LABORATORY MANNUAL

INTERNET OF THINGS

D. Ravi Babu
Assistant Professor

Computer Science & Engineering Department

S.R.K.R. ENGINEERING COLLEGE

CHINAAMIRAM::BHIMAVARAM

CSE 4.1.9	INTERNET OF THINGS LAB	
Instruction: 3 Periods	Univ. Exam: 3 Hours	**Credits: 2**
Internal: 50 Marks	**University Exam: 50 Marks**	**Total: 100 Marks**

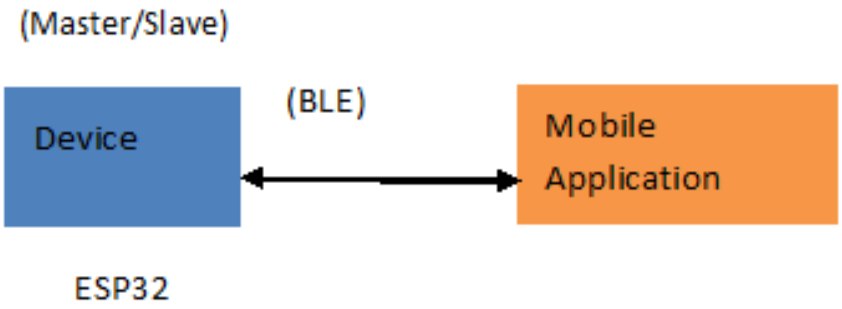
Contents

Experiment -1: Application of Bluetooth in IoT Systems

Aim: To study the applications of Bluetooth in Internet of Things (IoT) where to connect the Bluetooth to devices for authentication and send the commands to operate the appliances as per the requirements and the state of Bluetooth is enabled by programming.

Objectives: Student should get the knowledge of Bluetooth communication devices by designing Mobile App.

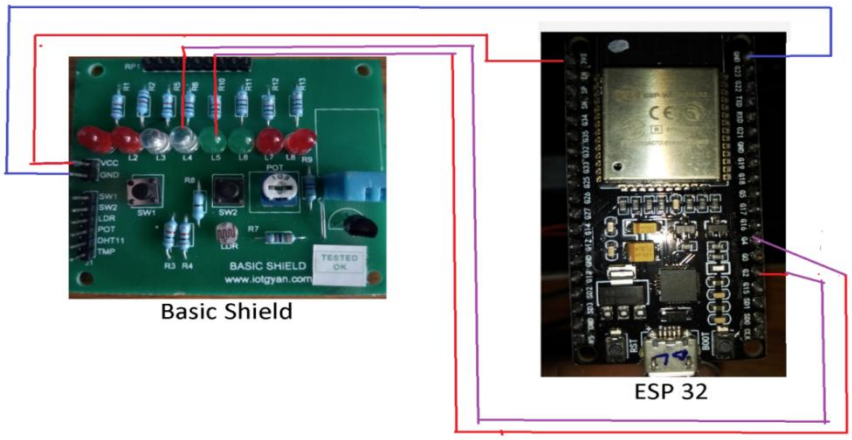
Outcomes: Student will be aware of communication by the Bluetooth background devices by using Mobile App.

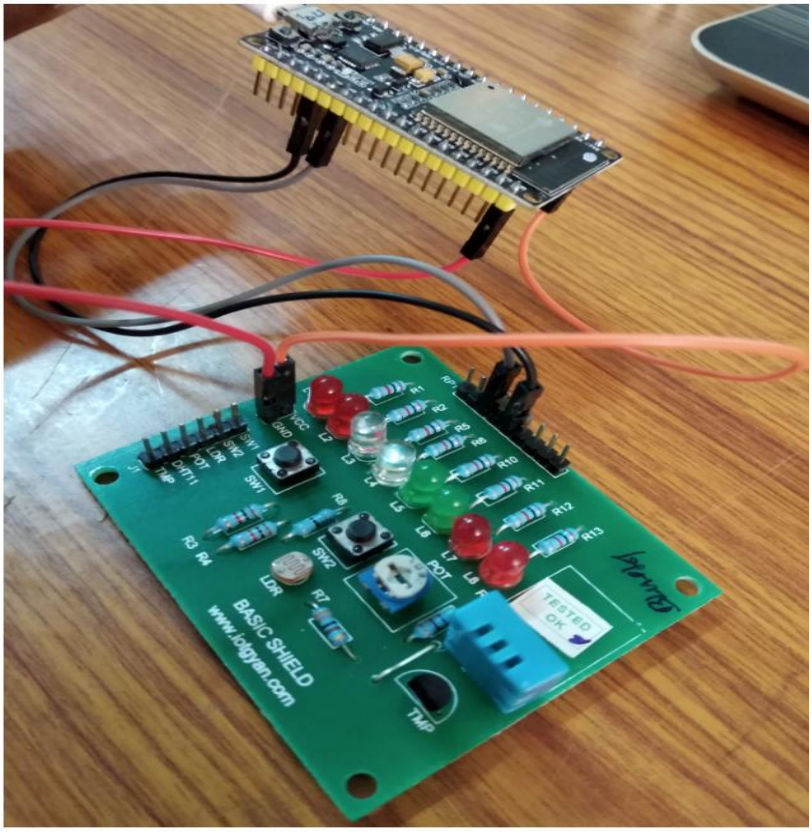


Hardware Recommended:

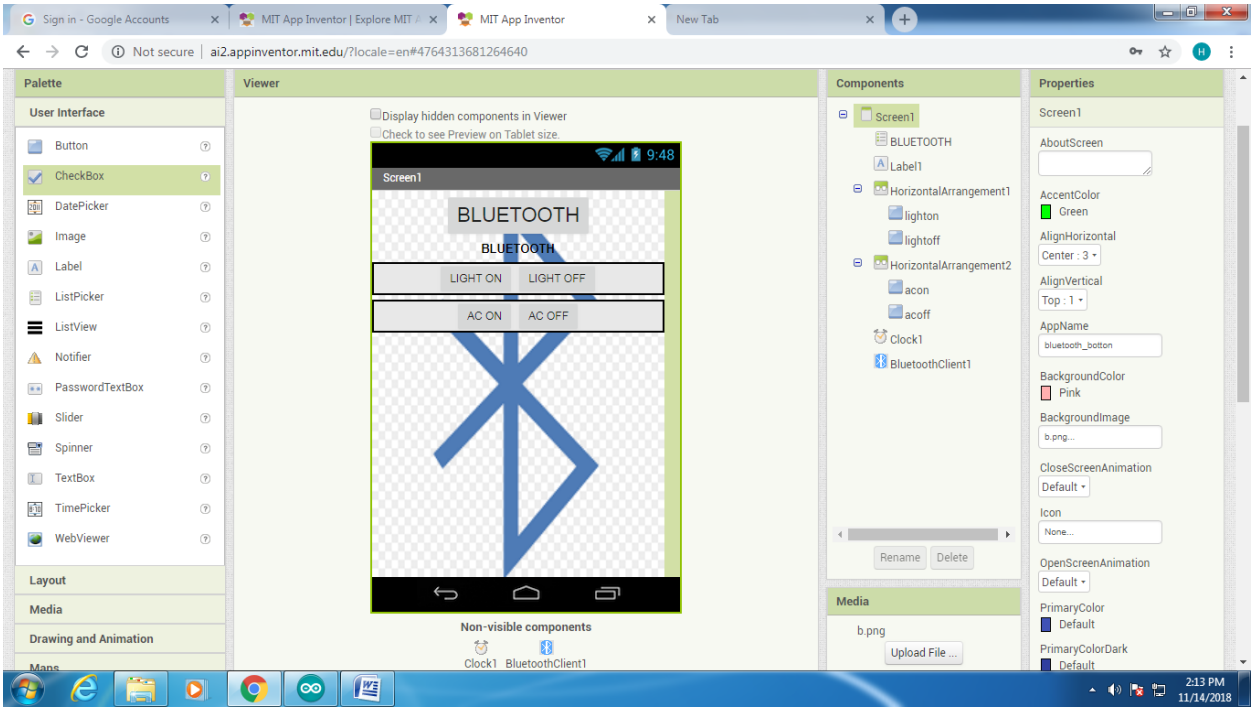
S.No.	Devices/Sensors	Qty, Nos.
1	ESP32 Development Kit	1
2	Basic Shield of SB	1
3	Connecting wires	10

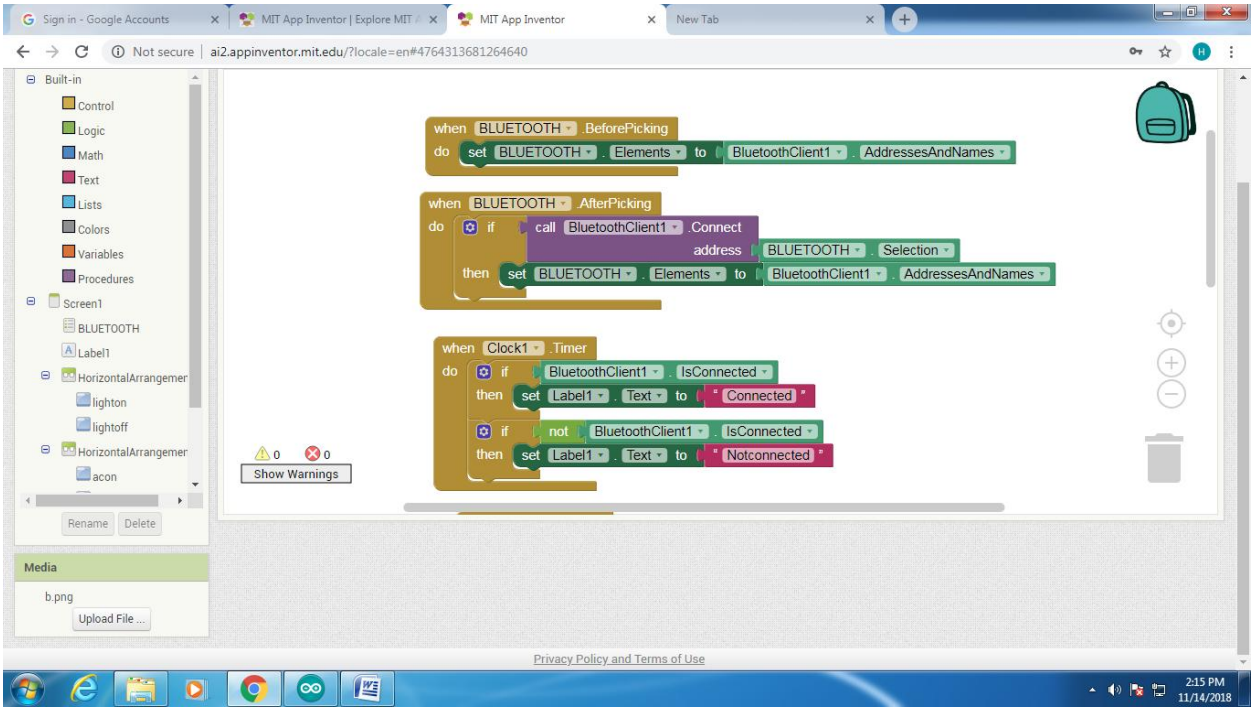
Block Diagram:

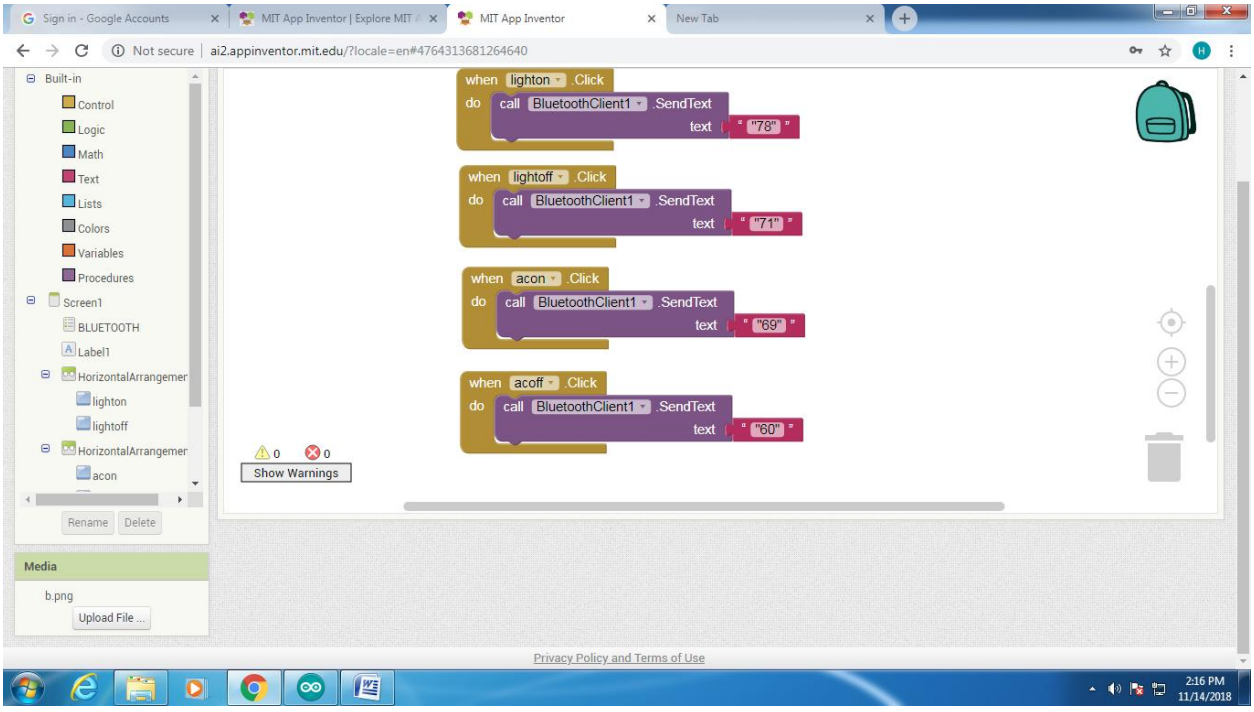




Bluetooth App design: (MIT APP INVENTOR TOOL)




Sign in - Google Accounts
MIT App Inventor | Explore MIT
MIT App Inventor
New Tab
Not secure | ai2.appinventor.mit.edu/?locale=en#4764313681264640
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen1
BLUETOOTH
Label1
HorizontalArrangemer
lighton
lightoff
HorizontalArrangemer
acon
Rename
Delete
Media
b.png
Upload File ...
when BLUETOOTH .BeforePicking
do set BLUETOOTH . Elements to BluetoothClient1 . AddressesAndNames
when BLUETOOTH .AfterPicking
do if call BluetoothClient1 .Connect
address BLUETOOTH . Selection
then set BLUETOOTH . Elements to BluetoothClient1 . AddressesAndNames
when Clock1 .Timer
do if BluetoothClient1 . IsConnected
then set Label1 . Text to " Connected "
if not BluetoothClient1 . IsConnected
then set Label1 . Text to " Notconnected "
0 0
Show Warnings
Privacy Policy and Terms of Use
2:15 PM
11/14/2018


Sign in - Google Accounts
MIT App Inventor | Explore MIT
MIT App Inventor
New Tab
Not secure | ai2.appinventor.mit.edu/?locale=en#4764313681264640
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen1
BLUETOOTH
Label1
HorizontalArrangemer
lighton
lightoff
HorizontalArrangemer
acon
Rename
Delete
Media
b.png
Upload File ...
when lighton .Click
do call BluetoothClient1 .SendText
text " "78" "
when lightoff .Click
do call BluetoothClient1 .SendText
text " "71" "
when acon .Click
do call BluetoothClient1 .SendText
text " "69" "
when acoff .Click
do call BluetoothClient1 .SendText
text " "60" "
0 0
Show Warnings
Privacy Policy and Terms of Use
2:16 PM
11/14/2018

Program Code:

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
BluetoothSerial SerialBT;
String state;
void setup() {
pinMode(2, OUTPUT);
pinMode(4, OUTPUT);
Serial.begin(115200);
SerialBT.begin("chandu"); //Bluetooth device name
Serial.println("The device started, now you can pair it with bluetooth!");
}

void loop() {
if (Serial.available())
{
SerialBT.write(Serial.read());
Serial.println("hello");
}
if (SerialBT.available()) {
state=SerialBT.read();
Serial.print("State :");
Serial.println(state);
if (state.equals("55")) {
  digitalWrite(2, HIGH);
  Serial.println("Light On");
}
// if the state is 'LED1OFF' the led1 will turn off
else  if (state.equals("49")){
digitalWrite(2, LOW);
Serial.println("Light Off");
}
// if the state is 'LED2ON' the Led2 will turn on
else  if (state.equals("57")){
digitalWrite(4, HIGH);
Serial.println("AC On");
}
```

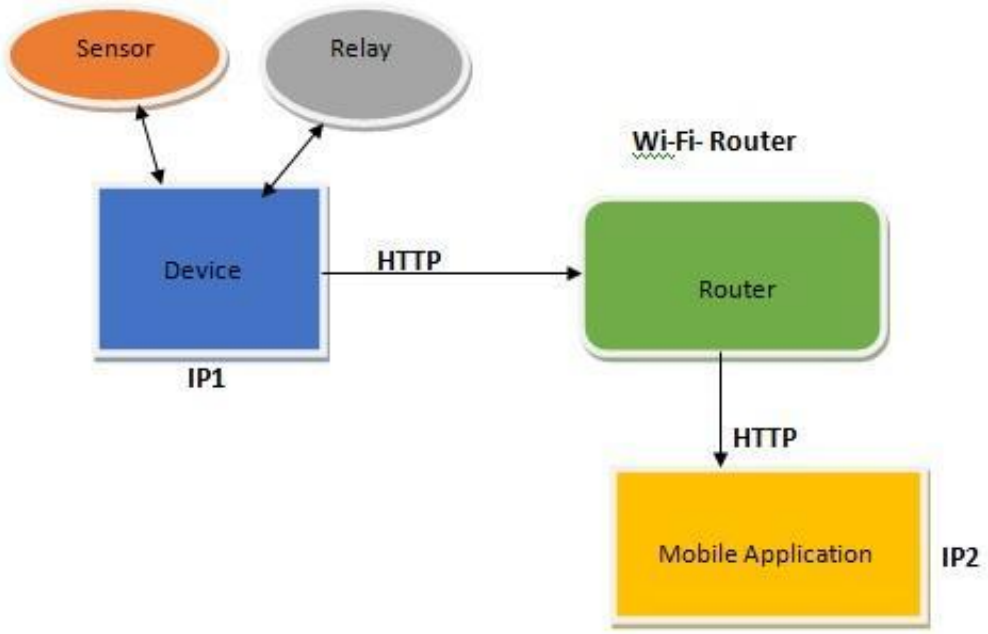
```
// if the state is 'LED2OFF' the led2 will turn off
else  if (state.equals("48")){
digitalWrite(4, LOW);
Serial.println("AC Off");
}
// if the state is 'ALLON' the Led's will turn on
else if (state.equals("53")) {
digitalWrite(2, HIGH);
digitalWrite(4, HIGH);
Serial.println("All On");
}
// if the state is 'ALLOFF' the Led's will turn off
else if (state.equals("54")) {
digitalWrite(2, LOW);
digitalWrite(4, LOW);
Serial.println("All Off");
}

}
state="";
delay(200);
}
```

Experiment -2: Application of WiFi in IoT Systems.

Aim: To study the various applications of Wireless Fidelity (Wi-Fi) in Internet of Things (IoT) by fetching the data from sensors and relay kit where such in a case by using HTTP protocol, the information is being transferred within the devices and the devices are operated automatically.

Objectives: Student should get the knowledge of Wi-Fi communication devices by Designing Mobile App.

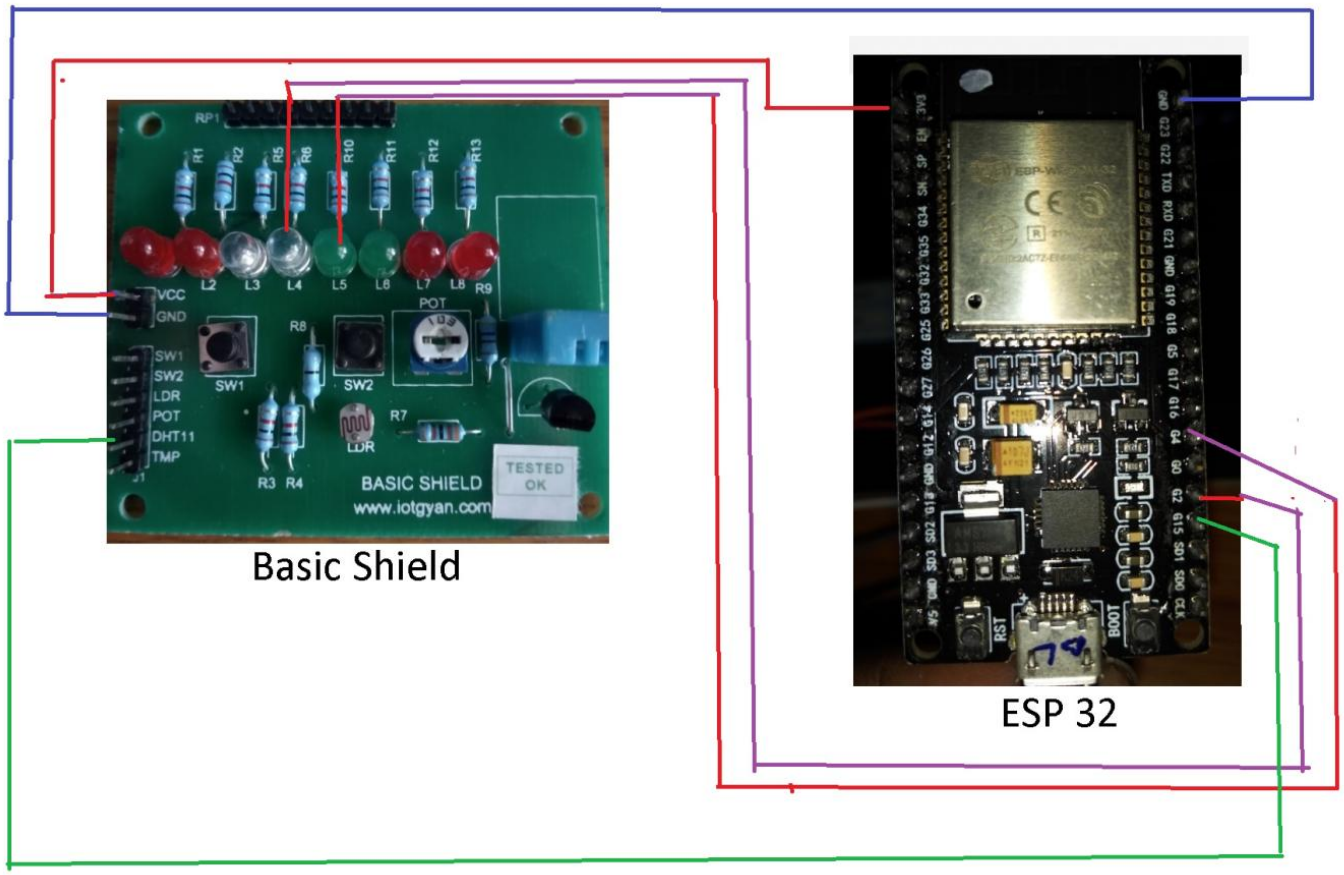
Outcomes: Student will be aware of communication using Wi-Fi background devices buy using Mobile App

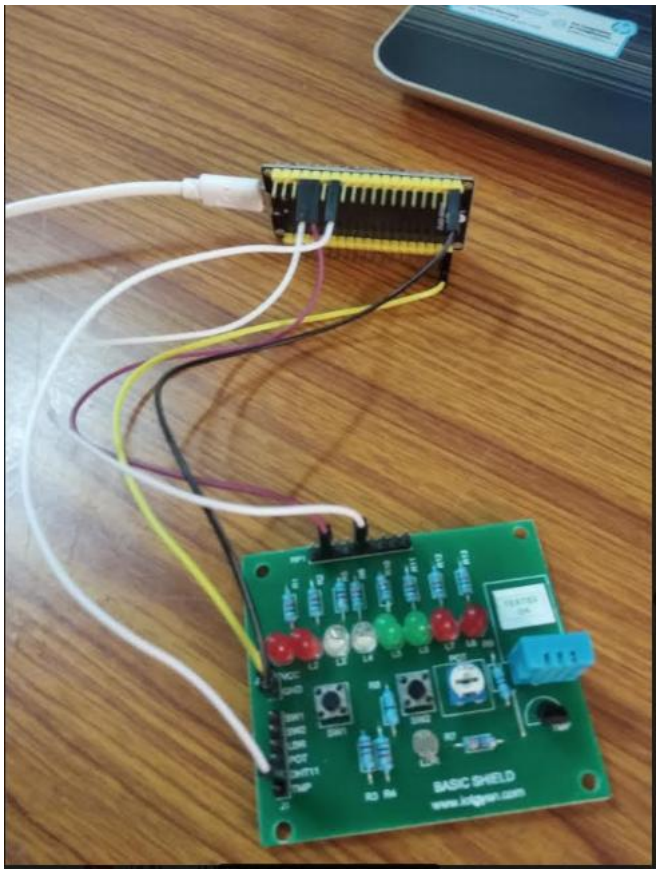


Hardware Recommended:

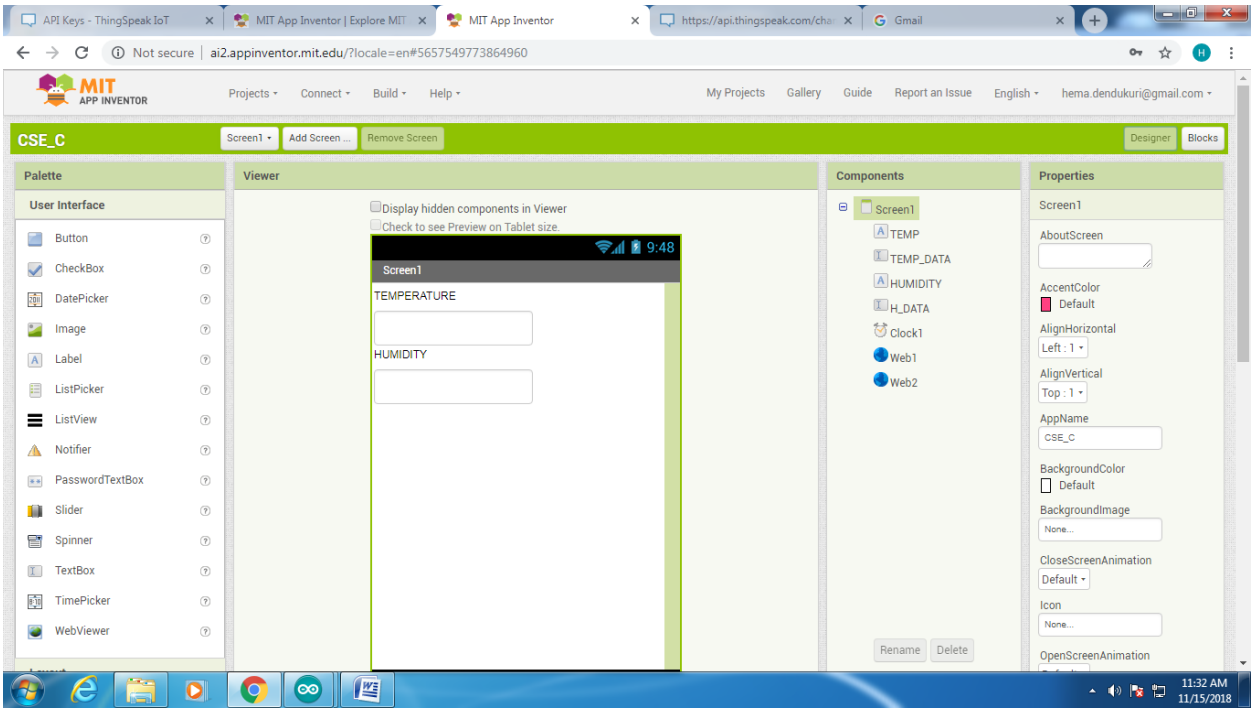
S.No.	Devices/Sensors	Qty, Nos.
1	ESP32 Hardware Development Board	1
2	Basic Shield of SB	1
3	Connecting wires	15
4	2 channel Relay Board	1

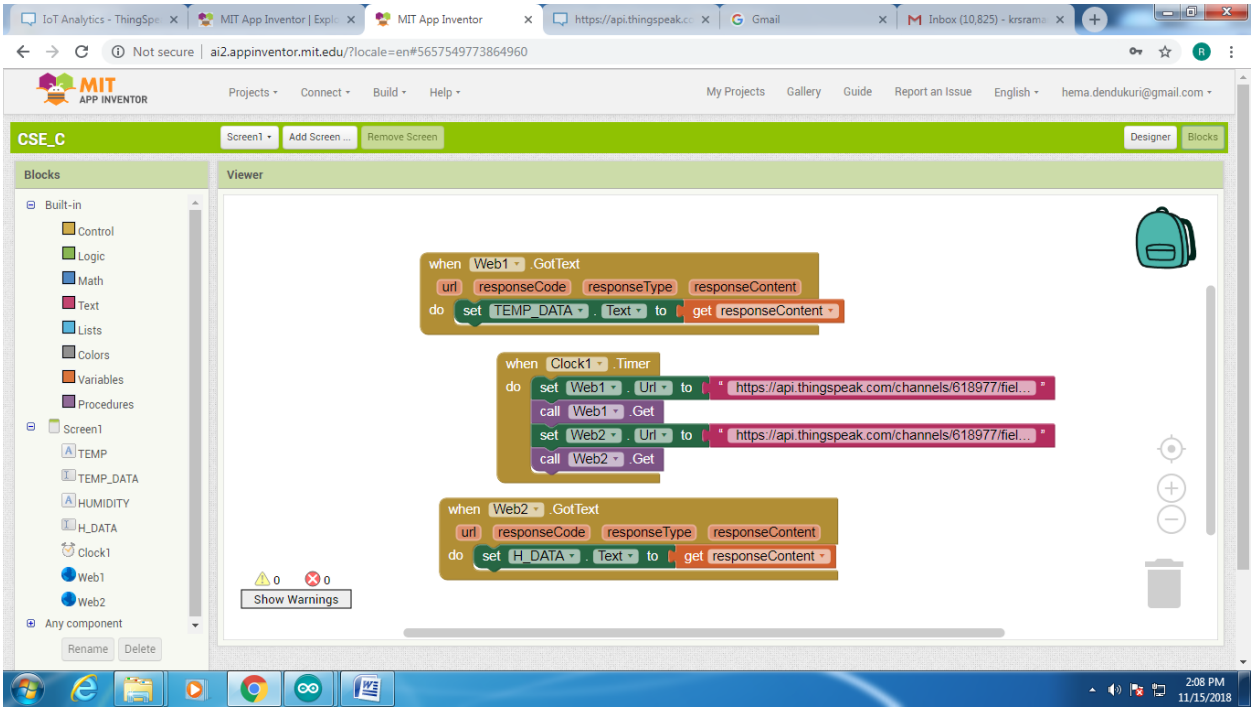
Block Diagram:





App designing (using MIT inventor Tool) : To display temperature and humidity



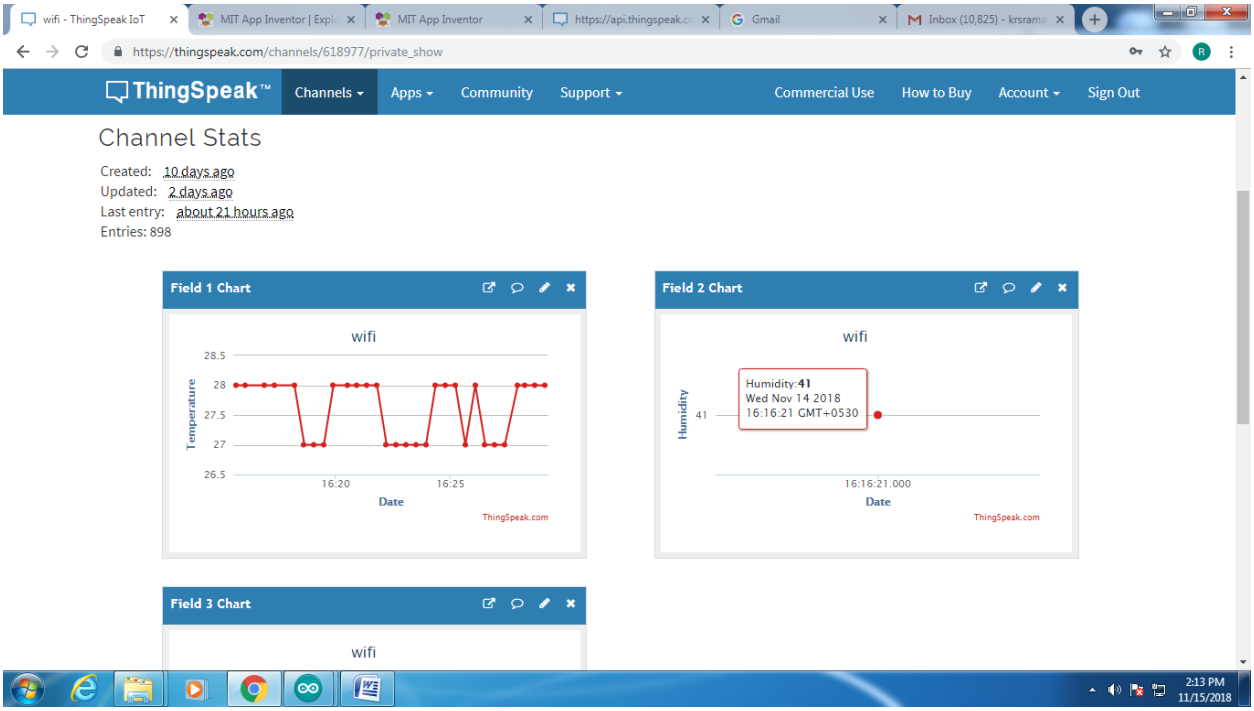


URL for Temperature field: **https://api.thingspeak.com/channels/618977/fields/1/last?result=0**

URL for Humidity field: **https://api.thingspeak.com/channels/618977/fields/2/last?result=0**

<u>Cloud space using ThingSpeak.com host:</u>

Channel Creation and Data view in ThingSpeak host



Program:

```
#include <WiFi.h>
#include "DHT.h"
#define DHTPIN 4     // what pin we're connected to
#define DHTTYPE DHT11   // define type of sensor DHT 11
DHT dht (DHTPIN, DHTTYPE);

const char* ssid     = "JioFi4_134EAC";//Enter the ssid of your router
const char* password = "cp19qexyh5";//Enter the password of your router

const char* host = "api.thingspeak.com";
const char* privateKey = "SYANE5W2LLUBYMGT";//read key
const char* privateKey1 = "TUEKQN0T47U4KA15";//write key
String line,line1;
float h,t;
#define light 2
#define fan 15

void setup() {
  Serial.begin(115200);
  pinMode(light, OUTPUT);//setting led as output
  pinMode(fan, OUTPUT);//setting led as output
  dht.begin();
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

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

```
void loop()
{
  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("temperature:");
  Serial.println(t);
  Serial.print("Humidity:");
  Serial.println(h);
  upload();
  delay(10000);
  retrieve_from_Cloud();
  delay(10000);
}

void upload()
{

   Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;

  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

   // We now create a URI for the request
  String url = "/update";
  url += "?api_key=";
  url += privateKey1;
  url += "&field1=";
  url += t;
  url += "&field2=";
  url += h;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
```

```
  delay(1000);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available())
  {
  String line1 = client.readStringUntil('\r');
  Serial.print(line1);
  }
  Serial.println();
  Serial.println("closing connection");
}

/*.....................................retrieving to Cloud.....................*/

void retrieve_from_Cloud()
{
  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url1 = "/channels/539141";
  url1 += "/fields/3/last?";
   url1 += "api_key=";
  url1 += privateKey;

  Serial.print("Requesting URL: ");
  Serial.println(url1);

  // This will send the request to the server
  client.print(String("GET ") + url1 + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(1000);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){

     if(client.find("@"))
```

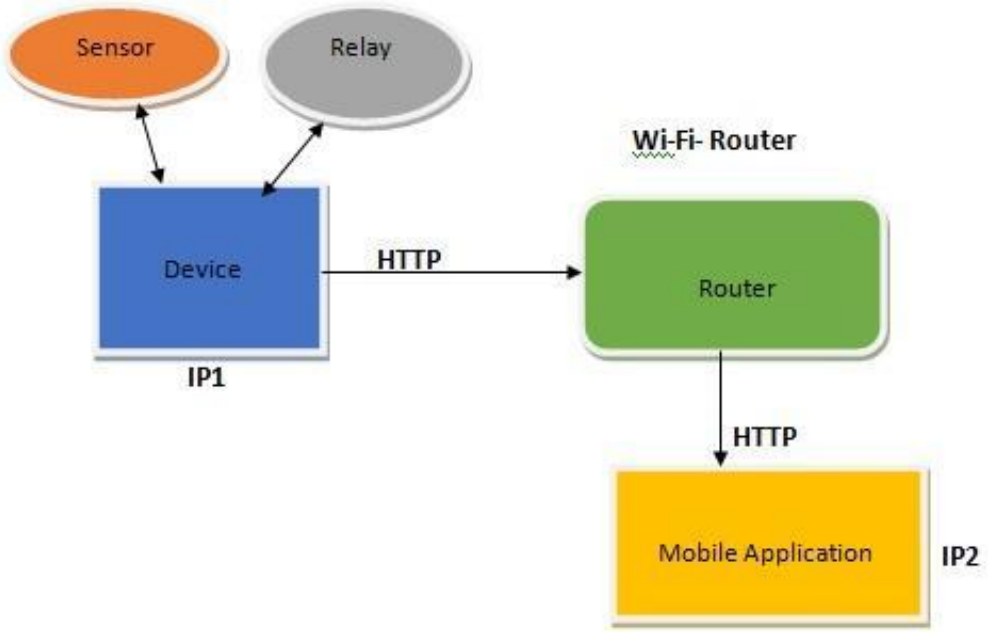
```
    {
      line = client.readString();
      Serial.print(line);

       if(line == "LIGHTON")
       {
       digitalWrite(light,HIGH);
       Serial.println(".......light on.........");
       }

       if(line == "LIGHTOFF")
       {
       digitalWrite(light,LOW);
       Serial.println(".......light off.........");
       }
       if(line == "FANON")
       {
       digitalWrite(fan,HIGH);
       Serial.println(".......fan on.........");
       }

       if(line == "FANOFF")
       {
       digitalWrite(fan,LOW);
       Serial.println(".......fan off.........");
       }
       line="";
     }
  }

  Serial.println();
  Serial.println("closing connection");
}
```

Experiment -3: App design for WiFi application to ON/OFF Light.

Aim: To study the various applications of Wireless Fidelity (Wi-Fi) in Internet of Things (IoT) by fetching the data from sensors and relay kit where such in a case by using HTTP protocol the information is being transferred within the devices and the devices are operated automatically.

Objectives: Student should get the knowledge of Wi-Fi communication devices by Designing Mobile App to ON or OFF light.

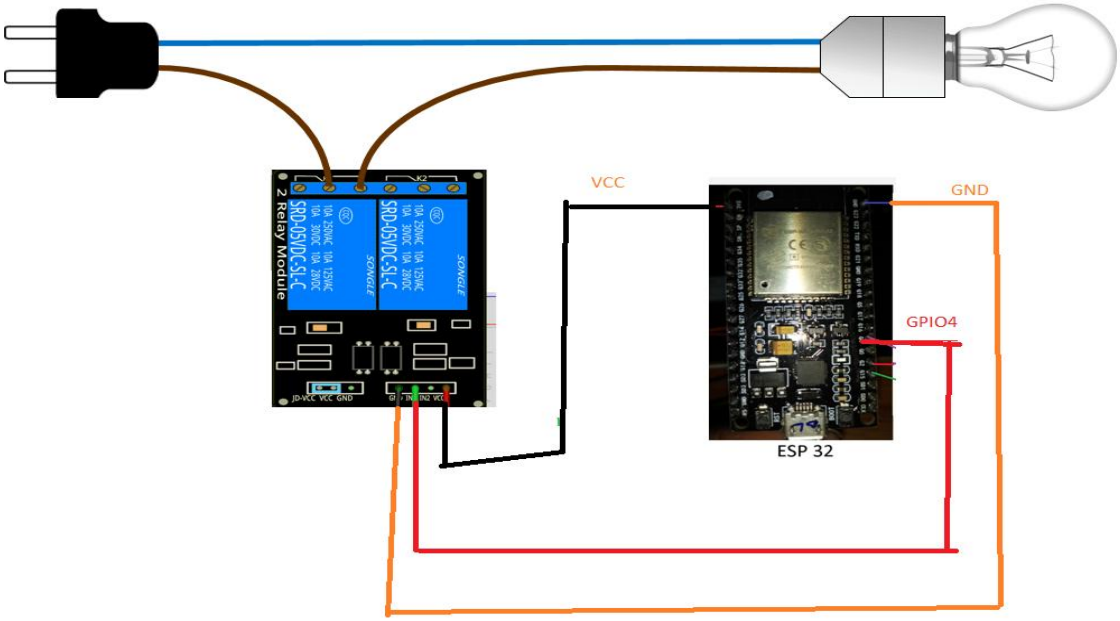
Outcomes: Student will be aware of communication using Wi-Fi background devices buy using Mobile App.

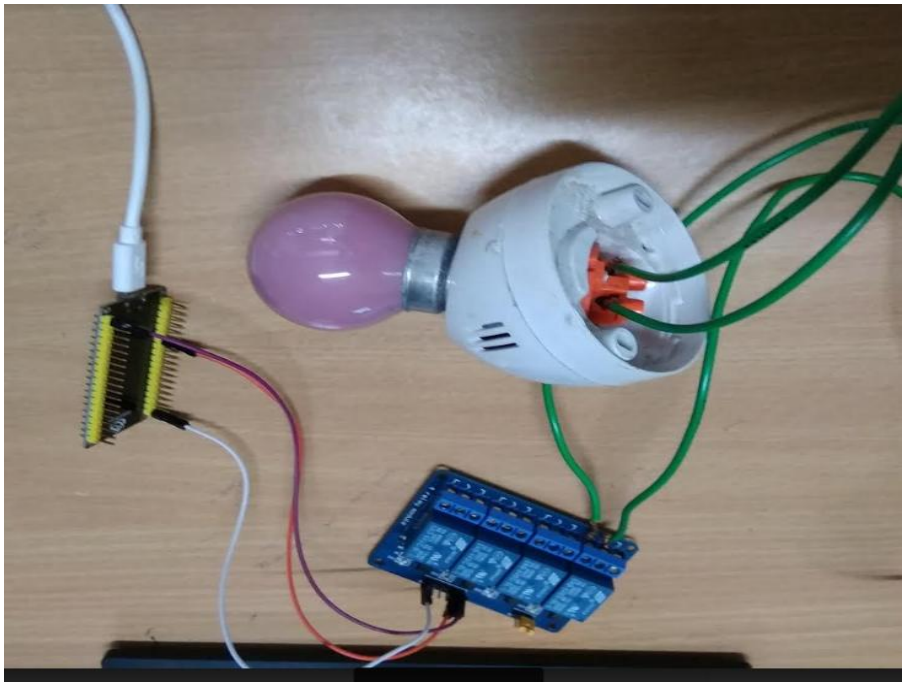


Hardware Recommended:

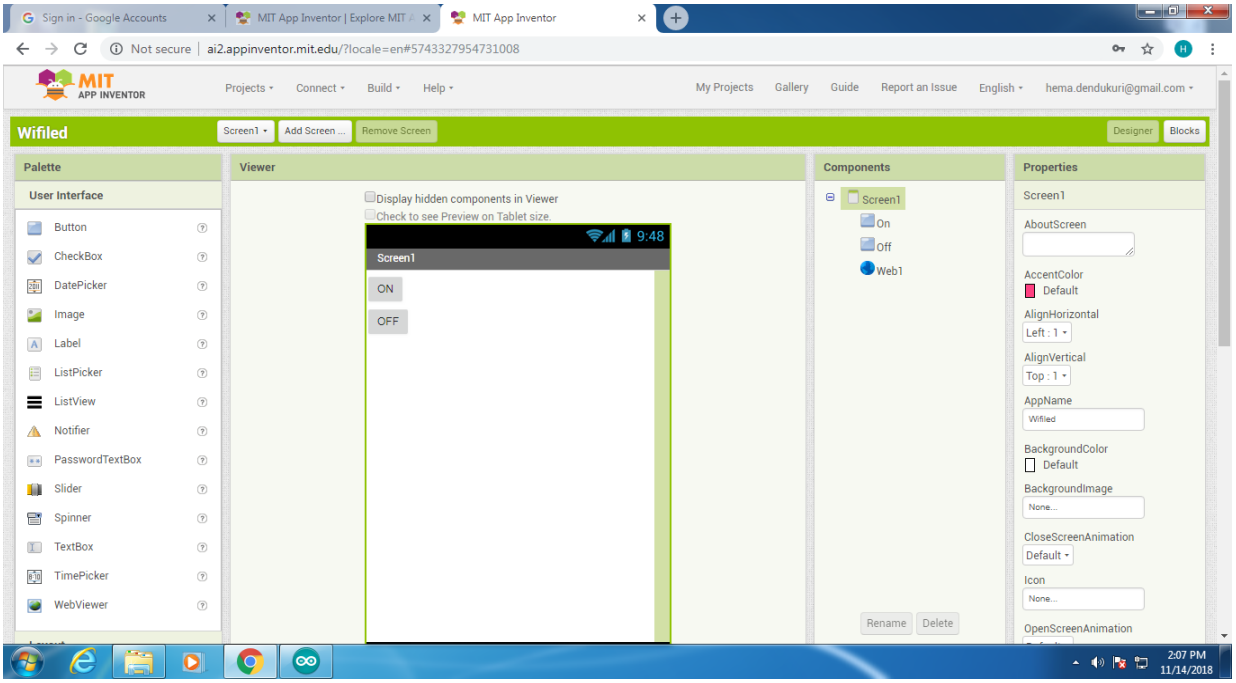
S.No.	Devices/Sensors	Qty, Nos.
1	ESP32 Hardware Development Board	1
2	Basic Shield of SB	1
3	Connecting wires	15
4	2 channel Relay Board	1

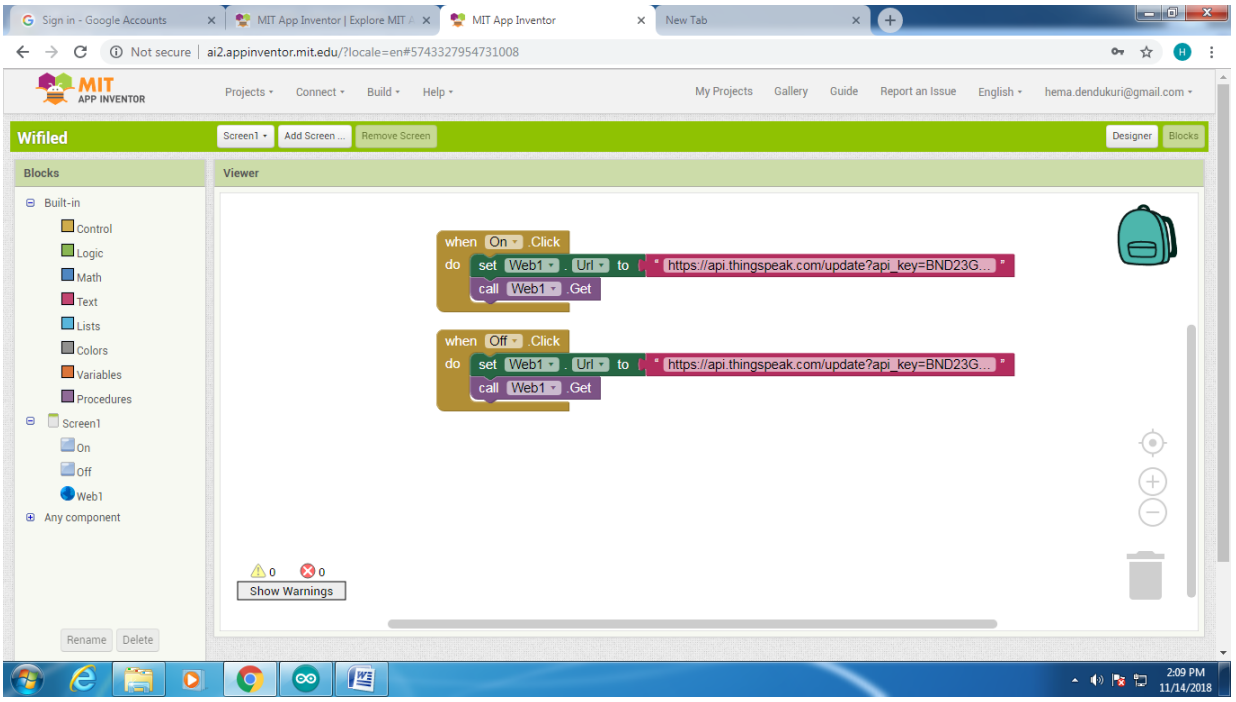
Block Diagram:





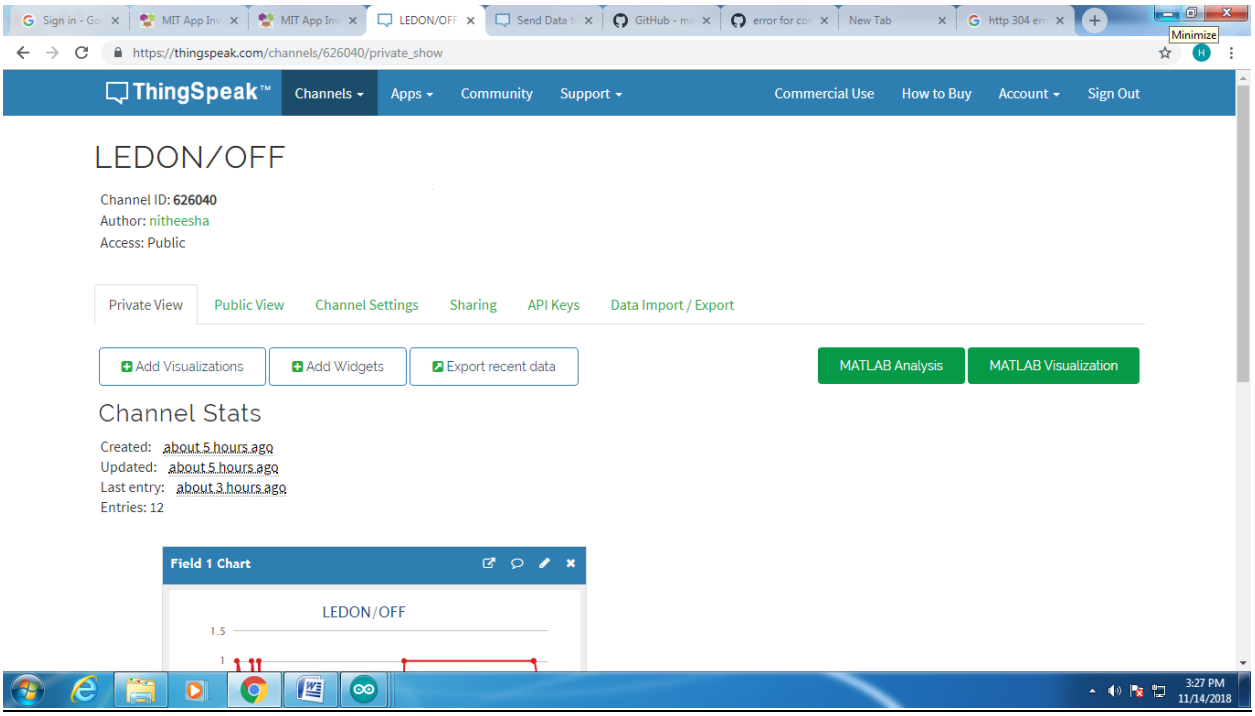
WiFi App design(using MIT App inventor Tool):

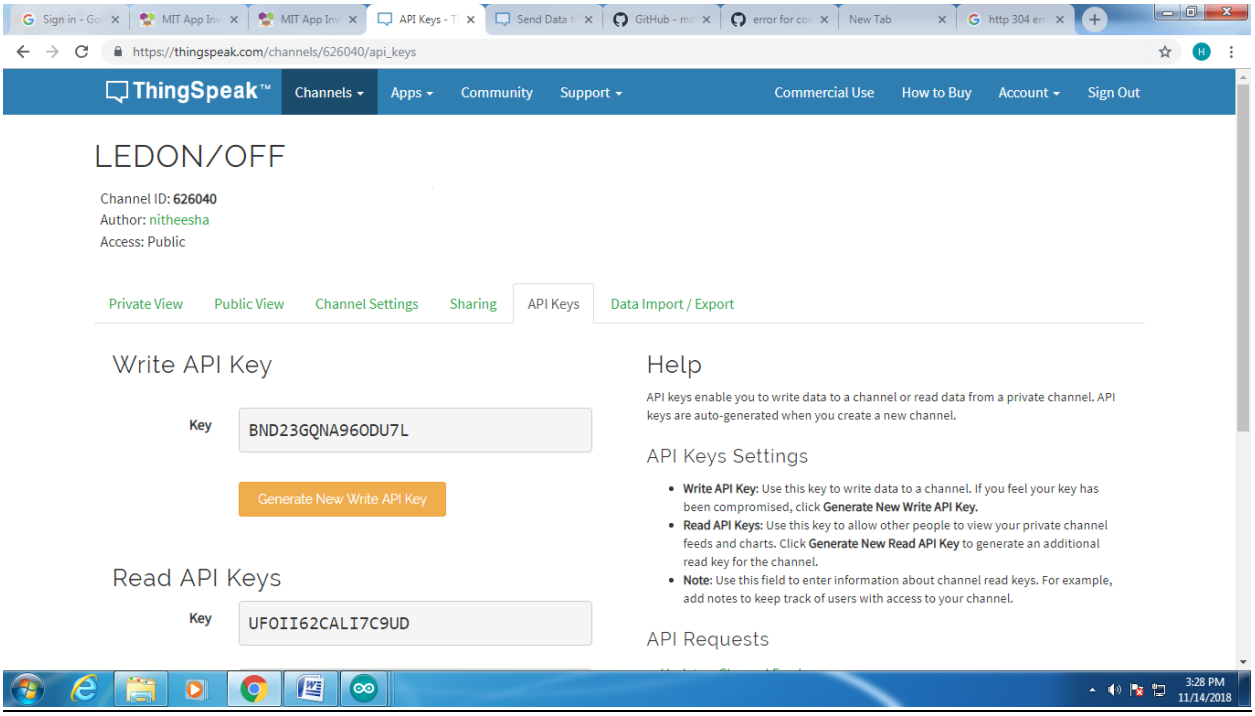




URL for ON button:
https://api.thingspeak.com/update?api_key=BND23GQNA96ODU7L&field1=1

URL for OFF button:
https://api.thingspeak.com/update?api_key=BND23GQNA96ODU7L&field1=0





Program Code:

```
#include <ThingSpeak.h>
#include <WiFi.h>
WiFiClient  client;
const char* ssid     = "ONLYCSE";//Enter the ssid of your router
const char* password = "keepsmile";//Enter the password of your router

const char* host = "api.thingspeak.com";
const char* privateKey = "UOZ3XFFHY6J345IO";//read key
const char* privateKey1 = "9HO22G0EY0D8XYD4";//write key

void setup() {
 Serial.begin(115200);
  pinMode(4, OUTPUT);//setting led as output
  //pinMode(fan, OUTPUT);//setting led as output
  //dht.begin();
  ThingSpeak.begin(client);
  delay(10);
```

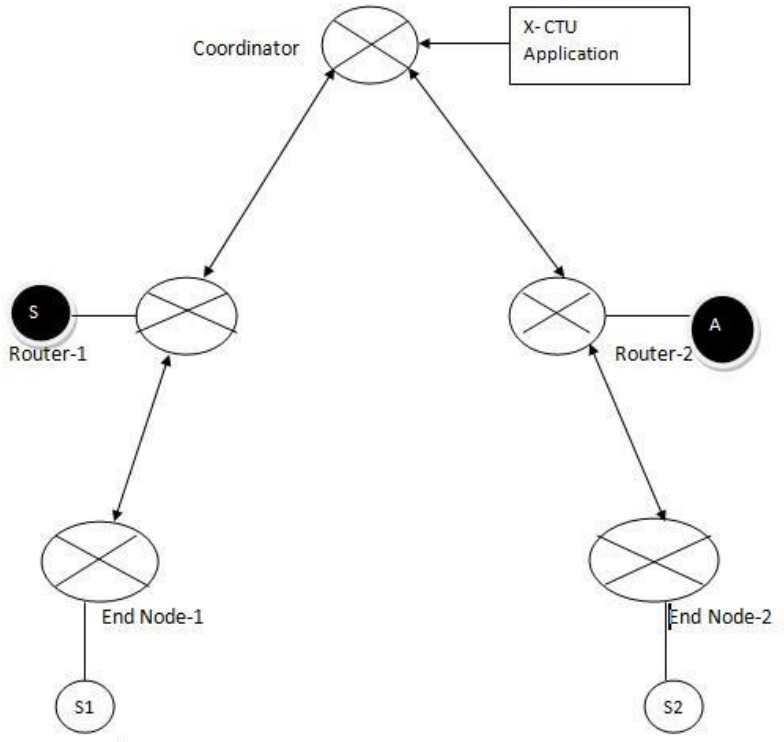
```
  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void loop() {

//float h= ThingSpeak.readFloatField( 626040,2);
int d= ThingSpeak.readIntField( 626082,1);
Serial.print(d);
if(d==1)
{
digitalWrite(4,HIGH);
Serial.print("LED ON");
Serial.println("");
}
if(d==0)

{digitalWrite(4,LOW);
Serial.print("LED OFF");
Serial.println("");
}

delay(1000);
}
```

Experiment -4: Application of 802.15.4 Zigbee in IoT Systems.

Aim: To study the applications of Zigbee in digital information transfer by creating a mesh networks by connecting the nodes to the server or router and establish a channel for information transfer. This is been operated by X-CTU application.

Objectives: Student should get the knowledge of Zigbee communication devices

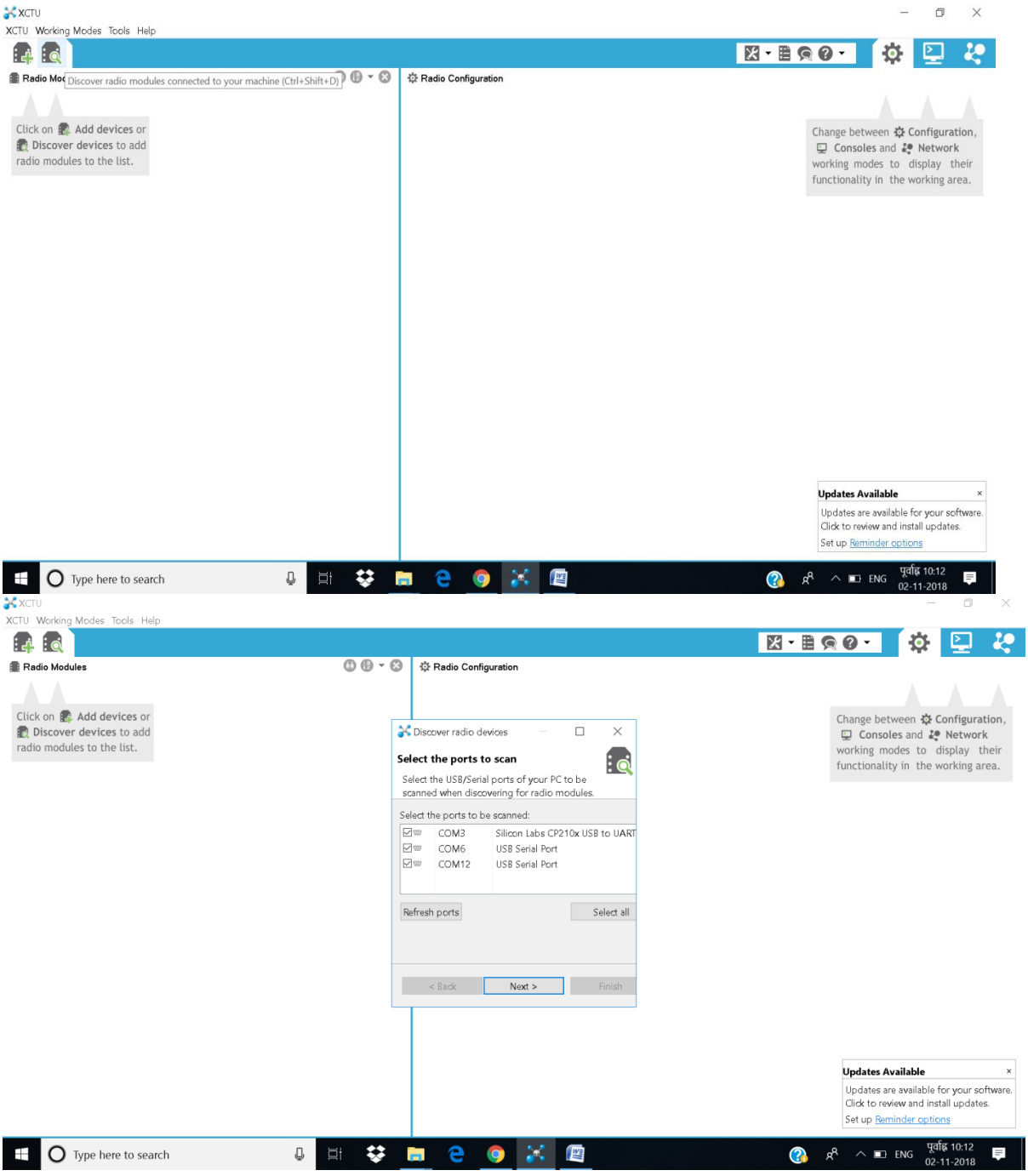
Outcomes: Student will be aware of communication using Zigbee background based devices

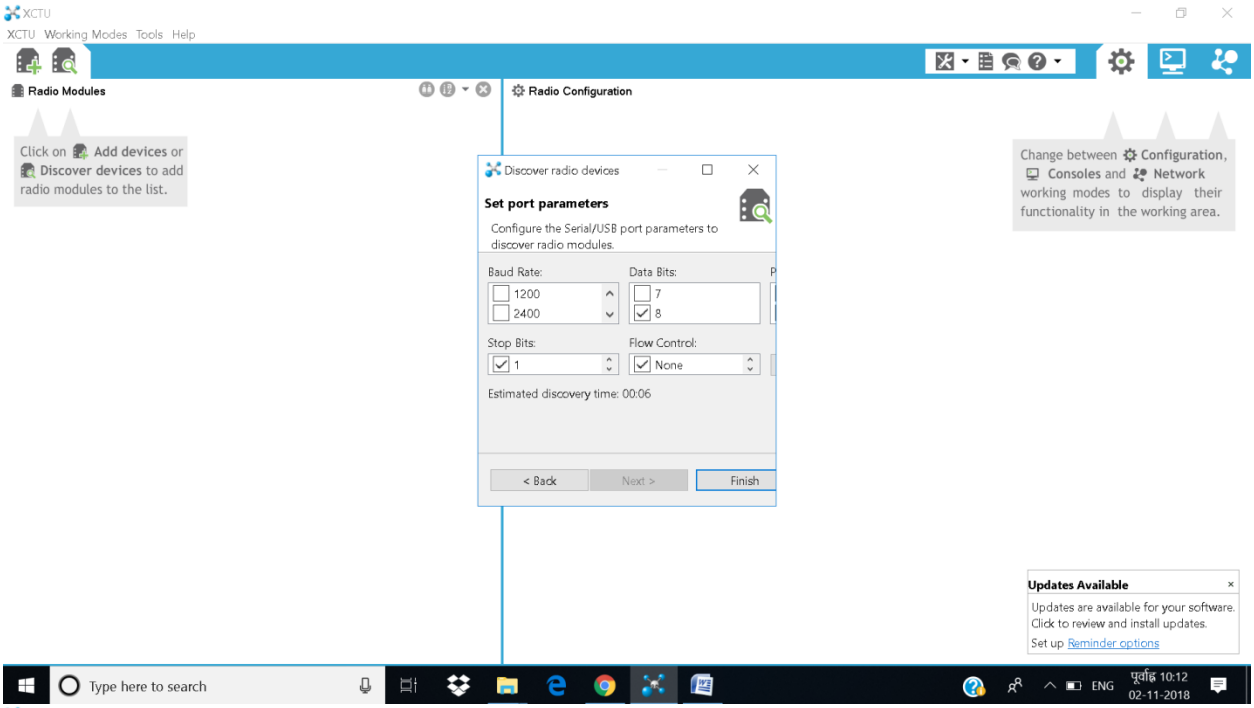


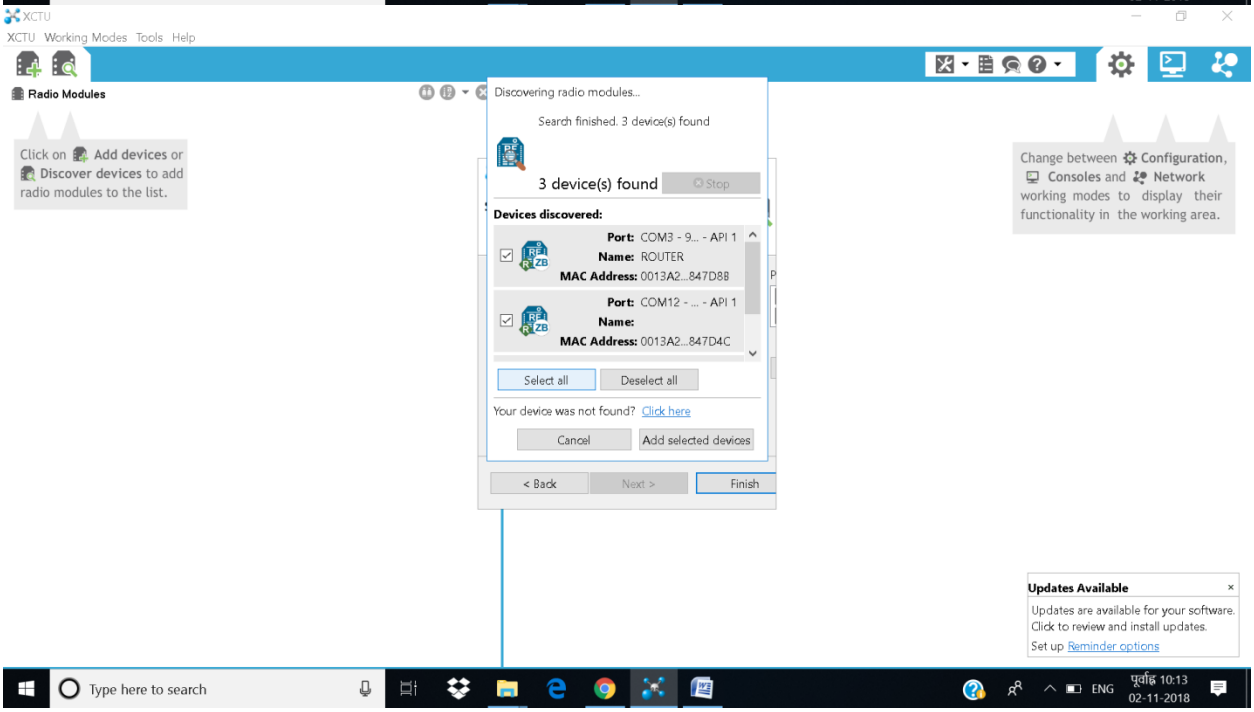
Hardware Recommended:

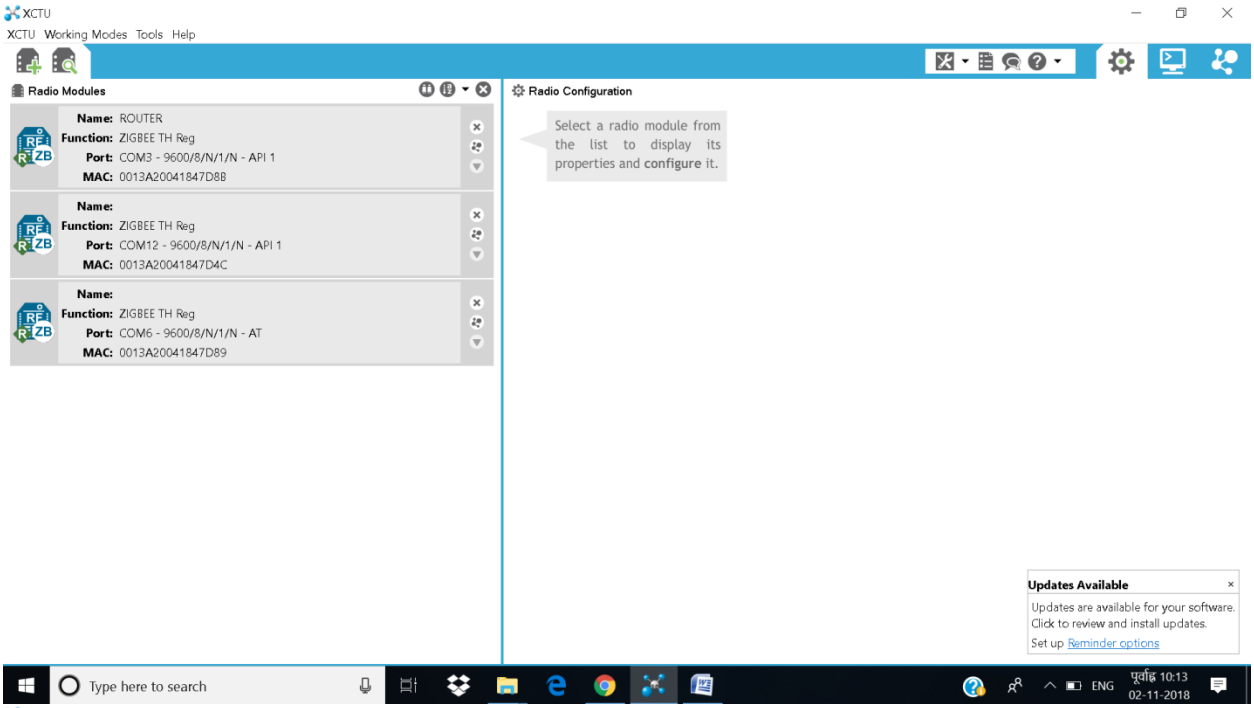
S.No.	Devices/Sensors	Qty, Nos.
1	ESP32 Hardware Development board	1
2	Zigbee Module + Base board	3
3	Basic Shield of SB	2
4	Connecting wires	30

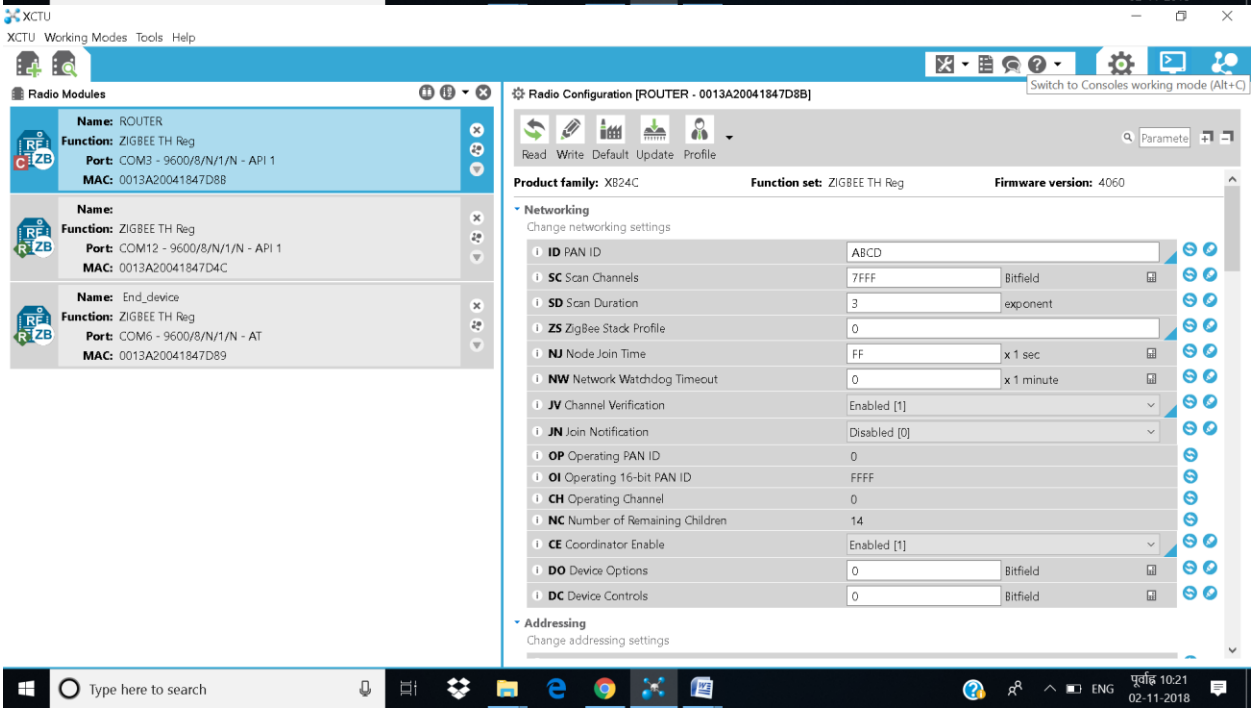
*Note: This experiment one network node only.

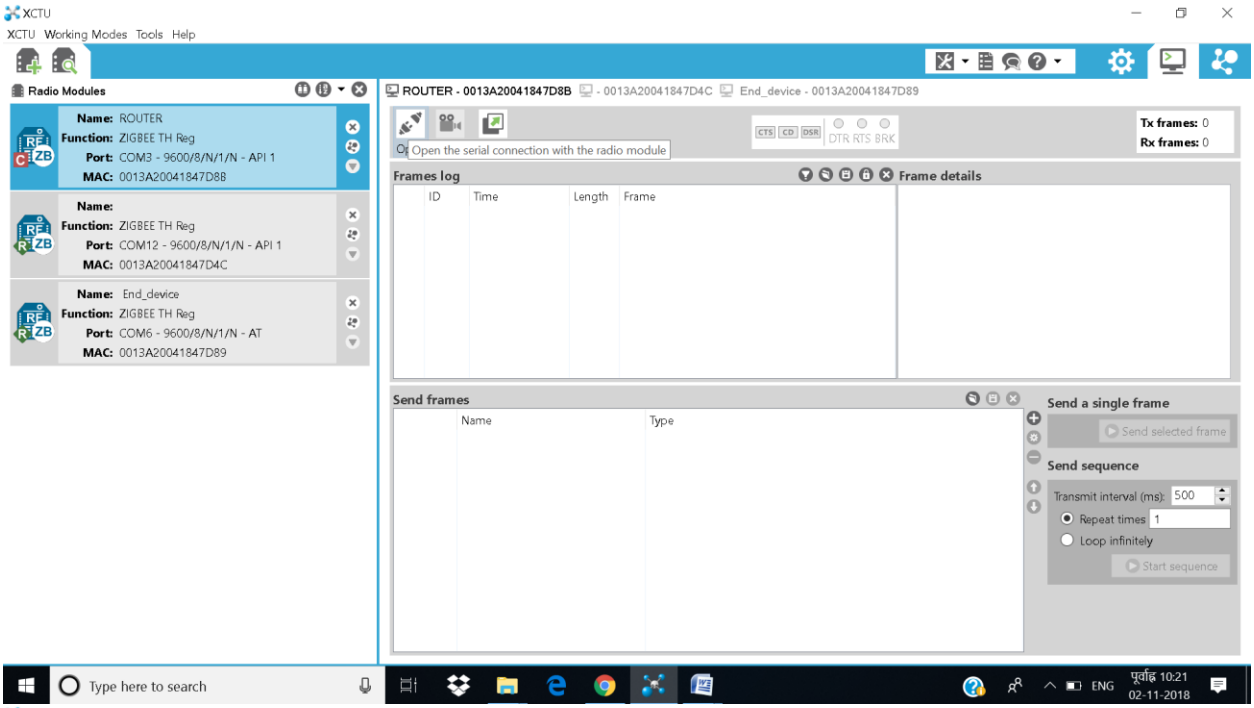

XCTU
XCTU Working Modes Tools Help
Radio Modules
Discover radio modules connected to your machine (Ctrl+Shift+D)
Radio Configuration
Click on Add devices or Discover devices to add radio modules to the list.
Change between Configuration, Consoles and Network working modes to display their functionality in the working area.
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Type here to search
ENG
पूर्वाह्न 10:12
02-11-2018
Discover radio devices
Select the ports to scan
Select the USB/Serial ports of your PC to be scanned when discovering for radio modules.
Select the ports to be scanned:
COM3 Silicon Labs CP210x USB to UART
COM6 USB Serial Port
COM12 USB Serial Port
Refresh ports
Select all
< Back
Next >
Finish

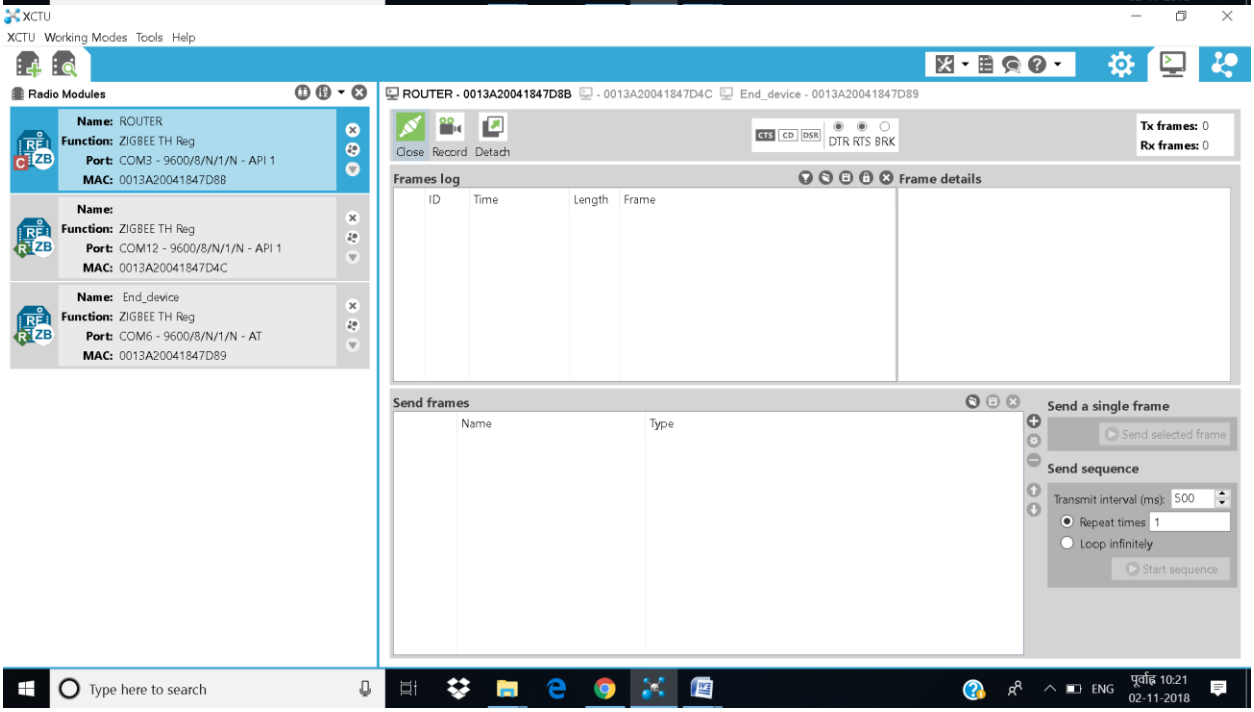

XCTU
XCTU Working Modes Tools Help
Radio Modules
Radio Configuration
Click on Add devices or Discover devices to add radio modules to the list.
Change between Configuration, Consoles and Network working modes to display their functionality in the working area.
Discover radio devices
Set port parameters
Configure the Serial/USB port parameters to discover radio modules.
Baud Rate:
1200
2400
Data Bits:
7
8
Stop Bits:
1
Flow Control:
None
Estimated discovery time: 00:06
< Back
Next >
Finish
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Type here to search
ENG
पूर्वाह्न 10:12
02-11-2018

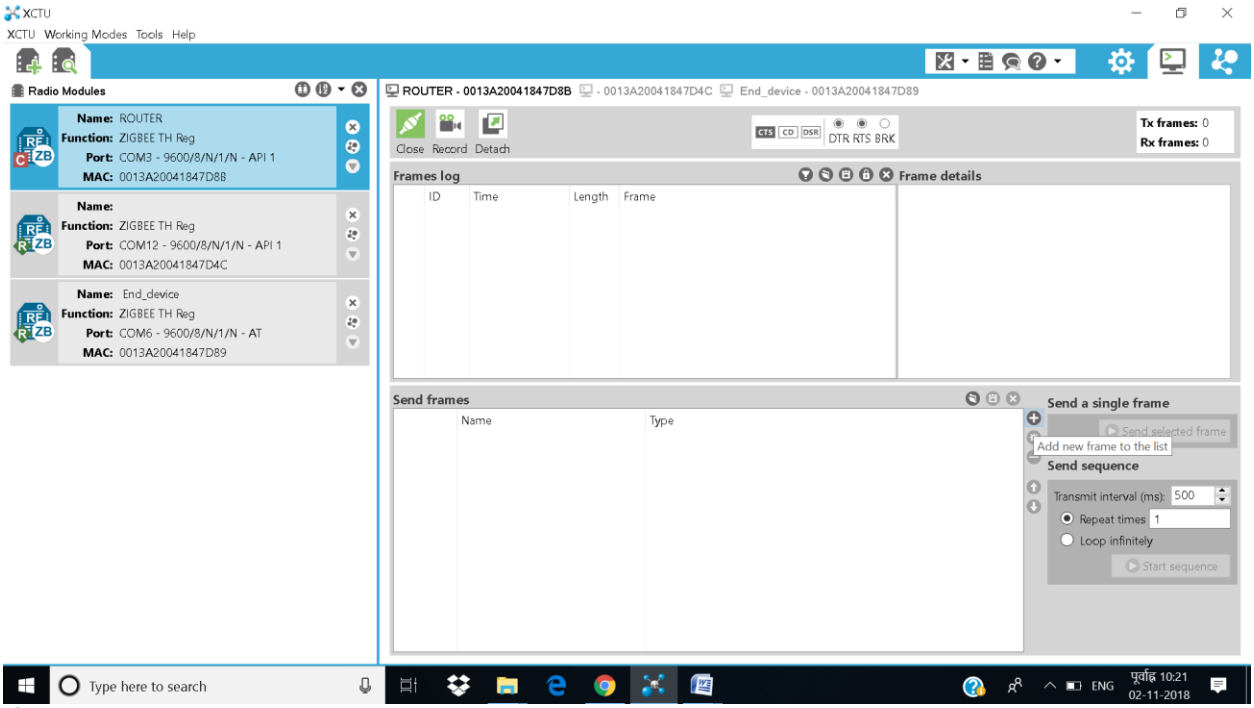

XCTU
XCTU Working Modes Tools Help
Radio Modules
Discovering radio modules...
Search finished. 3 device(s) found
3 device(s) found
Stop
Devices discovered:
Port: COM3 - 9... - API 1
Name: ROUTER
MAC Address: 0013A2...847D8B
Port: COM12 - ... - API 1
Name:
MAC Address: 0013A2...847D4C
Select all
Deselect all
Your device was not found? Click here
Cancel
Add selected devices
< Back
Next >
Finish
Click on Add devices or Discover devices to add radio modules to the list.
Change between Configuration, Consoles and Network working modes to display their functionality in the working area.
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Type here to search
ENG
पूर्वाह्न 10:13
02-11-2018

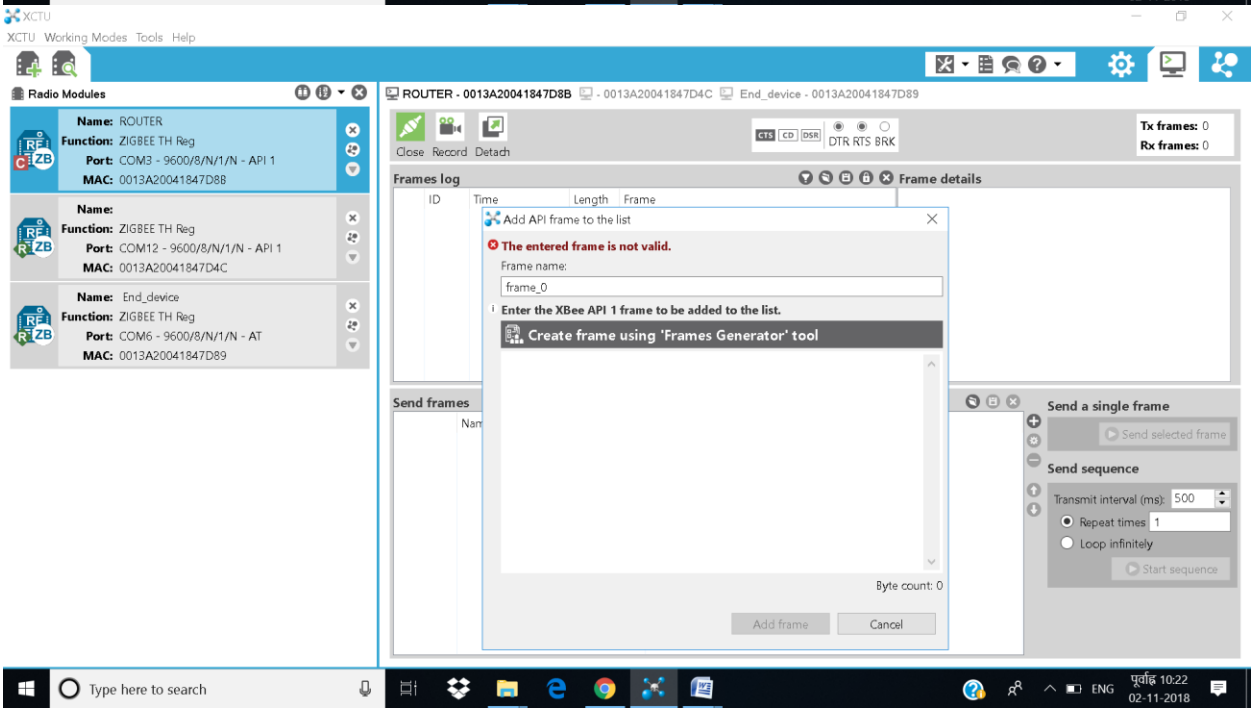

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name:
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
Radio Configuration
Select a radio module from the list to display its properties and configure it.
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Type here to search
ENG
पूर्वाह्न 10:13
02-11-2018

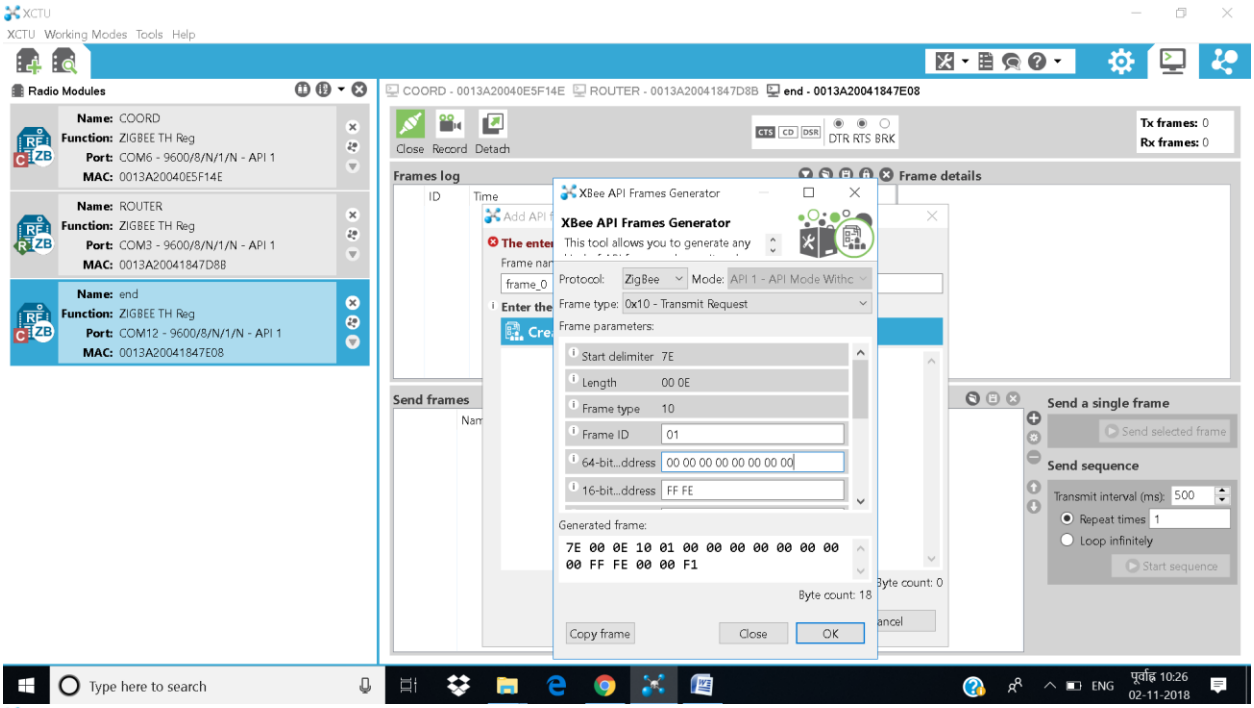

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: End_device
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
Radio Configuration [ROUTER - 0013A20041847D8B]
Switch to Consoles working mode (Alt+C)
Read
Write
Default
Update
Profile
Product family: XB24C
Function set: ZIGBEE TH Reg
Firmware version: 4060
Networking
Change networking settings
ID PAN ID ABCD
SC Scan Channels 7FFF Bitfield
SD Scan Duration 3 exponent
ZS ZigBee Stack Profile 0
NJ Node Join Time FF x 1 sec
NW Network Watchdog Timeout 0 x 1 minute
JV Channel Verification Enabled [1]
JN Join Notification Disabled [0]
OP Operating PAN ID 0
OI Operating 16-bit PAN ID FFFF
CH Operating Channel 0
NC Number of Remaining Children 14
CE Coordinator Enable Enabled [1]
DO Device Options 0 Bitfield
DC Device Controls 0 Bitfield
Addressing
Change addressing settings
Type here to search
ENG
पूर्वाह्न 10:21
02-11-2018

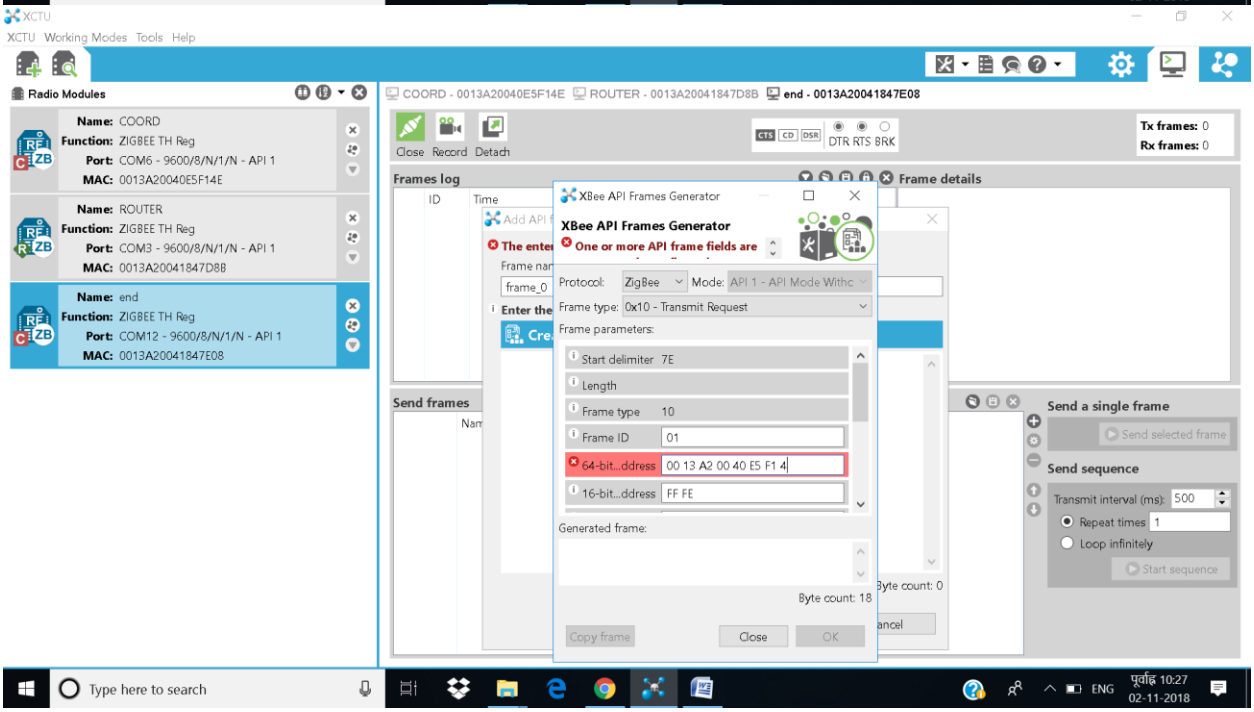

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D88
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: End_device
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
ROUTER - 0013A20041847D8B
0013A20041847D4C
End_device - 0013A20041847D89
Open the serial connection with the radio module
CTS CD DSR
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Send frames
Name
Type
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:21
02-11-2018

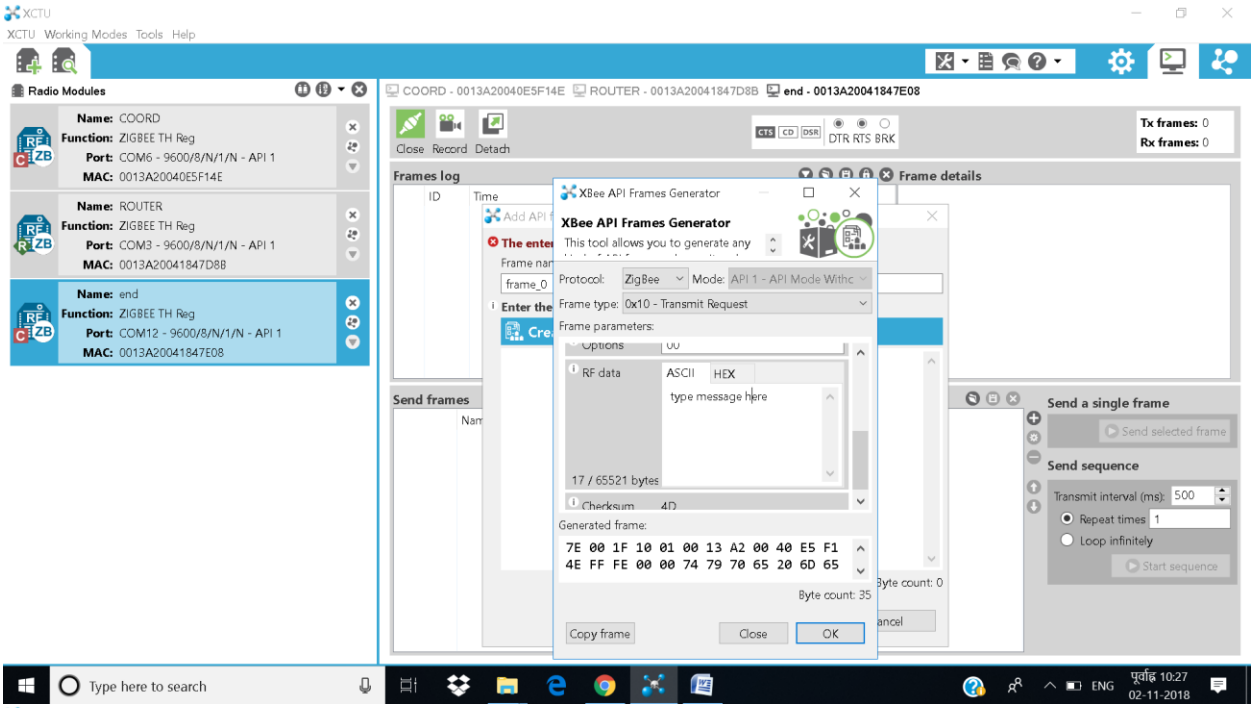

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D88
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: End_device
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
ROUTER - 0013A20041847D8B
0013A20041847D4C
End_device - 0013A20041847D89
Close
Record
Detach
CTS CD DSR
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Send frames
Name
Type
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:21
02-11-2018

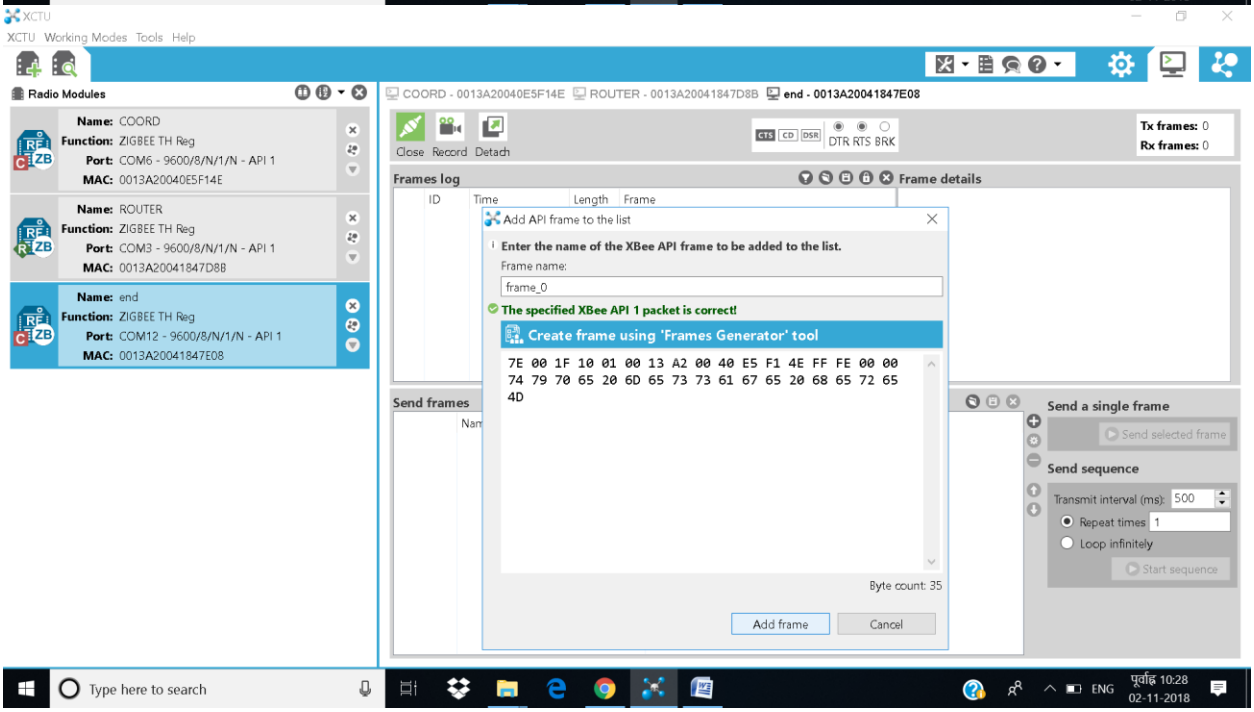

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D88
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: End_device
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
ROUTER - 0013A20041847D8B
0013A20041847D4C
End_device - 0013A20041847D89
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Send frames
Name
Type
Send a single frame
Send selected frame
Add new frame to the list
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:21
02-11-2018

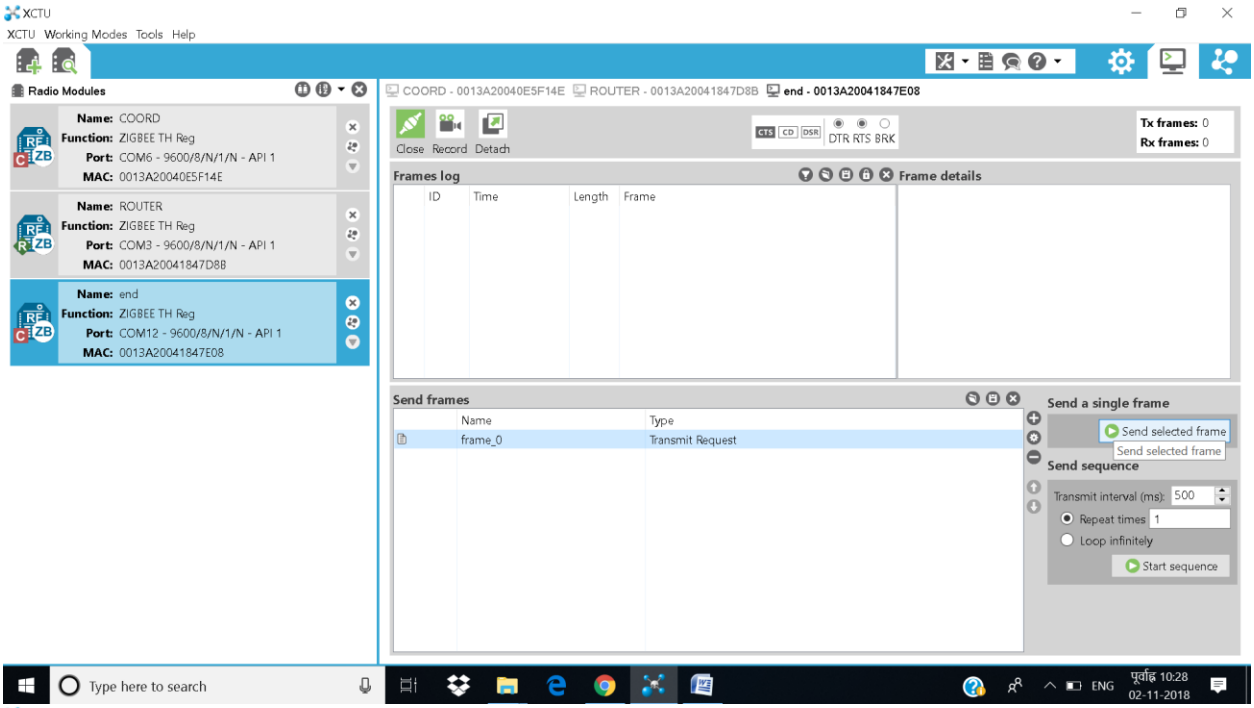

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D88
Name:
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: End_device
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - AT
MAC: 0013A20041847D89
ROUTER - 0013A20041847D8B
0013A20041847D4C
End_device - 0013A20041847D89
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Add API frame to the list
The entered frame is not valid.
Frame name:
frame_0
Enter the XBee API 1 frame to be added to the list.
Create frame using 'Frames Generator' tool
Byte count: 0
Add frame
Cancel
Send frames
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:22
02-11-2018

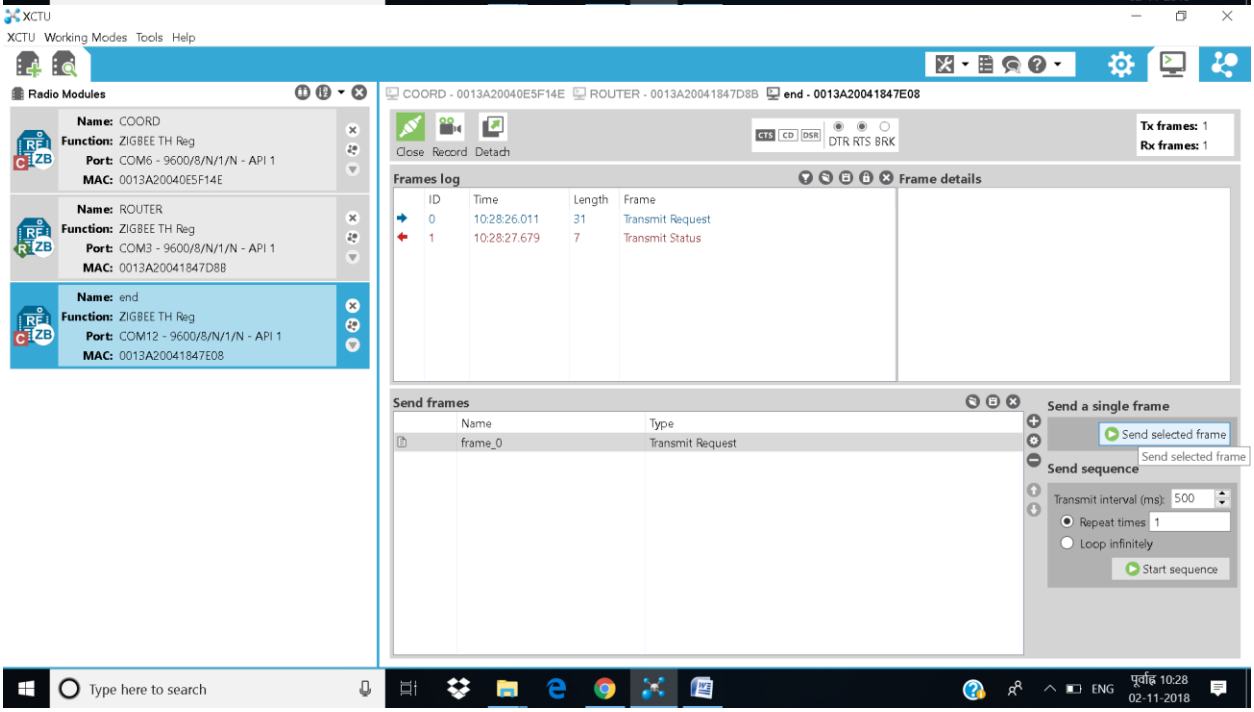

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
XBee API Frames Generator
XBee API Frames Generator
This tool allows you to generate any
Protocol: ZigBee
Mode: API 1 - API Mode Withc
Frame type: 0x10 - Transmit Request
Frame parameters:
Start delimiter 7E
Length 00 0E
Frame type 10
Frame ID 01
64-bit...ddress 00 00 00 00 00 00 00 00
16-bit...ddress FF FE
Generated frame:
7E 00 0E 10 01 00 00 00 00 00 00 00 00 FF FE 00 00 F1
Byte count: 18
Copy frame
Close
OK
Send frames
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Byte count: 0
Type here to search
ENG
पूर्वाह्न 10:26
02-11-2018

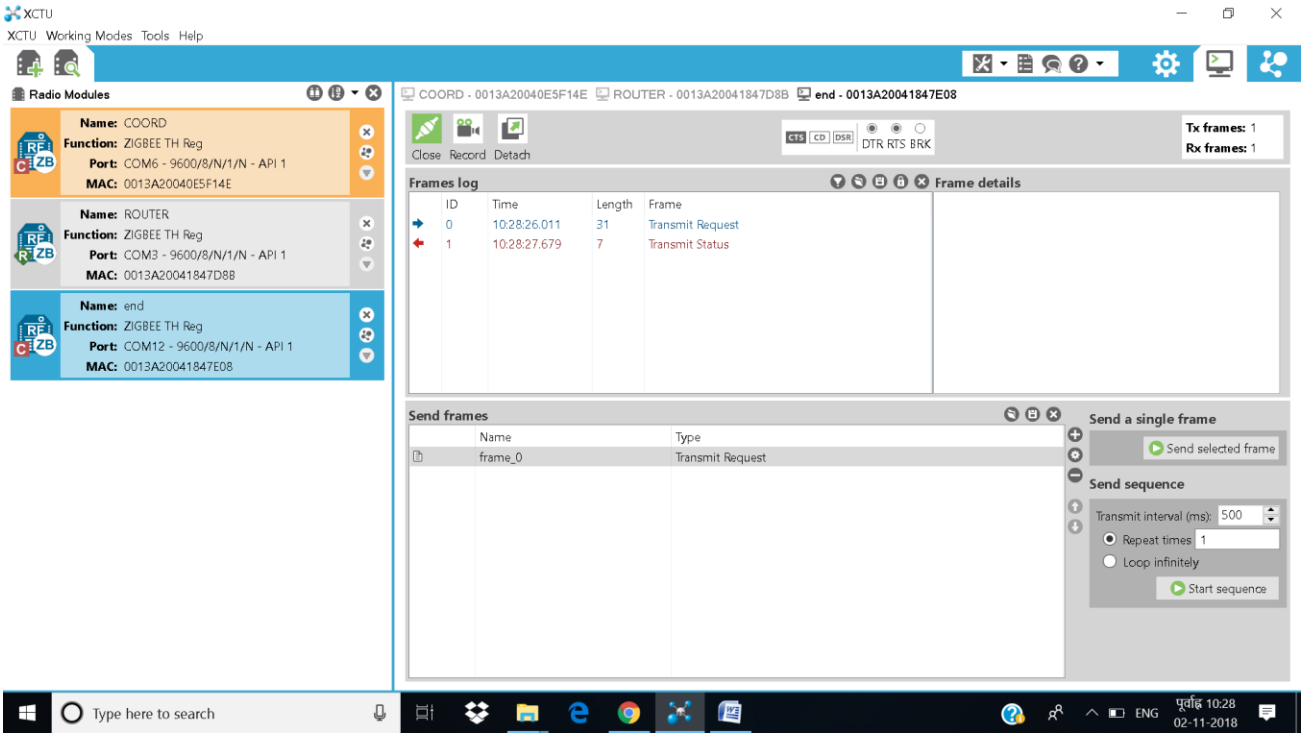

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
XBee API Frames Generator
XBee API Frames Generator
One or more API frame fields are
Protocol: ZigBee
Mode: API 1 - API Mode Withc
Frame type: 0x10 - Transmit Request
Frame parameters:
Start delimiter 7E
Length
Frame type 10
Frame ID 01
64-bit...ddress 00 13 A2 00 40 E5 F1 4
16-bit...ddress FF FE
Generated frame:
Byte count: 18
Copy frame
Close
OK
Send frames
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Byte count: 0
Type here to search
ENG
पूर्वाह्न 10:27
02-11-2018

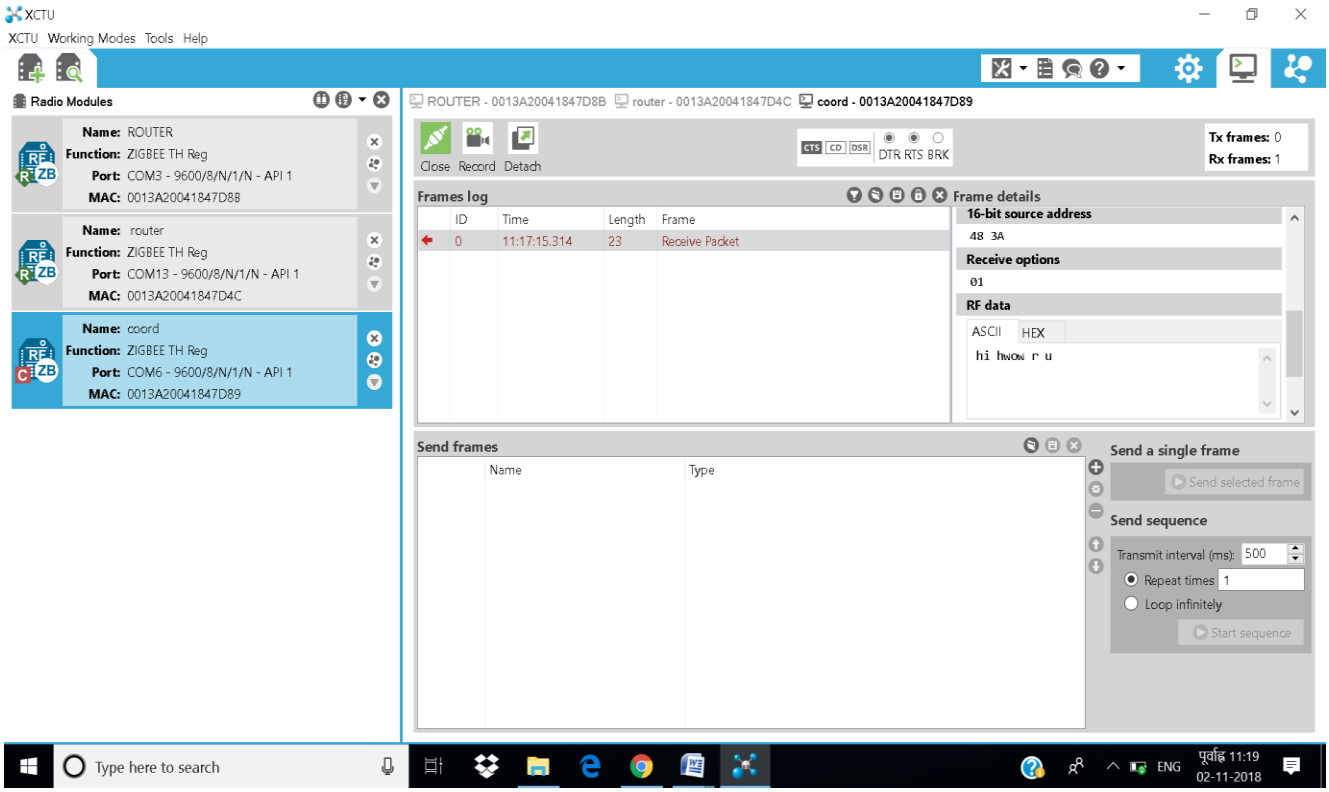

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
XBee API Frames Generator
XBee API Frames Generator
This tool allows you to generate any
Protocol: ZigBee
Mode: API 1 - API Mode Withc
Frame type: 0x10 - Transmit Request
Frame parameters:
RF data
ASCII
HEX
type message here
17 / 65521 bytes
Checksum 4D
Generated frame:
7E 00 1F 10 01 00 13 A2 00 40 E5 F1
4E FF FE 00 00 74 79 70 65 20 6D 65
Byte count: 35
Copy frame
Close
OK
Send frames
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:27
02-11-2018


XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Add API frame to the list
Enter the name of the XBee API frame to be added to the list.
Frame name:
frame_0
The specified XBee API 1 packet is correct!
Create frame using 'Frames Generator' tool
7E 00 1F 10 01 00 13 A2 00 40 E5 F1 4E FF FE 00 00
74 79 70 65 20 6D 65 73 73 61 67 65 20 68 65 72 65
4D
Byte count: 35
Add frame
Cancel
Send frames
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:28
02-11-2018


XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 0
Frames log
Frame details
ID
Time
Length
Frame
Send frames
Name
Type
frame_0
Transmit Request
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:28
02-11-2018


XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 1
Rx frames: 1
Frames log
Frame details
ID
Time
Length
Frame
0
10:28:26.011
31
Transmit Request
1
10:28:27.679
7
Transmit Status
Send frames
Name
Type
frame_0
Transmit Request
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:28
02-11-2018


XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: COORD
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20040E5F14E
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: end
Function: ZIGBEE TH Reg
Port: COM12 - 9600/8/N/1/N - API 1
MAC: 0013A20041847E08
COORD - 0013A20040E5F14E
ROUTER - 0013A20041847D8B
end - 0013A20041847E08
Close
Record
Detach
DTR RTS BRK
Tx frames: 1
Rx frames: 1
Frames log
Frame details
ID
Time
Length
Frame
0
10:28:26.011
31
Transmit Request
1
10:28:27.679
7
Transmit Status
Send frames
Name
Type
frame_0
Transmit Request
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 10:28
02-11-2018


XCTU
XCTU Working Modes Tools Help
Radio Modules
Name: ROUTER
Function: ZIGBEE TH Reg
Port: COM3 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D8B
Name: router
Function: ZIGBEE TH Reg
Port: COM13 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D4C
Name: coord
Function: ZIGBEE TH Reg
Port: COM6 - 9600/8/N/1/N - API 1
MAC: 0013A20041847D89
ROUTER - 0013A20041847D8B
router - 0013A20041847D4C
coord - 0013A20041847D89
Close
Record
Detach
DTR RTS BRK
Tx frames: 0
Rx frames: 1
Frames log
Frame details
ID
Time
Length
Frame
0
11:17:15.314
23
Receive Packet
16-bit source address
48 3A
Receive options
01
RF data
ASCII
HEX
hi hwOw r u
Send frames
Name
Type
Send a single frame
Send selected frame
Send sequence
Transmit interval (ms): 500
Repeat times 1
Loop infinitely
Start sequence
Type here to search
ENG
पूर्वाह्न 11:19
02-11-2018

Experiment -5: Design a simple IoT System comprising sensor, Wireless Network connection, Data Analytics

Aim: To study the various applications of Wireless Fidelity (Wi-Fi) in Internet of Things (IoT) by fetching the data from sensors and relay kit where such in a case by using HTTP protocol, the information is being transferred within the devices and the devices are operated automatically

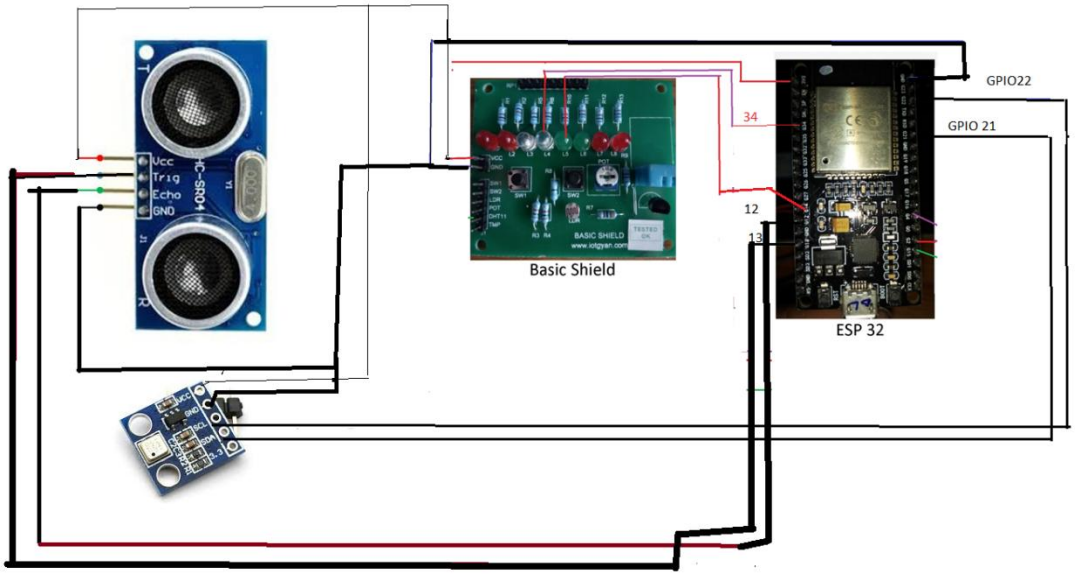
Objectives: Student should get the knowledge of comprising sensor, Wireless network connection and Data analytics.

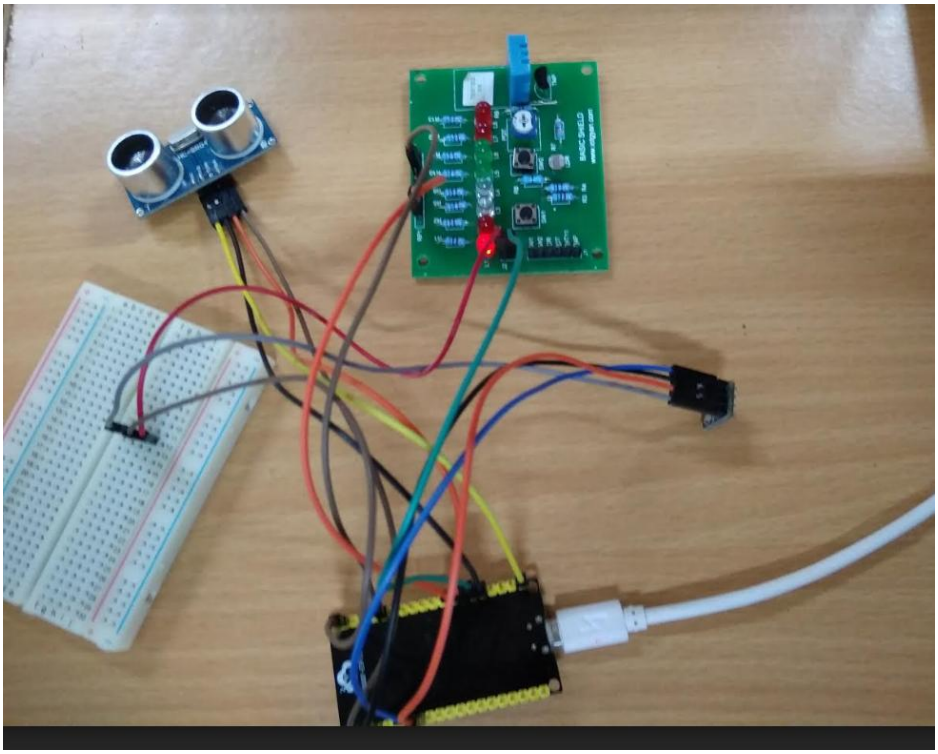
Outcomes: Student will be aware of comprising sensor, wireless network connection and Data analytics.

Hardware Recommended:

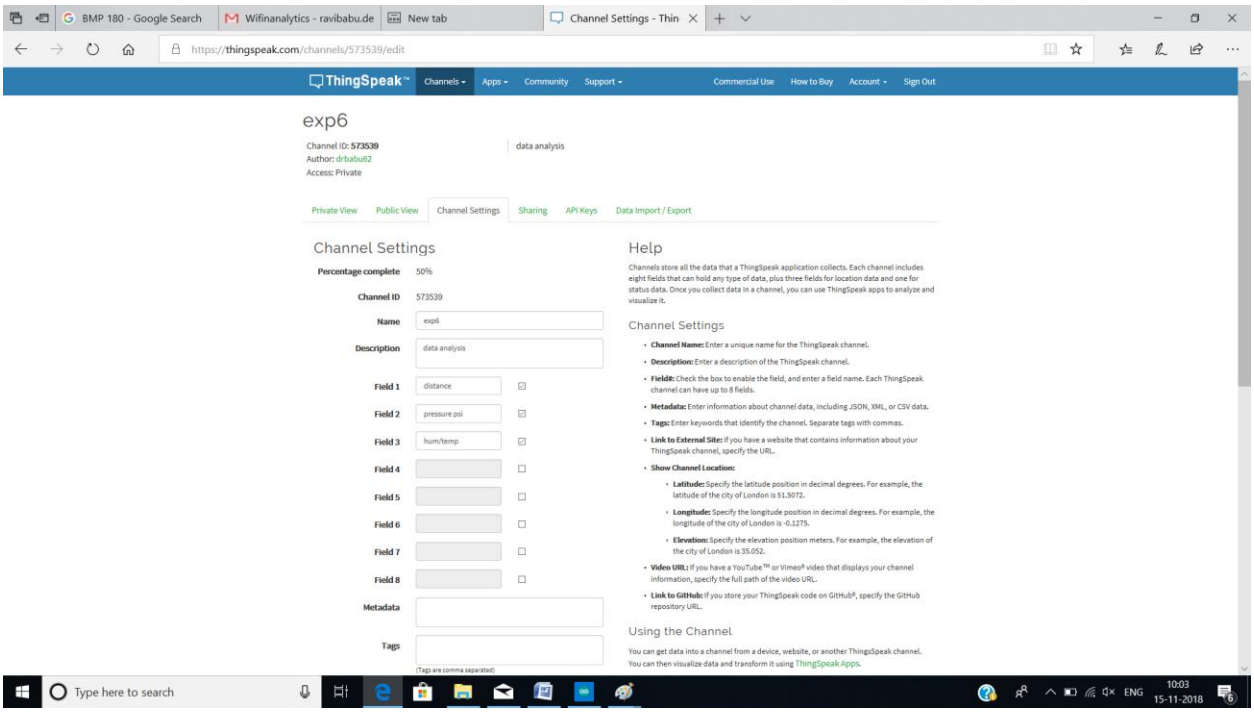
S.No.	Devices/Sensors	Qty, Nos.
1	Esp32 Hardware Development Board	2
2	Raspberry PI	1
3	Basic Shield of SB	3
4	Connecting wires	30
5	2 channel Relay Board	1
6	Ultrasonic sensor	1
7	BMP180	1

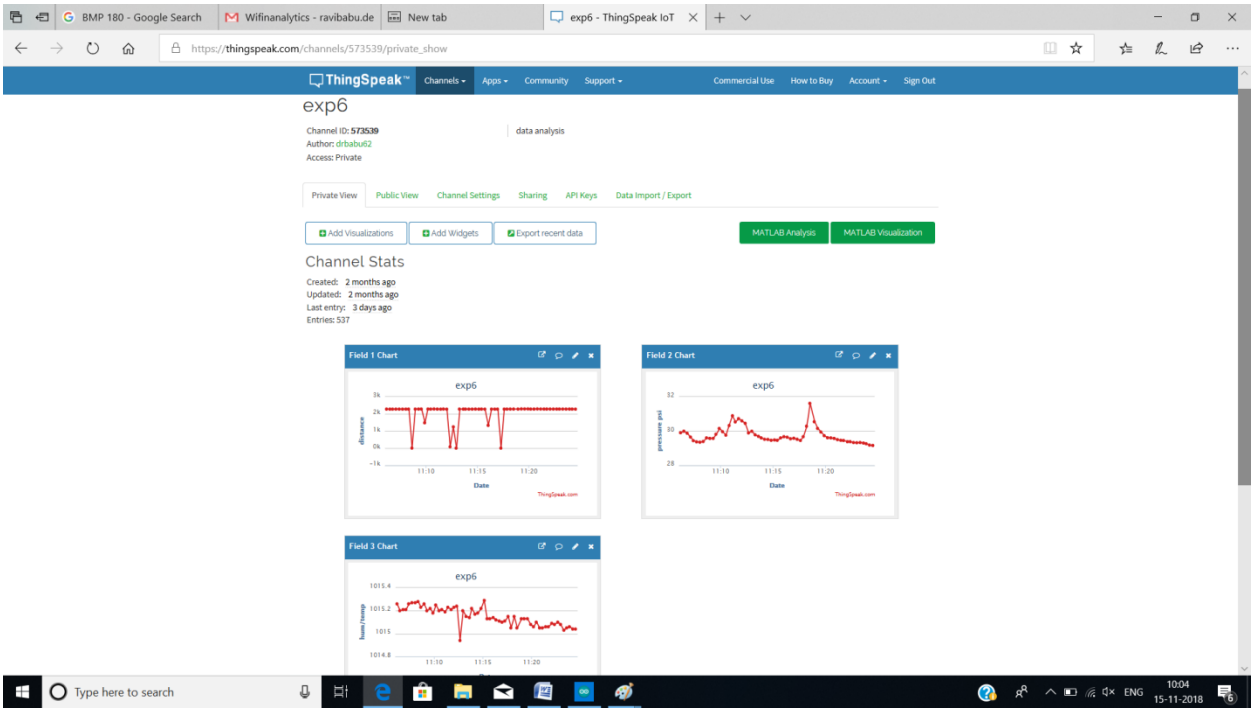
Block Diagram:

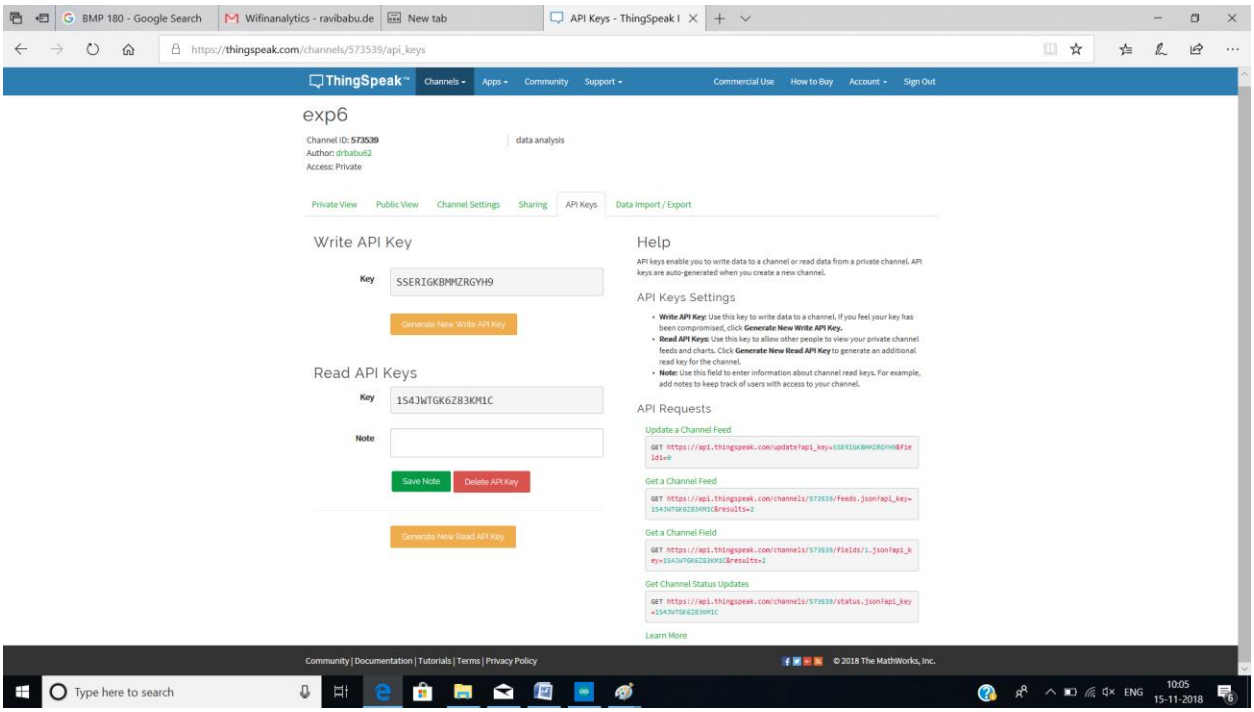




Channel creation in ThingSpeak host




BMP 180 - Google Search
Wifinanalytics - ravibabu.de
New tab
exp6 - ThingSpeak IoT
https://thingspeak.com/channels/573539/private_show
ThingSpeak
Channels
Apps
Community
Support
Commercial Use
How to Buy
Account
Sign Out
exp6
Channel ID: 573539
Author: drbabu62
Access: Private
data analysis
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Channel Stats
Created: 2 months ago
Updated: 2 months ago
Last entry: 3 days ago
Entries: 537
Field 1 Chart
Field 2 Chart
Field 3 Chart
Type here to search
ENG
10:04
15-11-2018


BMP 180 - Google Search
Wifinanalytics - ravibabu.de
New tab
API Keys - ThingSpeak I
https://thingspeak.com/channels/573539/api_keys
ThingSpeak
Channels
Apps
Community
Support
Commercial Use
How to Buy
Account
Sign Out
exp6
Channel ID: 573539
Author: drbabu62
Access: Private
data analysis
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Write API Key
Key
SSERIGKBMMZRGYH9
Generate New Write API Key
Read API Keys
Key
1S4JWTGK6Z83KM1C
Note
Save Note
Delete API Key
Generate New Read API Key
Help
API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.
API Keys Settings
Write API Key: Use this key to write data to a channel. If you feel your key has been compromised, click Generate New Write API Key.
Read API Keys: Use this key to allow other people to view your private channel feeds and charts. Click Generate New Read API Key to generate an additional read key for the channel.
Note: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.
API Requests
Update a Channel Feed
Get a Channel Feed
Get a Channel Field
Get Channel Status Updates
Learn More
Community | Documentation | Tutorials | Terms | Privacy Policy
© 2018 The MathWorks, Inc.
Type here to search
ENG
10:05
15-11-2018

Program code:

```
#include <WiFi.h>
#include <SFE_BMP180.h>
#include <Wire.h>

 #define trigPin 13
 #define echoPin 12
 #define led1 14
 #define led2 34
// You will need to create an SFE_BMP180 object, here called "pressure" SCL to GPIO22, SDA to GPIO 21:

SFE_BMP180 pressure;

#define ALTITUDE 1655.0 // Altitude of SparkFun's HQ in Boulder, CO. in meters

const char* ssid     = "ONLYCSE";//"VSES";//Enter the ssid of your router
const char* password = "keepsmile";//"gnir33nignEtr@mS";//Enter the password of your router

const char* host = "api.thingspeak.com";
const char* privateKey1 = "SSERIGKBMMZRGYH9";//write key
String line,line1;
int duration, distance;
  double T,P,p0,a;
  void setup() {
  Serial.begin(115200);
  Serial.println("REBOOT");
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(led1, INPUT);
  pinMode(led2, INPUT);
  Serial.print("Ultrasonic Sensor Starting!!!");
  Serial.println("");

  // Initialize the sensor (it is important to get calibration values stored on the device).

  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
```

```
    // Oops, something went wrong, this is usually a connection problem,
    // see the comments at the top of this sketch for the proper connections.

    Serial.println("BMP180 init fail\n\n");
    while(1); // Pause forever.
   }


  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

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

void loop()
{
  digitalWrite(trigPin, HIGH);
  delay(1000);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance =(duration/2) * 0.0343;// time * speed of sound(0.0343cm/microsec)
  Serial.print("Distance is");
 Serial.println(distance);
   if (distance < 10)
  {
    digitalWrite(led1, LOW);
```

```
  Serial.println("Low");
 }
 else  if (distance <= 100)
 {
   Serial.println("Half");
   digitalWrite(led1, HIGH);
 }
 else
 {
  digitalWrite(led2, HIGH);
  Serial.println("Out of Range");
  Serial.println("");
 }
 char status;

 // Loop here getting pressure readings every 10 seconds.

 // If you want sea-level-compensated pressure, as used in weather reports,
 // you will need to know the altitude at which your measurements are taken.
 // We're using a constant called ALTITUDE in this sketch:

//  Serial.println();
//  Serial.print("provided altitude: ");
//  Serial.print(ALTITUDE,0);
//  Serial.print(" meters, ");
//  Serial.print(ALTITUDE*3.28084,0);
//  Serial.println(" feet");

 // If you want to measure altitude, and not pressure, you will instead need
 // to provide a known baseline pressure. This is shown at the end of the sketch.

 // You must first get a temperature measurement to perform a pressure reading.

 // Start a temperature measurement:
 // If request is successful, the number of ms to wait is returned.
 // If request is unsuccessful, 0 is returned.

 status = pressure.startTemperature();
 if (status != 0)
 {
```

```
  // Wait for the measurement to complete:
  delay(status);

  // Retrieve the completed temperature measurement:
  // Note that the measurement is stored in the variable T.
  // Function returns 1 if successful, 0 if failure.

  status = pressure.getTemperature(T);
  if (status != 0)
  {
    // Print out the measurement:
    Serial.print("temperature: ");
    Serial.print(T,2);
    Serial.print(" deg C, ");
    Serial.print((9.0/5.0)*T+32.0,2);
    Serial.println(" deg F");

    // Start a pressure measurement:
    // The parameter is the oversampling setting, from 0 to 3 (highest res, longest wait).
    // If request is successful, the number of ms to wait is returned.
    // If request is unsuccessful, 0 is returned.

    status = pressure.startPressure(3);
    if (status != 0)
    {
      // Wait for the measurement to complete:
      delay(status);

      // Retrieve the completed pressure measurement:
      // Note that the measurement is stored in the variable P.
      // Note also that the function requires the previous temperature measurement (T).
      // (If temperature is stable, you can do one temperature measurement for a number of
pressure measurements.)
      // Function returns 1 if successful, 0 if failure.

      status = pressure.getPressure(P,T);
      if (status != 0)
      {
        // Print out the measurement:
        Serial.print("absolute pressure: ");
```

```
        Serial.print(P,2);
        Serial.print(" mb, ");
        Serial.print(P*0.0295333727,2);
        Serial.println(" inHg");

        // The pressure sensor returns abolute pressure, which varies with altitude.
        // To remove the effects of altitude, use the sealevel function and your current altitude.
        // This number is commonly used in weather reports.
        // Parameters: P = absolute pressure in mb, ALTITUDE = current altitude in m.
        // Result: p0 = sea-level compensated pressure in mb

        p0 = pressure.sealevel(P,ALTITUDE); // we're at 1655 meters (Boulder, CO)
        Serial.print("relative (sea-level) pressure: ");
        Serial.print(p0,2);
        Serial.print(" mb, ");
        Serial.print(p0*0.0295333727,2);
        Serial.println(" inHg");

        // On the other hand, if you want to determine your altitude from the pressure reading,
        // use the altitude function along with a baseline pressure (sea-level or other).
        // Parameters: P = absolute pressure in mb, p0 = baseline pressure in mb.
        // Result: a = altitude in m.
//
//          a = pressure.altitude(P,p0);
//          Serial.print("computed altitude: ");
//          Serial.print(a,0);
//          Serial.print(" meters, ");
//          Serial.print(a*3.28084,0);
//          Serial.println(" feet");
       }
       else Serial.println("error retrieving pressure measurement\n");
     }
     else Serial.println("error starting pressure measurement\n");
   }
   else Serial.println("error retrieving temperature measurement\n");
  }
  else Serial.println("error starting temperature measurement\n");
  upload();
  delay(15000);
}
```

```
void upload()
{
 Serial.print("connecting to ");
 Serial.println(host);

 // Use WiFiClient class to create TCP connections
 WiFiClient client;
 const int httpPort = 80;

 if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
 }

  // We now create a URI for the request
 String url = "/update";
 url += "?api_key=";
 url += privateKey1;
 url += "&field1=";
 url += distance;
 url += "&field2=";
 url += T;
 url += "&field3=";
 url += P;
  Serial.print("Requesting URL: ");
 Serial.println(url);
   // This will send the request to the server
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: close\r\n\r\n");
 delay(1000);
  // Read all the lines of the reply from server and print them to Serial
 while(client.available())
 {
 String line1 = client.readStringUntil('\r');
 Serial.print(line1);
 }
 Serial.println();
 Serial.println("closing connection");}
```

Experiment -6: Read Temperature and Humidity using DHT11 sensor with Raspberry pi

Aim: To study the DHT11 sensor with raspberry pi to read the Temperature and Humidity

Objectives: Student should get the knowledge of python script, Raspberry pi and DHT11 sensor.

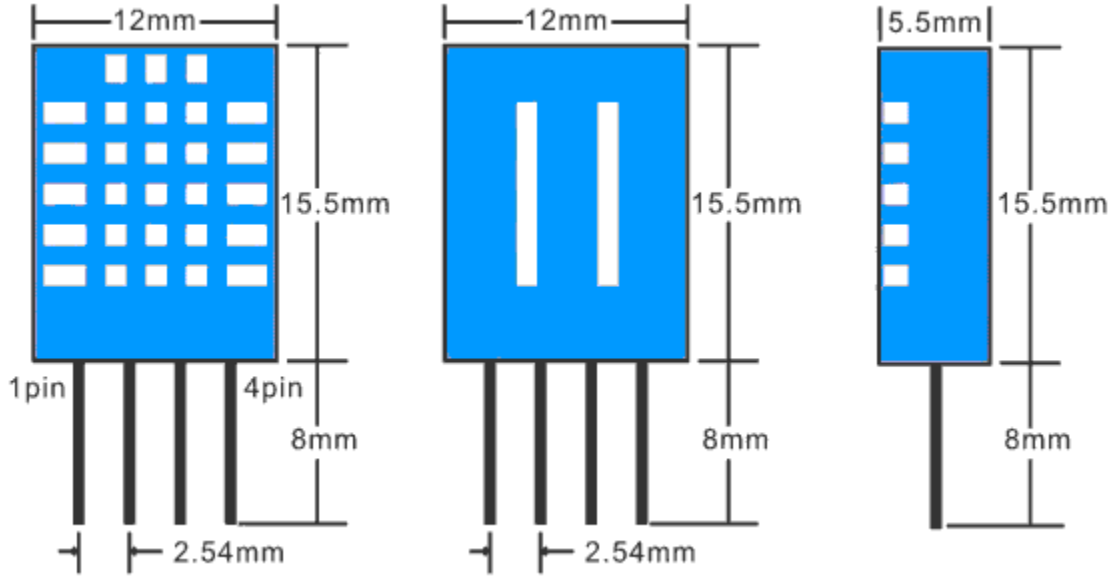
Outcomes: Student will be aware of python script, Raspberry pi and DHT11 sensor

DHT11: It is a low-cost temperature and humidity sensor. It isn't the fastest sensor around but its cheap price makes it useful for experimenting or projects where you don't require new readings multiple times a second.

The device only requires three connections to the Pi. +3.3v, ground and one GPIO pin.

DHT11 Specifications

The device itself has four pins but one of these is not used. You can buy the 4-pin device on its own or as part of a 3-pin module.



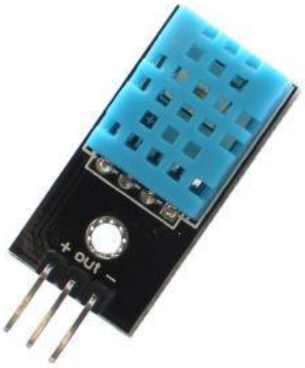
The modules have three pins and are easy to connect directly to the Pi's GPIO header.

Humidity : 20-80% (5% accuracy)

Temperature : 0-50°C (±2°C accuracy)

The manufacturers do not recommended that you read data from this device more than once per 2 seconds. If you do you may get incorrect readings.

Hardware Setup



The 4-pin device will require a resistor (4.7K-10K) to be placed between Pin 1 (3.3V) and Pin 2 (Data).

The 3-pin modules will usually have this resistor included which makes the wiring a bit easier. For this reason I got hold of the module which I could then attach to the Pi with a piece of 3-way Dupont cable.

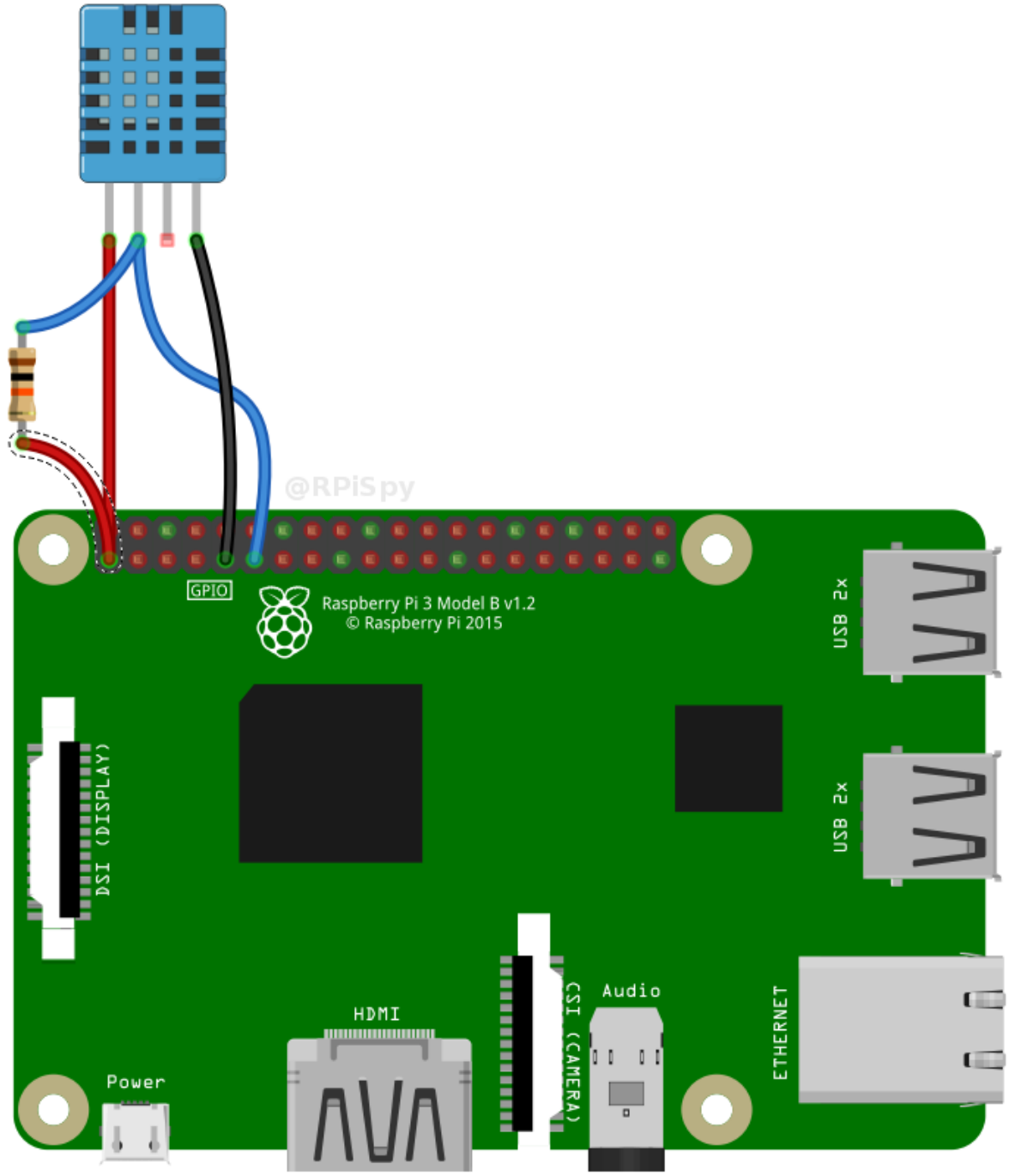
Different suppliers may wire the module pins differently so check the PCB markings to identify Vcc (+), data and Ground (-).

The 3 pins should be connected to the Pi as shown in the table below :

DHT Pin	Signal	Pi Pin
1	3.3V	1
2	Data/Out	11 (GPIO17)
3	not used	–

4	Ground	6 or 9

Your data pin can be attached to any GPIO pin you prefer. In my example I am using physical pin 11 which is GPIO 17. Here is a 4-pin sensor connected to the Pi's GPIO header. It has a 10K resistor between pin 1 (3.3V) and 2 (Data/Out).



Python Library

The DHT11 requires a specific protocol to be applied to the data pin. In order to save time trying to implement this yourself it's far easier to use the Adafruit DHT library.

The library deals with the data that needs to be exchanged with the sensor but it is sensitive to timing issues. The Pi's operating system may get in the way while performing other tasks so to compensate for this the library requests a number of readings from the device until it gets one that is valid.

Software Setup

To start with update your package lists and install a few Python libraries :

sudo apt-get update

sudo apt-get install build-essential python-dev

Then clone the Adafruit library from their repository :

git clone https://github.com/adafruit/Adafruit_Python_DHT.git

cd Adafruit_Python_DHT

Then install the library for Python 2 and Python 3 :

sudo python setup.py install

sudo python3 setup.py install

Hopefully at this point the library is installed and ready to be used within a Python script.

Adafruit Example Python Script

Adafruit provide an example script that you can use to check your sensor is operating correctly.

cd ~

cd Adafruit_Python_DHT

cd examples

Then :

python DHT11.py 11 17

The example script takes two parameters. The first is the sensor type so is set to "11" to represent the DHT11. The second is the GPIO number so for my example I am using "17" for GPIO17. You can change this if you are using a different GPIO pin for your data/out wire.

You should see an output similar to this :

Temp=22.0* Humidity=68.0%

Program Code: DHT11.PY

```
import Adafruit_DHT

# Set sensor type : Options are DHT11,DHT22 or AM2302
sensor=Adafruit_DHT.DHT11

# Set GPIO sensor is connected to
gpio=17

# Use read_retry method. This will retry up to 15 times to
# get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

# Reading the DHT11 is very sensitive to timings and occasionally
# the Pi might fail to get a valid reading. So check if readings are valid.
if humidity is not None and temperature is not None:
print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
print('Failed to get reading. Try again!')
```

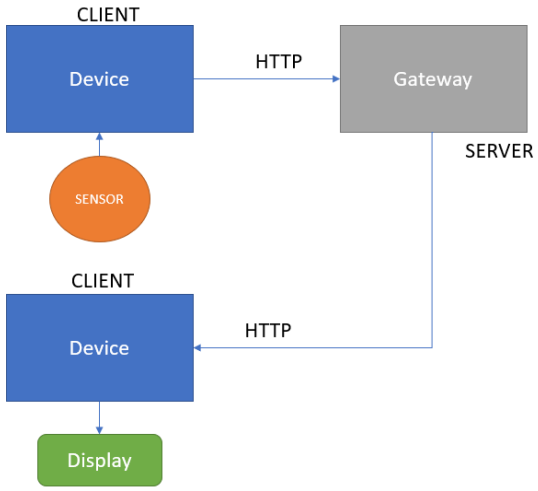
Experiment -7: Study of various network protocols used in IoT

To study the network protocols used in IoT, we shall establish either device to device communication or device to gateway communication. I recommend to establish device to gateway communication using the following hardware setup. The experiment can be performed with the help of device, gateway and applications. To work HTTP, user can configure devices or use open software tools (postmaster for HTTP, Mqtt lens for MQTT) in place of multiple devices.

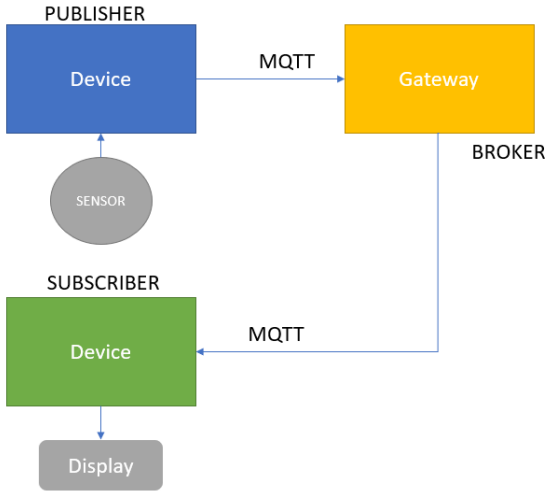
Objectives: Student should get the knowledge of network protocols used in IOT

Outcomes: Student will be aware the protocols HTTP and MQTT using in IOT

HTTP Protocol: Client – Server Model



MQTT Protocol: Publish – Subscribe model



MQTT:

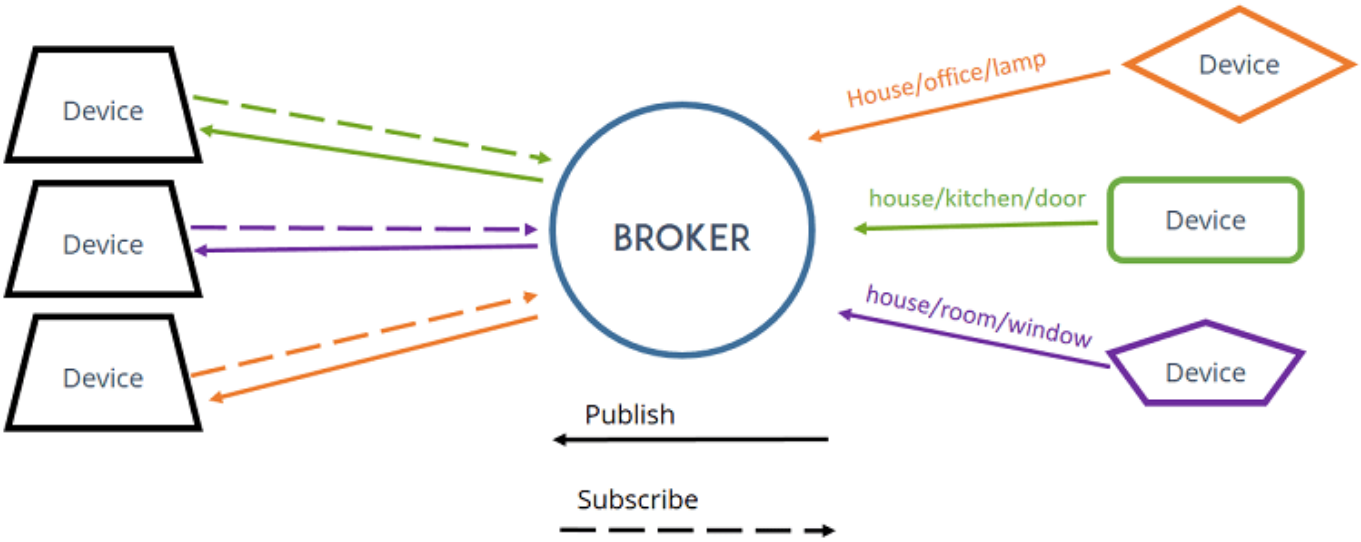
Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_sub command line MQTT clients.

MQTT – Broker

At last, you also need to be aware of the term *broker*.
The broker is primarily responsible for **receiving** all messages, **filtering** the messages, **decide** who is interested in them and then **publishing** the message to all subscribed clients.



There are several brokers you can use. In our home automation projects we use the Mosquitto broker which can be installed in the Raspberry Pi. Alternatively, you can use a cloud MQTT broker.

Hardware Recommended:

S.No.	Devices/Sensors	Qty, Nos.
1	ESP32 Hardware Development Board	2
2	Gateway (Raspberry Pi3)	1
3	Connecting wires	15
4	Basic shield of SB	1

Raspberry Pi GPIOs Pinout

The following figure shows the Raspberry Pi 3 GPIOs pinout that you can use for a future reference.

		Pin no.			
DC Power	3.3V	1	2	5V	DC Power
SDA1, I²C	GPIO 2	3	4	5V	DC Power
SCL1, I²C	GPIO 3	5	6	GND	
GPIO_GCLK	GPIO 4	7	8	GPIO 14	TXD0
	GND	9	10	GPIO 15	RXD0
GPIO_GEN0	GPIO 17	11	12	GPIO 18	GPIO_GEN1
GPIO_GEN2	GPIO 27	13	14	GND	
GPIO_GEN3	GPIO 22	15	16	GPIO 23	GPIO_GEN4
DC Power	3.3V	17	18	GPIO 24	GPIO_GEN5
SPI_MOSI	GPIO 10	19	20	GND	
SPI_MISO	GPIO 9	21	22	GPIO 25	GPIO_GEN6
SPI_CLK	GPIO 11	23	24	GPIO 8	SPI_CE0_N
	GND	25	26	GPIO 7	SPI_CE1_N
I²C ID EEPROM	DNC	27	28	DNC	I²C ID EEPROM
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
	GPIO 19	35	36	GPIO 16	
	GPIO 26	37	38	GPIO 20	
	GND	39	40	GPIO 21	

This pinout is the same for Raspberry Pi 2 Model B, Raspberry Pi 1 Model A+, Raspberry Pi Model B+, Raspberry Pi Zero, and Raspberry Pi Zero W.

Raspberry Pi 1 Model A and the Raspberry Pi 1 Model B Rev.2 only have the first 26 pins.

Program Code:

MQTT:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include<Servo.h>
#include "DHT.h"
#define DHTPIN D2
float t;// what pin we're connected to
#define DHTTYPE DHT11   // define type of sensor DHT 11
DHT dht (DHTPIN, DHTTYPE);
Servo myservo;

#define led1 D0  // BLUE LED  PIR
#define led2 D1  // RED LED  GAS
#define led3 D5  // RED LED  FLAME
#define led4 D4  // BLUE LED  PIR


#define gas_sensor A0
#define flame_sensor D3
#define pir_sensor D8

//servo D7

void reconnect();
void MQpublish();
String cmd;

// Update these with values suitable for your network.

const char* ssid = "VSES";
const char* password = "gnir33nignEtr@mS";
const char* mqtt_server = "192.168.0.121";


WiFiClient espClient;
PubSubClient client(espClient);

void setup()
{
  Serial.begin(115200);
  dht.begin();
 setup_wifi();
```

```
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
 myservo.attach(D7);
 pinMode(D7,OUTPUT);
 pinMode(led1,OUTPUT);
 pinMode(led2,OUTPUT);
 pinMode(led4,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(gas_sensor,INPUT);
  pinMode(flame_sensor,INPUT);
   pinMode(pir_sensor,INPUT);

}

void loop() {
  float h = dht.readHumidity();
 t= dht.readTemperature();
  delay(100);
  if (!client.connected())
  {
   reconnect();
  }

  client.loop();
  cmd ="";

MQpublish();
delay(2000);
}
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid,password);

  while (WiFi.status()!= WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length)
{
  Serial.print("Message arrived[");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    //Serial.print((char)payload[i]);
    cmd = cmd + String((char)payload[i]);
  }
  Serial.println(cmd);
    // Switch on the LED if an 1 was received as first character
    cmd ="";
}
void reconnect()
{
  // Loop until we're reconnected
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("espClient")) {
      Serial.println("connected");
      client.subscribe("testTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


void MQpublish()
{
String data=String(t);
client.publish("testTopic","20");

}
```