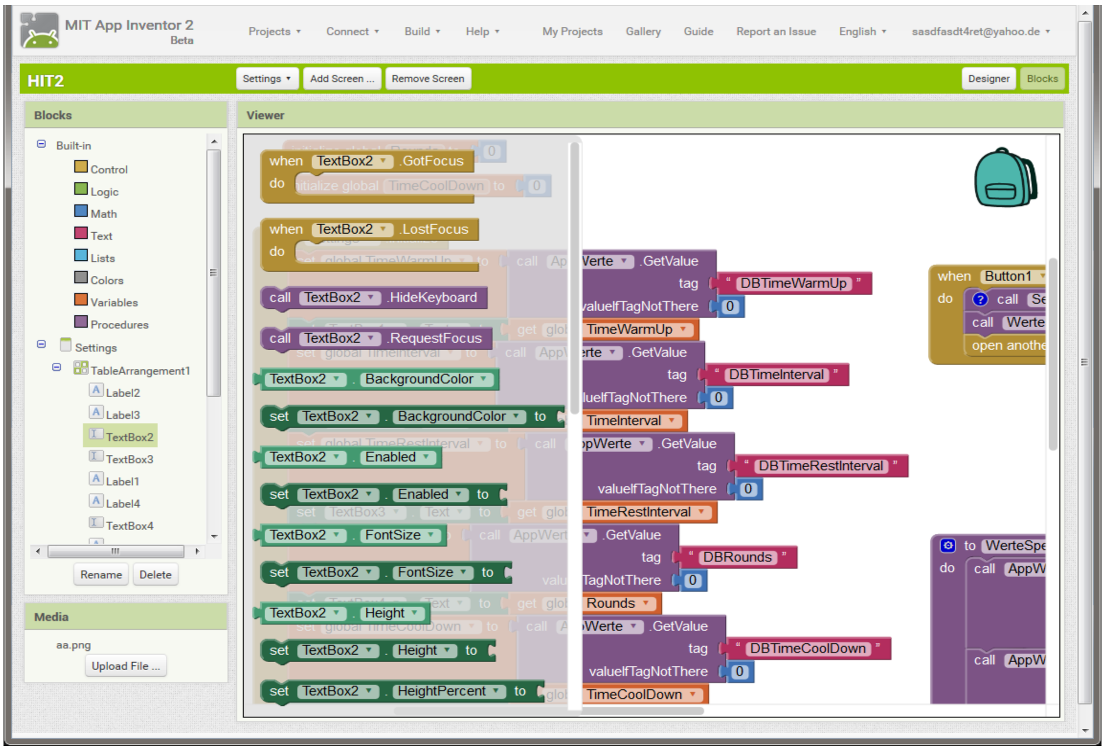
Node-RED

3. MIT app inventor-2:

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT), which allows newcomers to computer programming to create software applicationsfor the Android operating system (OS).

It uses a graphical interface very similar to Scratch and the StarLogo TNG user interface, which allows users to drag-and-dropvisual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

MIT
APP INVENTOR



Procedure

SOFTWARE IMPLEMENTATION

Arduino IDE:

We need to write a program to develop any project. Now first step is to write program in Arduino IDE. The program consists of transmitting and receiving of data from NRF Module. The NodeMCU is connected to the system and the NodeMCU board has to be selected to get the output. And we need to select the port(ex-COM3).

Transmitter Program:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#include "DHT.h"
#define DHTPIN D2
#define DHTTYPE DHT11

RF24 radio(D4,D8); // CE, CSN
const byte address[6] = "00001";
DHT dht(DHTPIN, DHTTYPE);


void setup()
{
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
  Serial.begin(115200);
  dht.begin();
}

void loop()
{

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t) )
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
  delay(5);
  radio.stopListening();
  //const char text[] = "&h";
  radio.write(&t, sizeof(t));
  delay(2000);
  radio.write(&h, sizeof(h));
  delay(2000);
}
```

Transmitter program will transmit the data to the receiver. In our program the NRF Module which is connected to Transmitter will transmit the data which is present in the program.

Receiver program:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(D4,D8); // CE, CSN
const byte address[6] = "00001";
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

//-------- Customise these values -----------
const char* ssid = "MLRITM@DGL";
const char* password = "digi@123";
#define ORG "pxft16"
#define DEVICE_TYPE "nodemcu32"
#define DEVICE_ID "kammasairam"
```

```
#define TOKEN "0123456789"
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char pubtopic[] = "iot-2/evt/dht11/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883,wifiClient);
void setup() {
  Serial.begin(115200);
   Serial.println();
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
   Serial.print("Connecting to ");
 Serial.print(ssid);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
 }
 Serial.println("");

 Serial.print("WiFi connected, IP address: ");
 Serial.println(WiFi.localIP());
}

void loop() {
  float t,h;
  if (radio.available()) {
    radio.read(&t, sizeof(t));
    Serial.print("temperature: ");
    Serial.println(t);
    delay(2000);
     radio.read(&h, sizeof(h));
     Serial.print("humidity: ");
    Serial.println(h);
    delay(2000);
  }
  PublishData(t,h);
delay(1000);
}
```

```
void PublishData(float t, float h){
 if (!client.connected()) {
 Serial.print("Reconnecting client to ");
 Serial.println(server);
 while (!client.connect(clientId, authMethod, token)) {
 Serial.print(".");
 delay(500);
 }
 Serial.println();
 }
  String payload = "{\"d\":{\"temperature\":";
  payload += t;
  payload+="," "\"humidity\":";
  payload += h;
  payload += "}}";
 Serial.print("Sending payload: ");
 Serial.println(payload);

 if (client.publish(pubtopic, (char*) payload.c_str()))
 {
 Serial.println("Publish ok");
 }
 else
 {
 Serial.println("Publish failed");
 }
}
```

Receiver program will get the data from the transmitter. In our program the NRF Module which is connected to receiver program will get the data from transmitter. The NRF Module is connected to the NodeMCU. The data is readied from DHT11 sensor.

Node-RED:

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

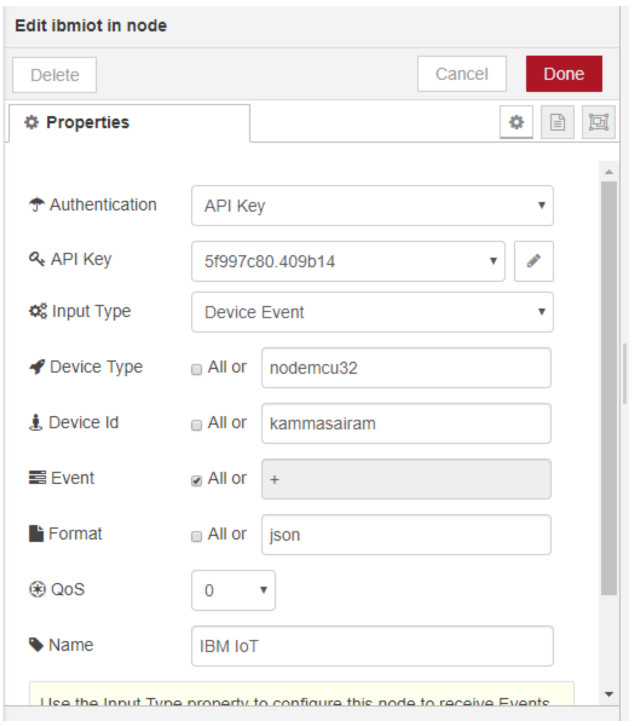
Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON.

we need to create a flow and we need to implement the process in given space. The following steps are used to implement the program.

- First, drag the input node

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets msg.payload to be a String containing the payload of the incoming message.

Set the properties of the input node as shown below.



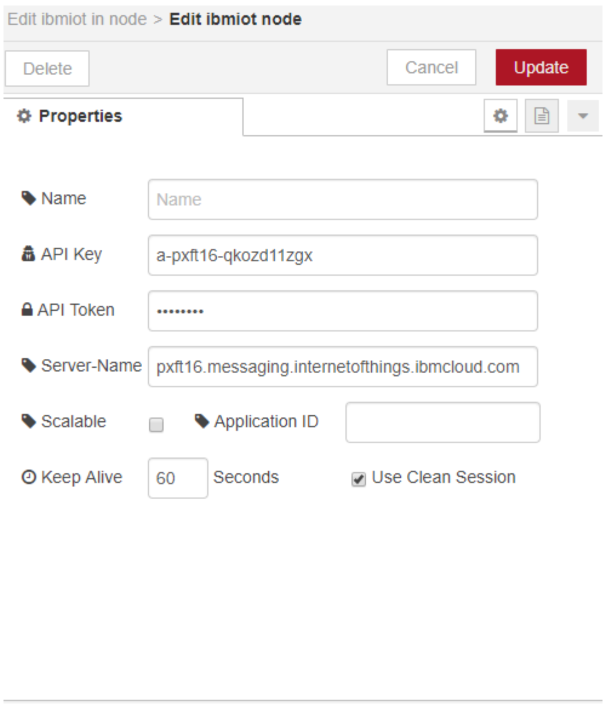
Edit ibmiot in node

Delete | Cancel | Done

Properties

Authentication	API Key
API Key	5f997c80.409b14
Input Type	Device Event
Device Type	☐ All or nodemcu32
Device Id	☐ All or kammasairam
Event	✓ All or +
Format	☐ All or json
QoS	0
Name	IBM IoT

Use the Input Type property to configure this node to receive Events



Edit ibmiot in node > Edit ibmiot node

Delete | Cancel | Update

Properties

Name	Name
API Key	a-pxft16-qkozd11zgx
API Token	••••••••
Server-Name	pxft16.messaging.internetofthings.ibmcloud.com
Scalable	☐
Application ID	
Keep Alive	60 Seconds
Use Clean Session	✓

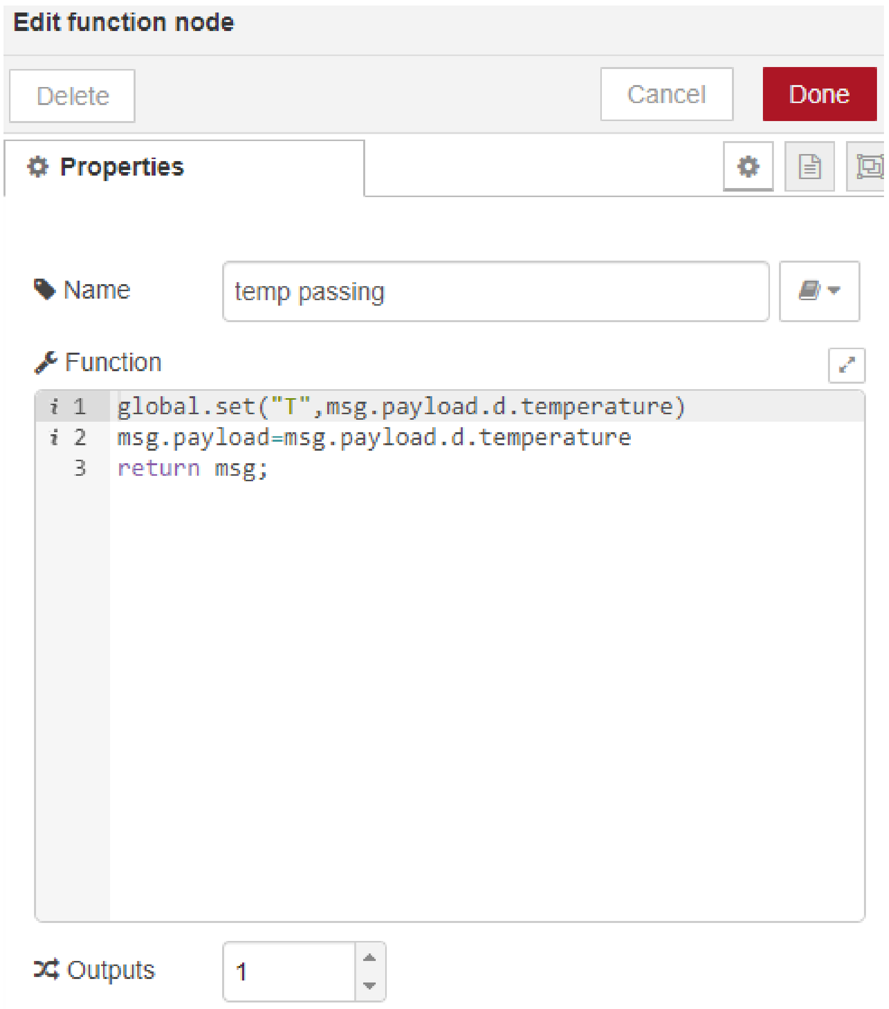
- Next drag the Debug node

The debug sidebar provides a structured view of the messages it is sent, making it easier to understand their structure. JavaScript objects and arrays can be collapsed and expanded as required. Buffer objects can be displayed as raw data or as a string if possible.

- Function node

A JavaScript function block to run against the messages being received by the node. The messages are passed in as a JavaScript object called msg. By convention it will have a msg.payloadproperty containing the body of the message. The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

Temperature function node



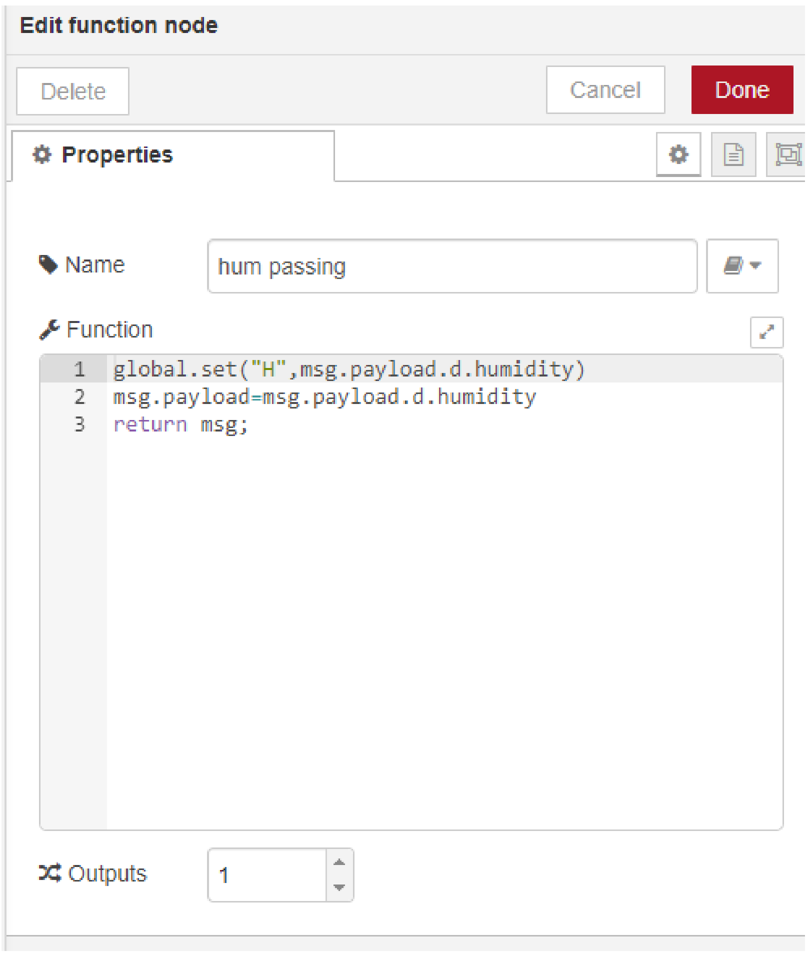
Name: temp passing

Function

```
global.set("T",msg.payload.d.temperature)
msg.payload=msg.payload.d.temperature
return msg;
```

Outputs: 1

Humidity function Node



Name: hum passing

Function

```
global.set("H",msg.payload.d.humidity)
msg.payload=msg.payload.d.humidity
return msg;
```

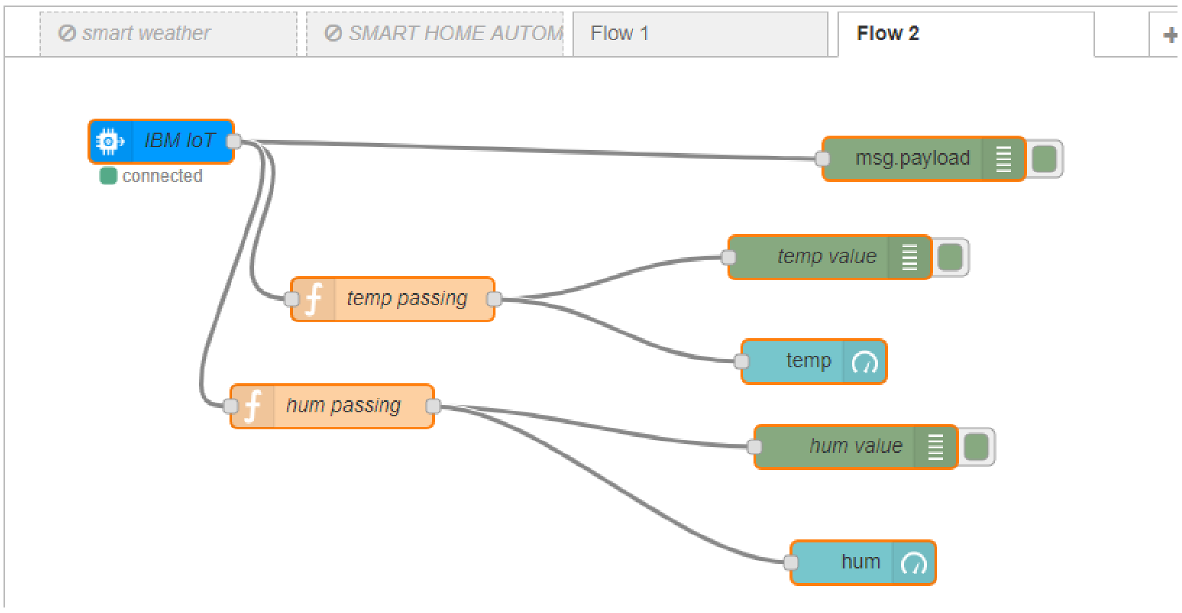
Outputs: 1

- Gauge node

Adds a gauge type widget to the user interface. The msg.payload is searched for a numeric value and is formatted in accordance with the defined Value Format, which can then be formatted using Angular filters.

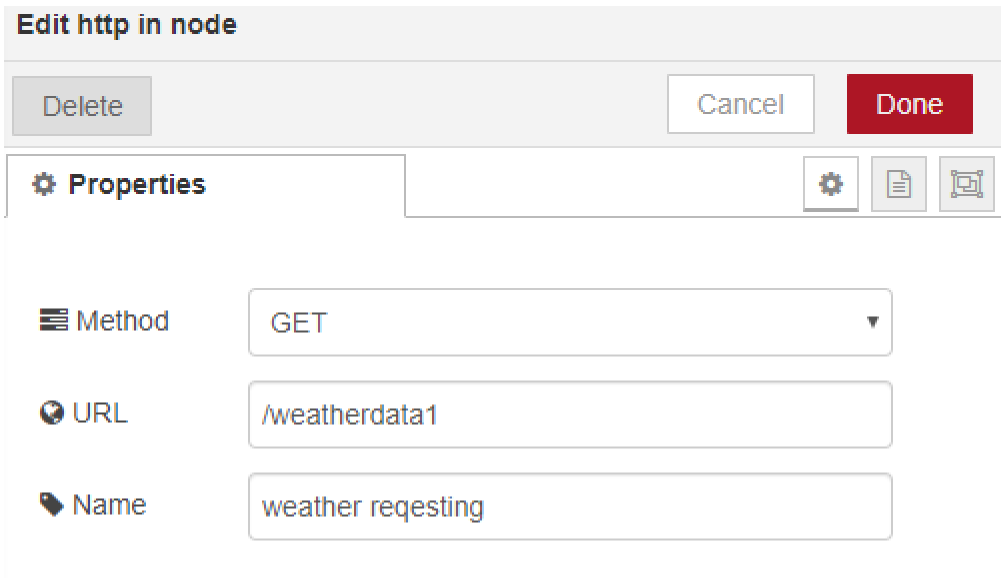
For example: {{value | number:1}}%will round the value to one decimal place and append a % sign.

- The connections of all node are shown below



- http node

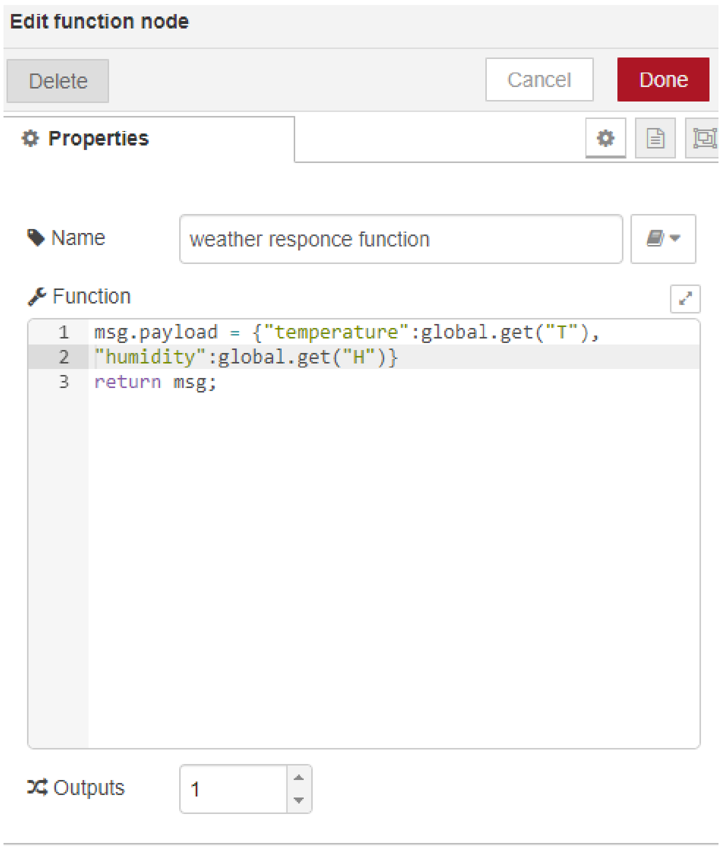
Creates an HTTP end-point for creating web services. And set an URL in the properties of the http node



Method: GET
URL: /weatherdata1
Name: weather reqesting

- **function node in http**

  A JavaScript function block to run against the messages being received by the node. We need to convert the json format to normal text



Name: weather responce function

Function

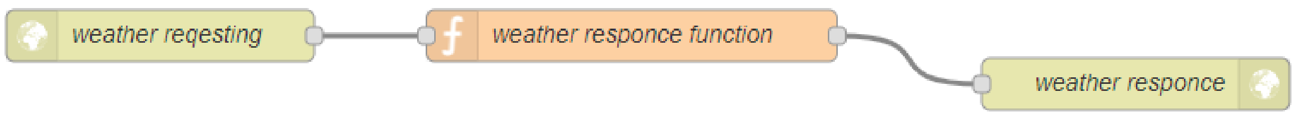
```
msg.payload = {"temperature":global.get("T"),
"humidity":global.get("H")}
return msg;
```

Outputs: 1

- **debug node of http**

  Sends responses back to requests received from an HTTP Input node.

  Final connection.



Mit app inventor 2

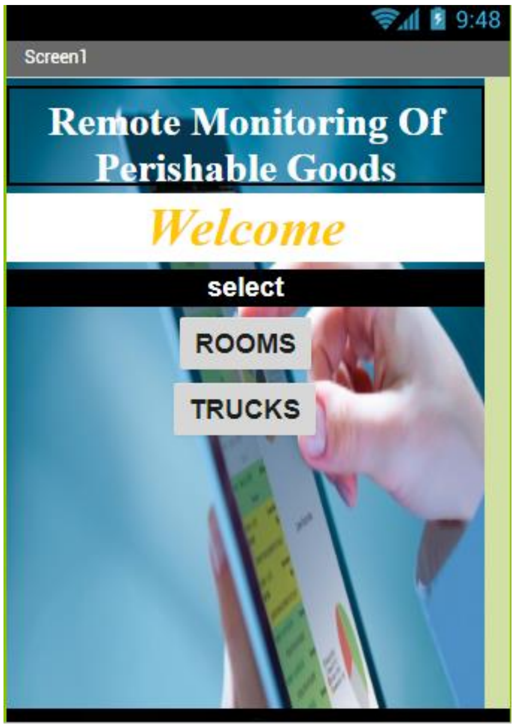
App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT), which allows newcomers to computer programming to create software applicationsfor the Android operating system (OS).

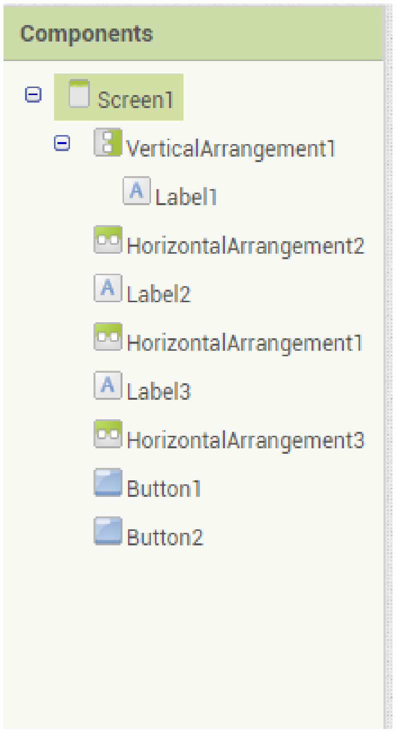
It uses a graphical interface very similar to Scratch and the StarLogo TNG user interface, which allows users to drag-and-dropvisual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

Steps:

- create new project
- drag the label to the screen and edit the properties
- drag the layout components and in it drag textbox, labels and set its properties.
- And add buttons as our requirement
- In blocks give conditions to each of the component.
- Add screens as per our requirement.

1.

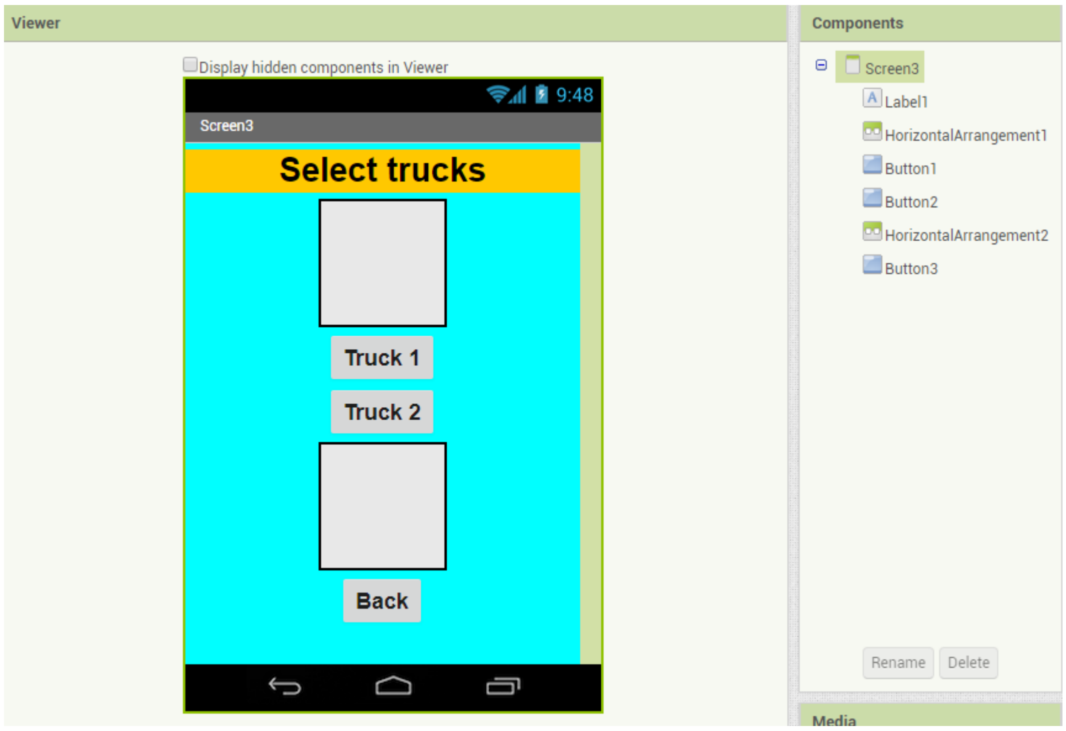




Components

- Screen1
  - VerticalArrangement1
    - Label1
  - HorizontalArrangement2
  - Label2
  - HorizontalArrangement1
  - Label3
  - HorizontalArrangement3
  - Button1
  - Button2

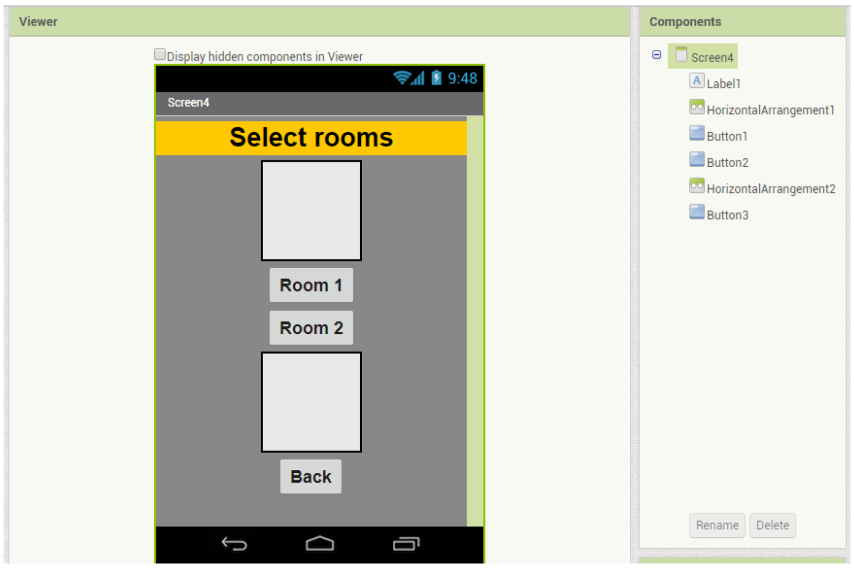
2.

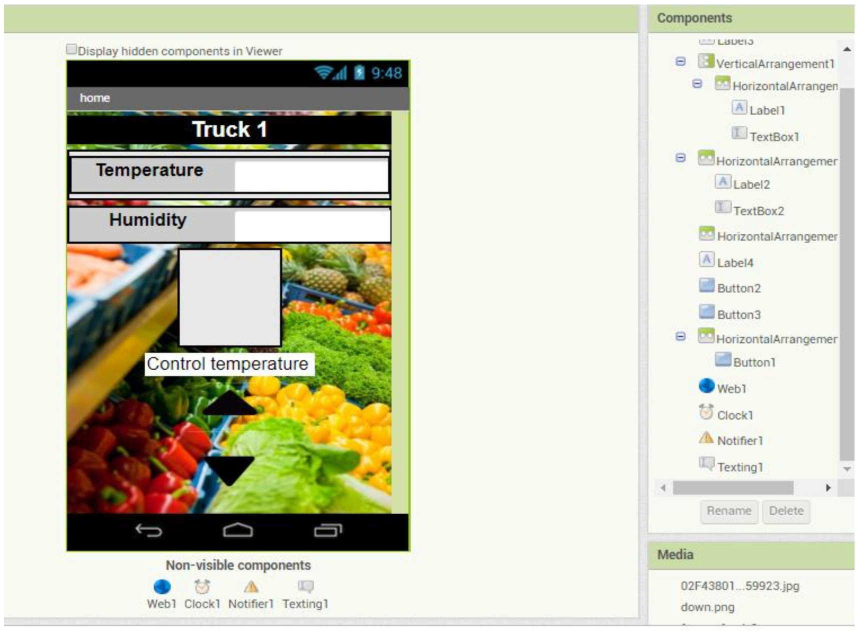


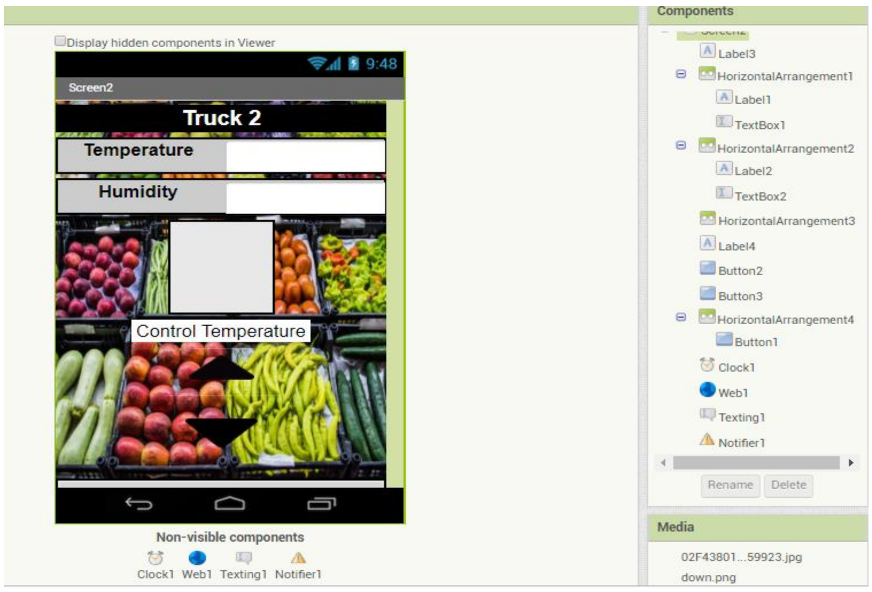
Components

- Screen3
  - Label1
  - HorizontalArrangement1
  - Button1
  - Button2
  - HorizontalArrangement2
  - Button3

3.

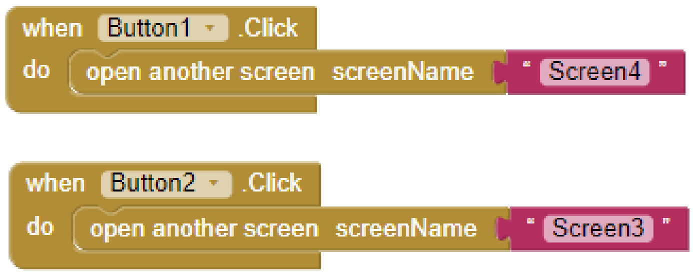

Viewer
Display hidden components in Viewer
9:48
Screen4
Select rooms
Room 1
Room 2
Back
Components
Screen4
Label1
HorizontalArrangement1
Button1
Button2
HorizontalArrangement2
Button3
Rename
Delete


Display hidden components in Viewer
9:48
home
Truck 1
Temperature
Humidity
Control temperature
Non-visible components
Web1
Clock1
Notifier1
Texting1
Components
VerticalArrangement1
HorizontalArrangem
Label1
TextBox1
HorizontalArrangemer
Label2
TextBox2
HorizontalArrangemer
Label4
Button2
Button3
HorizontalArrangemer
Button1
Web1
Clock1
Notifier1
Texting1
Rename
Delete
Media
02F43801...59923.jpg
down.png


Display hidden components in Viewer
9:48
Screen2
Truck 2
Temperature
Humidity
Control Temperature
Non-visible components
Clock1
Web1
Texting1
Notifier1
Components
Label3
HorizontalArrangement1
Label1
TextBox1
HorizontalArrangement2
Label2
TextBox2
HorizontalArrangement3
Label4
Button2
Button3
HorizontalArrangement4
Button1
Clock1
Web1
Texting1
Notifier1
Rename
Delete
Media
02F43801...59923.jpg
down.png

Coding part:

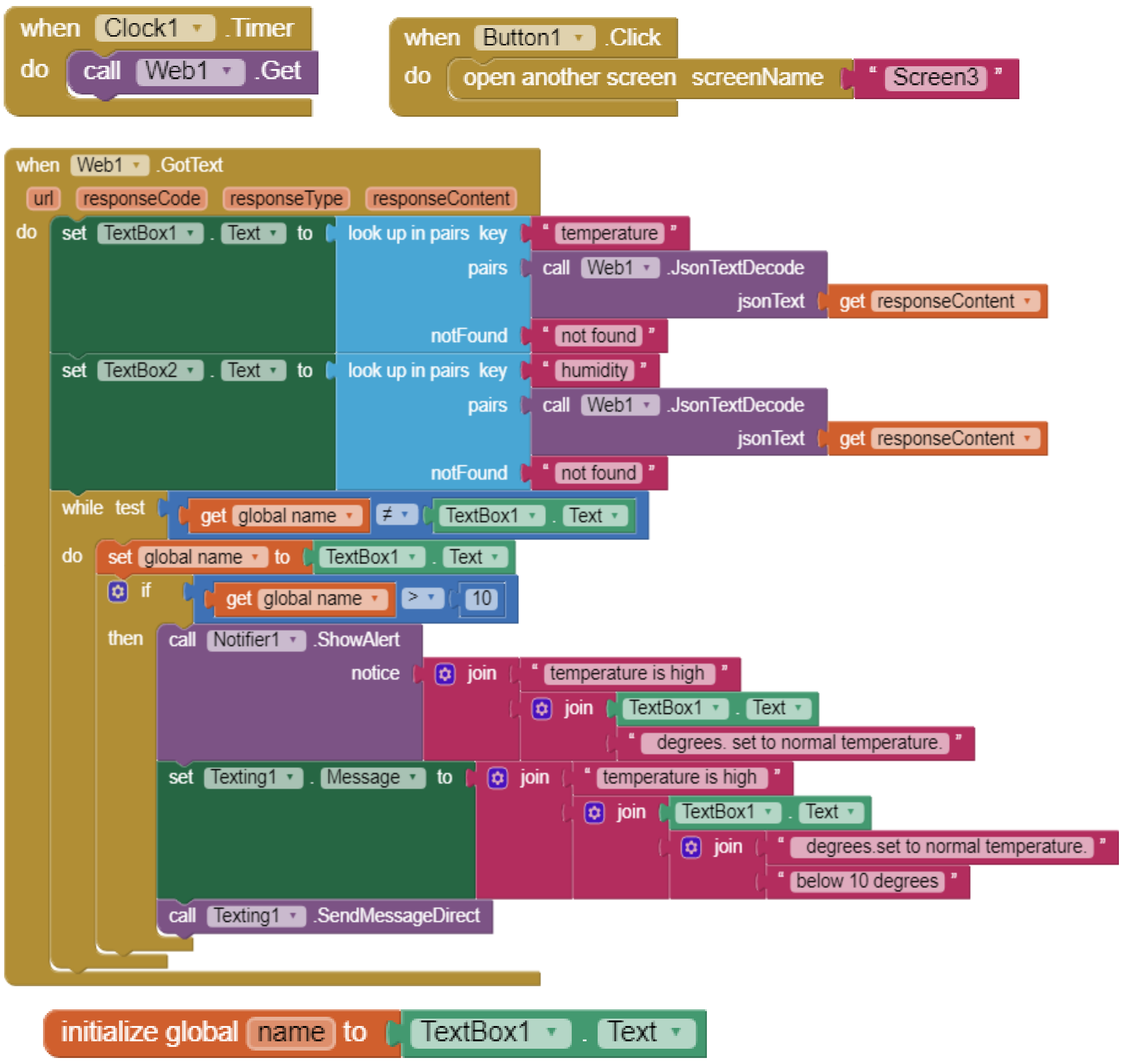
- **Screen 1**



```
when Button1 .Click
do  open another screen  screenName  " Screen4 "

when Button2 .Click
do  open another screen  screenName  " Screen3 "
```

- **Code of truck and room**



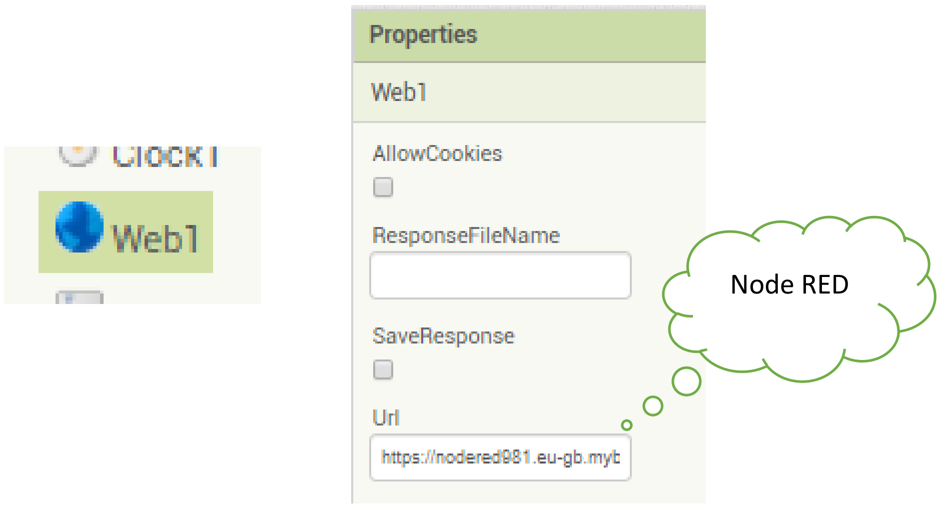
```
when Clock1 .Timer
do  call Web1 .Get

when Button1 .Click
do  open another screen  screenName  " Screen3 "

when Web1 .GotText
 url  responseCode  responseType  responseContent
do  set TextBox1 . Text to  look up in pairs  key  " temperature "
                                              pairs  call Web1 .JsonTextDecode
                                                                 jsonText  get responseContent
                                              notFound  " not found "
    set TextBox2 . Text to  look up in pairs  key  " humidity "
                                              pairs  call Web1 .JsonTextDecode
                                                                 jsonText  get responseContent
                                              notFound  " not found "
    while test  get global name ≠ TextBox1 . Text
    do  set global name to  TextBox1 . Text
        if  get global name > 10
        then  call Notifier1 .ShowAlert
                           notice  join  " temperature is high "
                                         join  TextBox1 . Text
                                               " degrees. set to normal temperature. "
              set Texting1 . Message to  join  " temperature is high "
                                               join  TextBox1 . Text
                                                     join  " degrees.set to normal temperature. "
                                                           " below 10 degrees "
              call Texting1 .SendMessageDirect

initialize global name to  TextBox1 . Text
```

- **Getting data from cloud**



Properties

Web1

AllowCookies ☐

ResponseFileName

SaveResponse ☐

Url https://nodered981.eu-gb.myb

The data will be shown in the mobile app.so we can monitor the temperature and control. If the temperature is high, we will get a message alert to our mobile.

HARDWARE IMPLEMENTATION

NodeMCU:

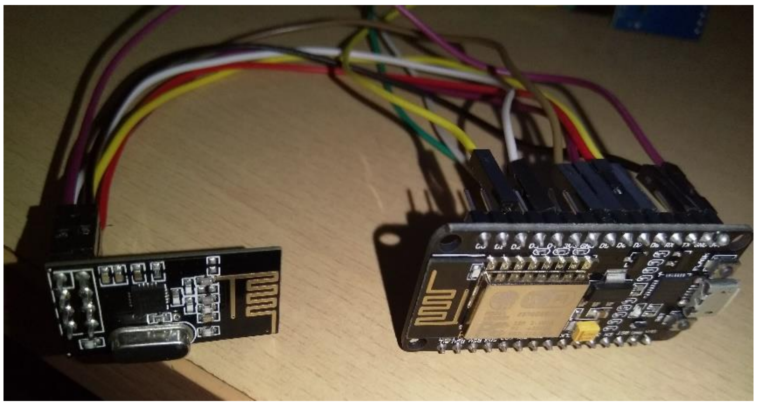
NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266.

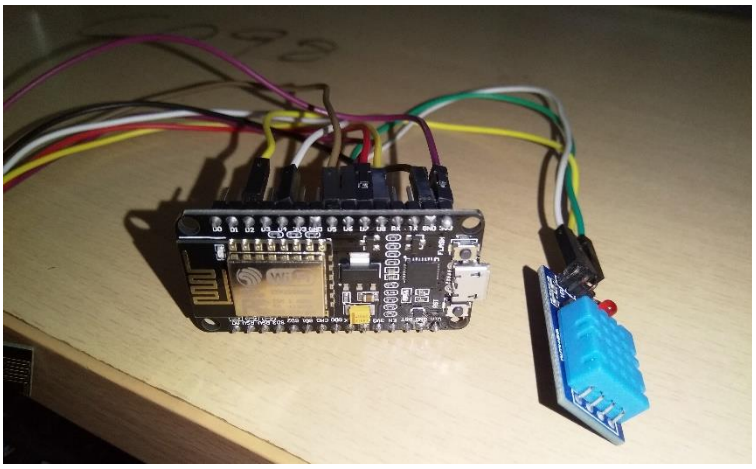
I/O index	ESP8266 pin	I/O index	ESP8266 pin
0 [*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3
3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12	-	-

NodeMCU Transmitter connections:

NRF Module	-	**NodeMCU**			
GND	-	GND	CE	-	D4
SCK	-	D5	MISO	-	D6
VCC	-	3.3V	CSN	-	D8
MOSI	-	D7			

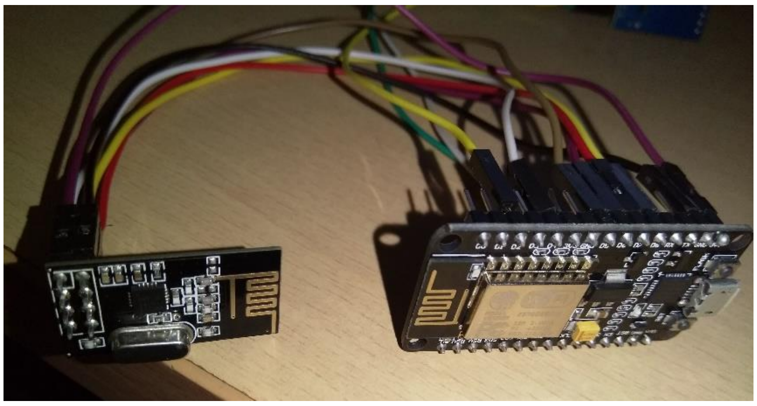
NodeMCU	-	DHT11 sensor
3.3V	-	VCC
GND	-	GND
D2	-	DATA





NodeMCU Receiver connections:

NRF Module	**-**	**NodeMCU**
GND	-	GND
CE	-	D4
SCK	-	D5
MISO	-	D6
VCC	-	3.3V
CSN	-	D8
MOSI	-	D7



- USB cable is used to give power supply to the NodeMCU. NodeMCU has a micro-USB for power.
- The NRF module is connected to the NodeMCU and transmits the data between the two NRF modules.

Project Description

- Improper temperature control is a key reason why medicines like vaccinations and other perishable cargo are wasted in the supply chain. A cold chain must be established and maintained to ensure goods have been properly refrigerated during every step of the process, making temperature monitoring a critical business function. To achieve this, we have designed a device which uses wireless sensor networks, gateway and cloud application. Alerts are triggered when the temperature is not in optimal degrees.
- NRF modules are used as wireless sensor networks, which are installed in any environment along the cold chain including trucks, storage units and production facilities. These NRF Modules collect sensor data (temperature) and pass the sensor data to a centralized gateway.
- The gateway sends the data to cloud so that customers can view data in real time.

Project Highlights:

- Wireless Senor Network with cloud integration
- NRF protocol to exchange the data between sensor nodes and gateway
- Data Visualization Application with Notification System

Conclusion

Food safety plays important role in countries economy and human health. WSN plays a vital role in controlling and monitoring applications, which further connected to IoT enables remote monitoring. Data storage is another problem in sensor networks and cloud databases.

NodeMCU will send the data to cloud with the help of Wi-Fi module in NodeMCU. The all other modules are connected to the NodeMCU. The NRF module will transmit the data between 2 NRF modules. In feature we may implement more than this according to the period and requirement.