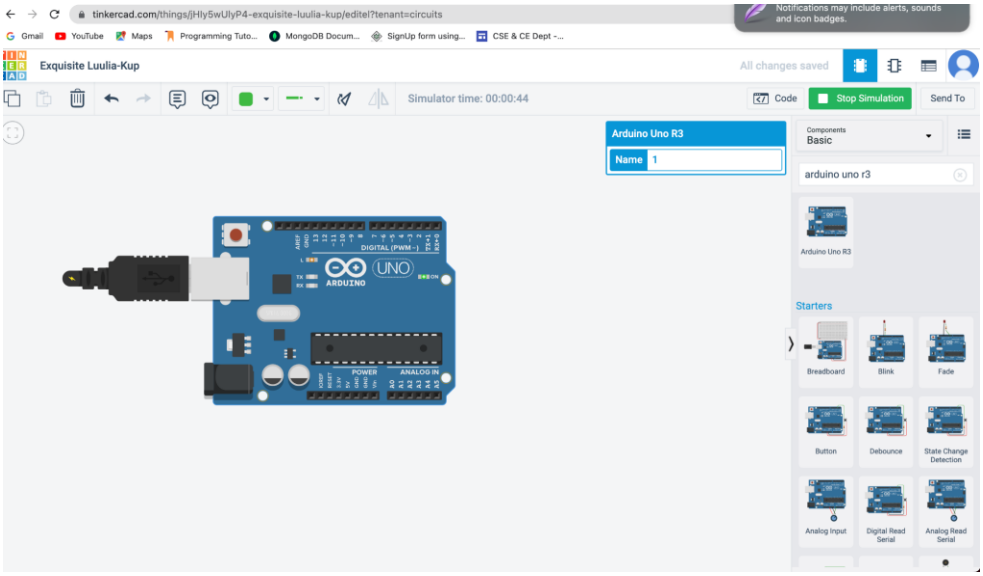
Part 2

Program 1

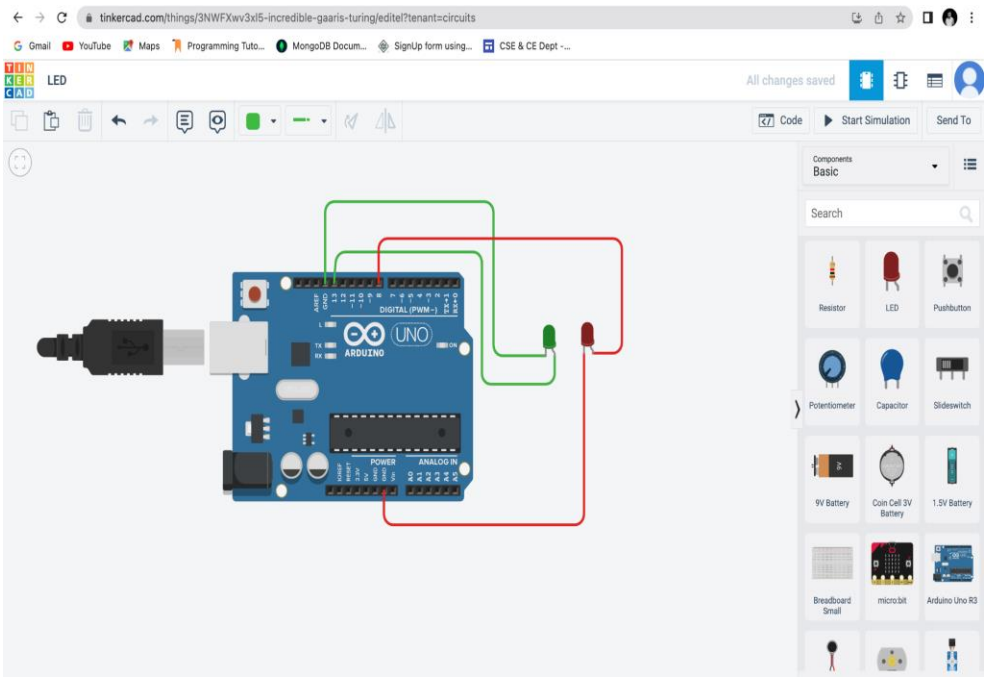
Aim: Perform LED on-off using tinkercad.

Implementation:

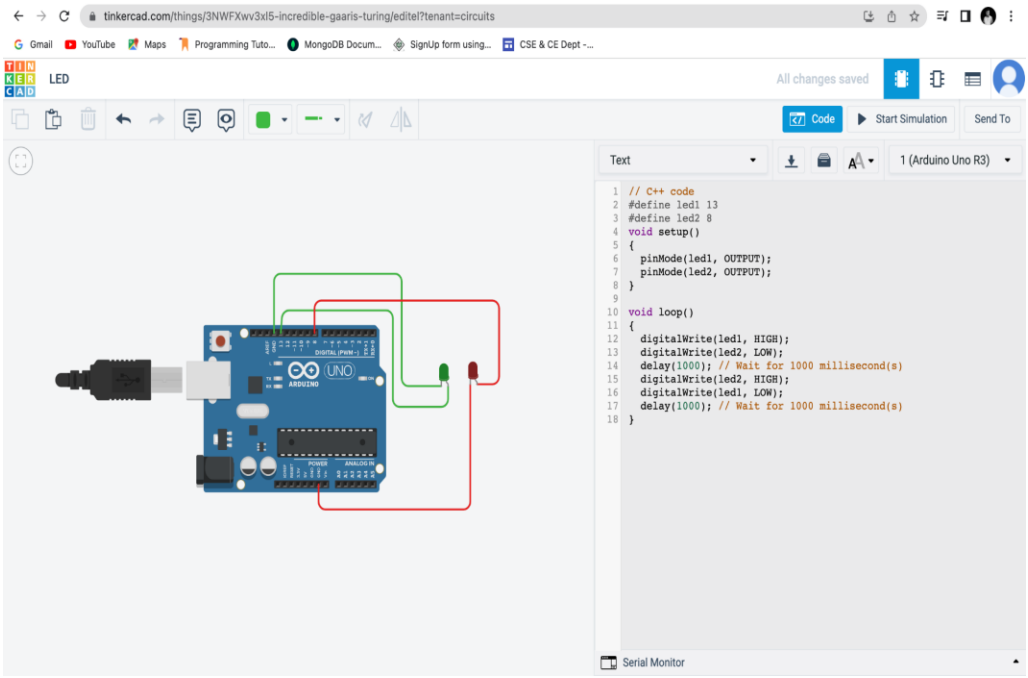
Step1: Create a new circuit and search for “arduino uno r3”.



Step 2: Search for LED light and connect with arduino uno r3



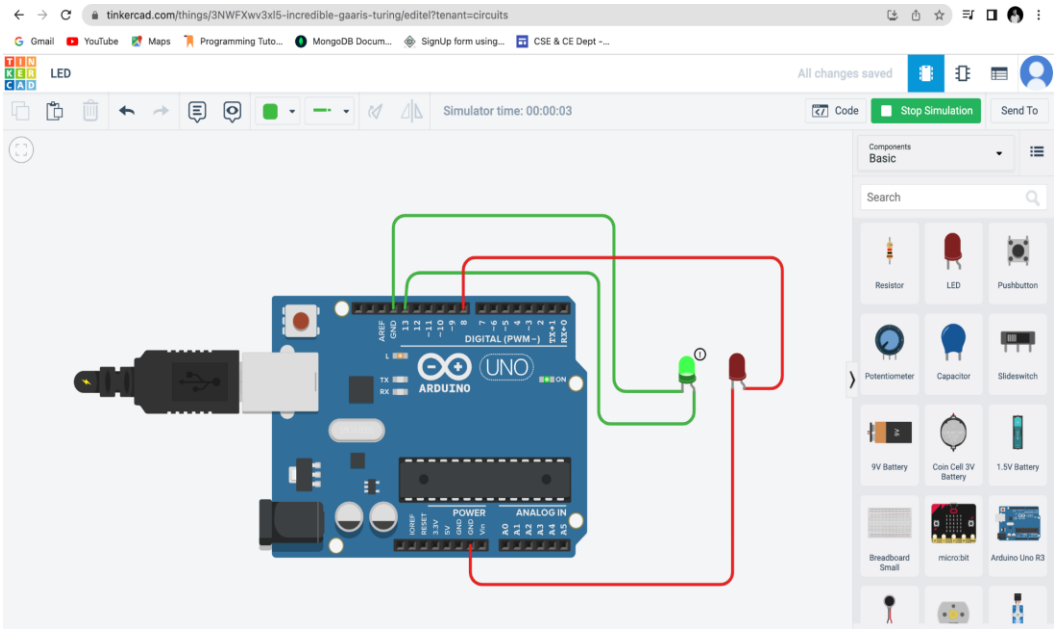
Step 3: Now, write the code to ON-OFF the LED lights.

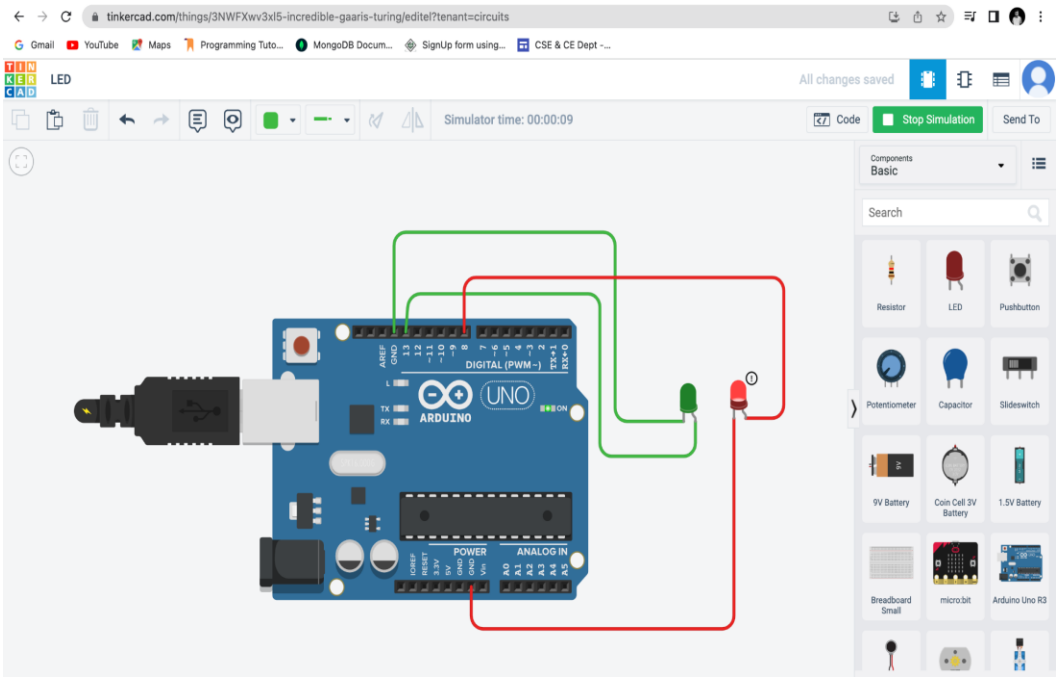


Explanation of Code:-

```
// C++ code
#define led1 13  //Green colour LED name given
#define led2 8   //RED colour LED name given
void setup()
{
  pinMode(led1, OUTPUT); // led is use to show output
  pinMode(led2, OUTPUT);//That's Why pinmode use as output
}

void loop()
{
  digitalWrite(led1, HIGH);//Green colour LED will on
  digitalWrite(led2, LOW);//RED colour LED will off
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(led2, HIGH);//RED colour LED will on
  digitalWrite(led1, LOW);//Green colour LED will of
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Step 4: Start simulation and see the Output.


tinkercad.com/things/3NWFXwv3xI5-incredible-gaaris-turing/editel?tenant=circuits
LED
All changes saved
Simulator time: 00:00:03
Code
Stop Simulation
Send To
Components
Basic
Search
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
DIGITAL (PWM~)
ARDUINO
UNO
POWER
ANALOG IN


tinkercad.com/things/3NWFXwv3xI5-incredible-gaaris-turing/editel?tenant=circuits
LED
All changes saved
Simulator time: 00:00:09
Code
Stop Simulation
Send To
Components
Basic
Search
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
DIGITAL (PWM~)
ARDUINO
UNO
POWER
ANALOG IN

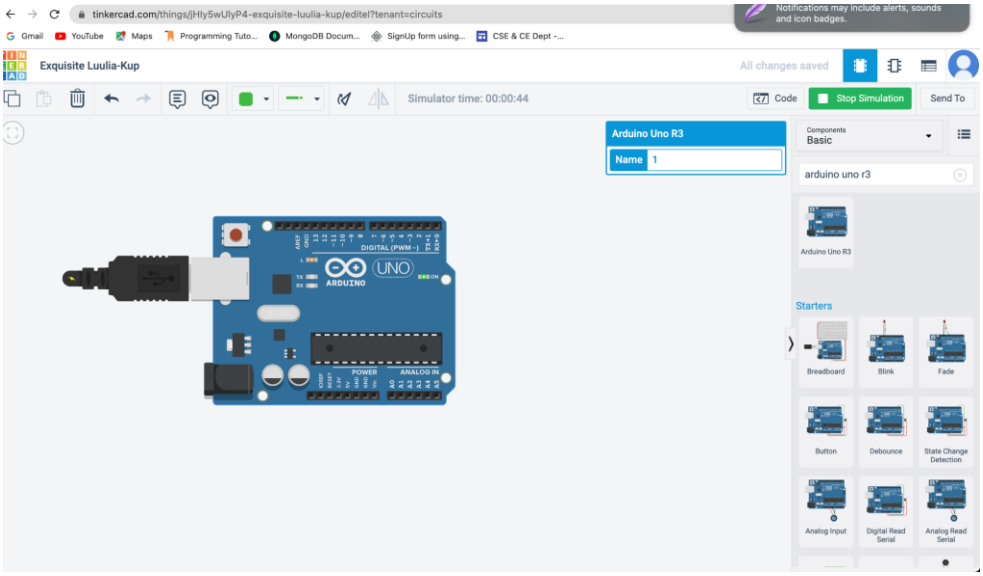
Part 2

Program 2

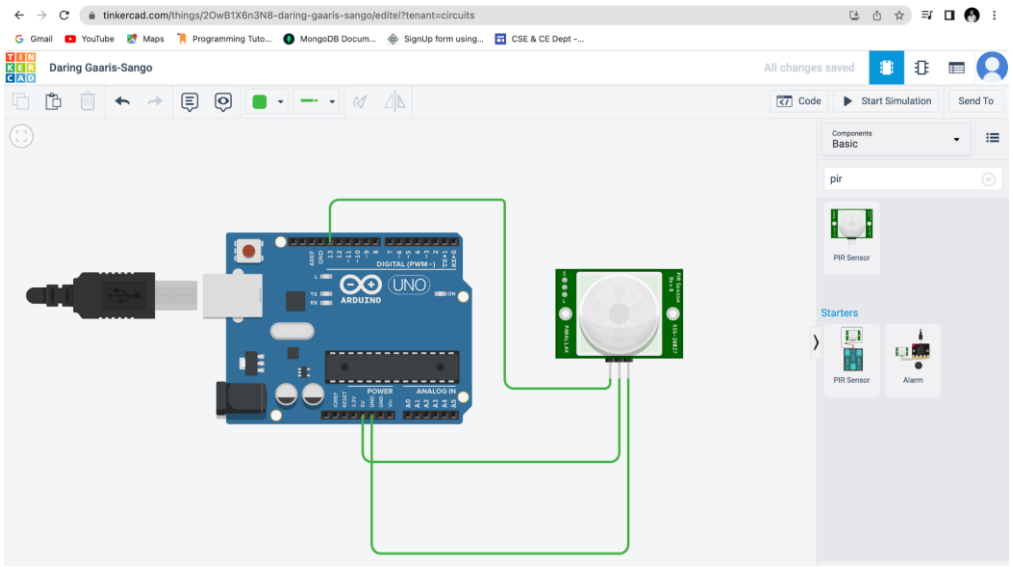
Aim: Analyse the performance of PIR sensor on tinkercad.

Implementation:

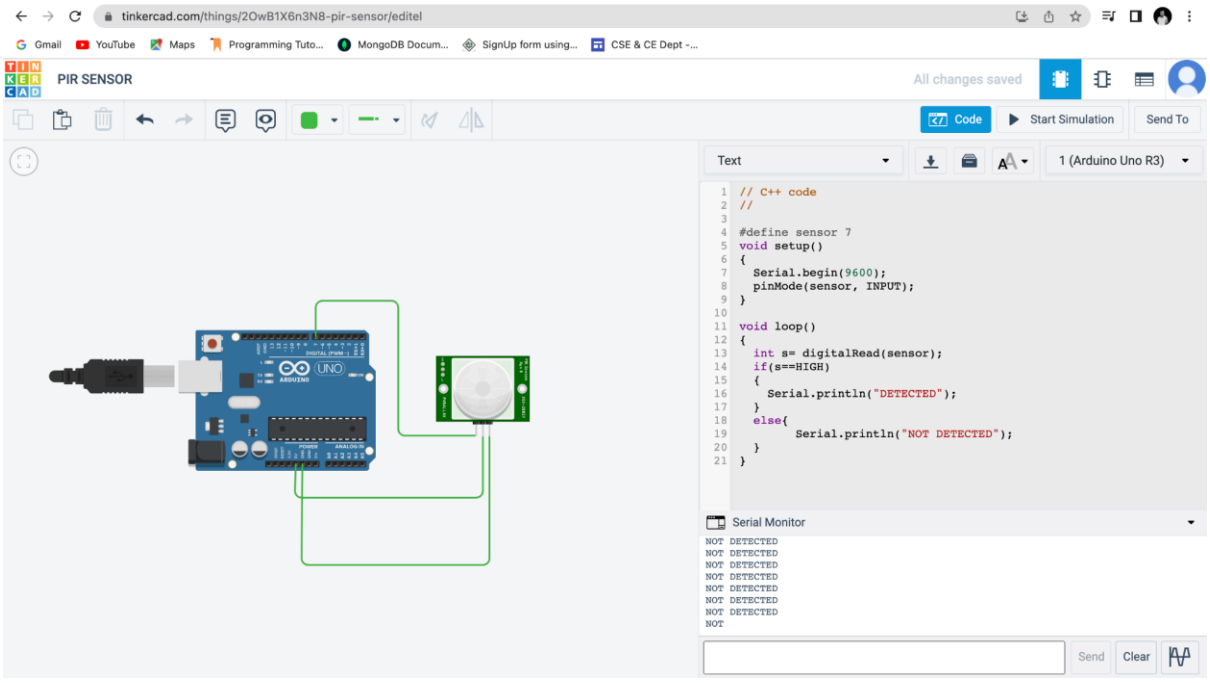
Step1: Create a new circuit and search for “arduino uno r3”.



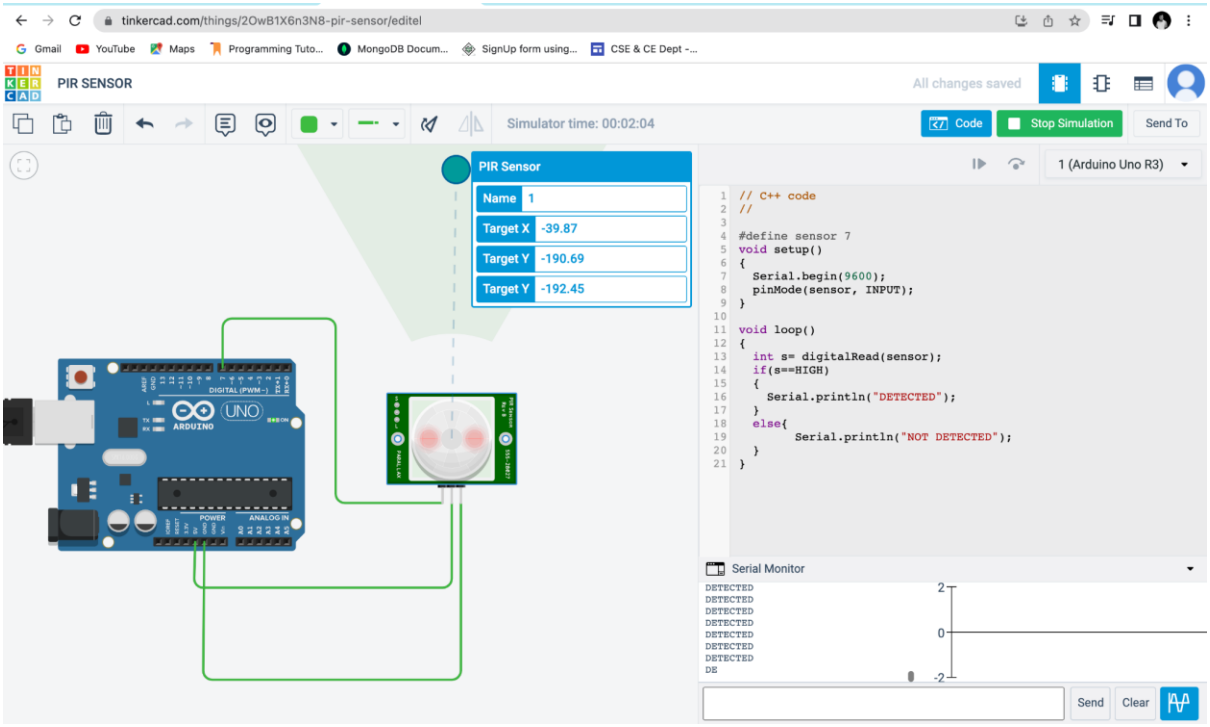
Step 2: Search for PIR Sensor and connect with arduino uno r3



Step 3: Now, write the code to sense to object using PIR Sensor.



Step 4: Start simulation and see the Output.



Part - 3

Practical – 4

Aim: Installation of Node-Red on Standalone system and perform these tasks:

1. Building a First flow: Hello World

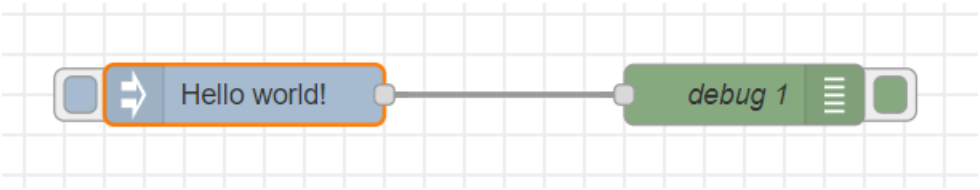
2. Building a Second flow: Retrieving data from a web page

3. Build a Fourth flow: Demo of MQTT using Node-RED

Answer:

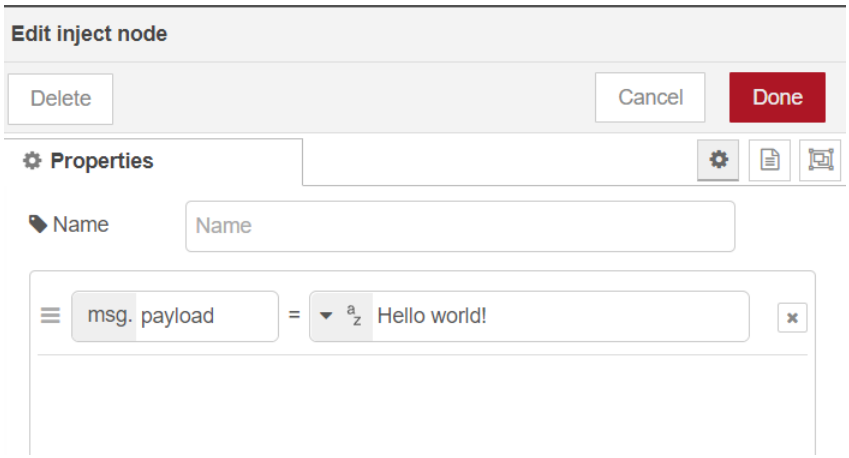
1. **Building a First flow: Hello World**

Flow:

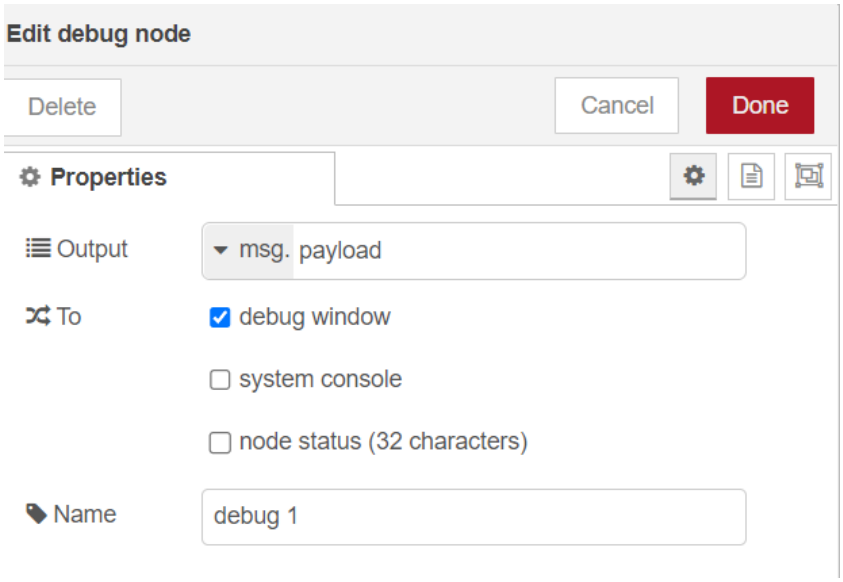


Node configuration:

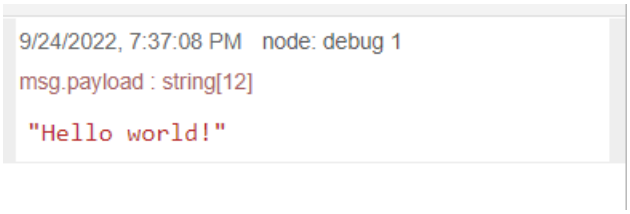
*Input node node:



*Debug(output) node:

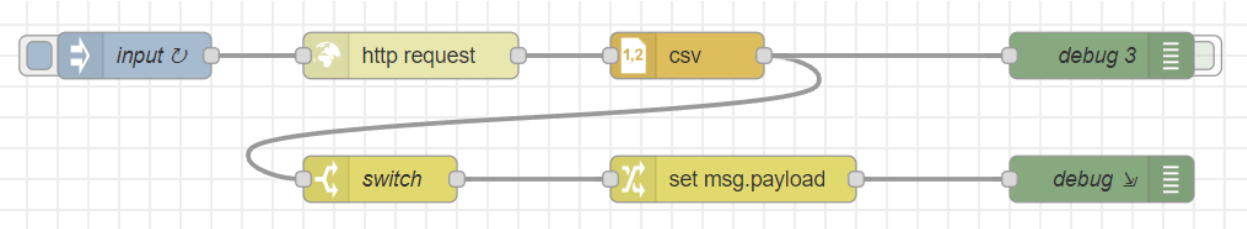


Output:



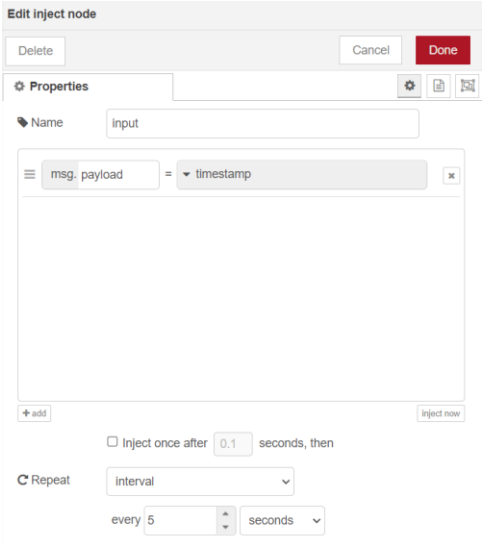
2. Building a Second flow: Retrieving data from a web page

Flow:

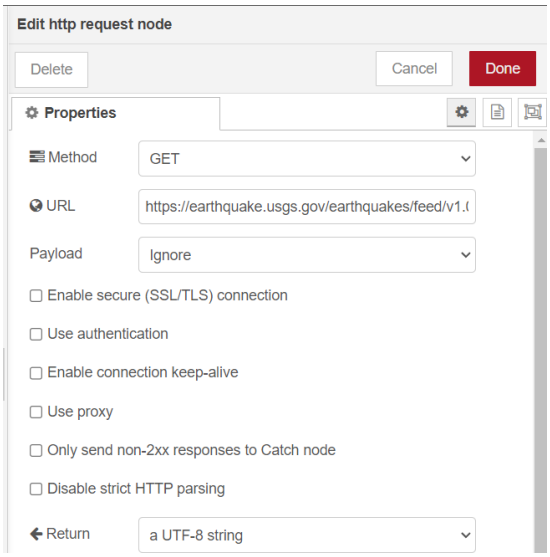


Node Configuration:

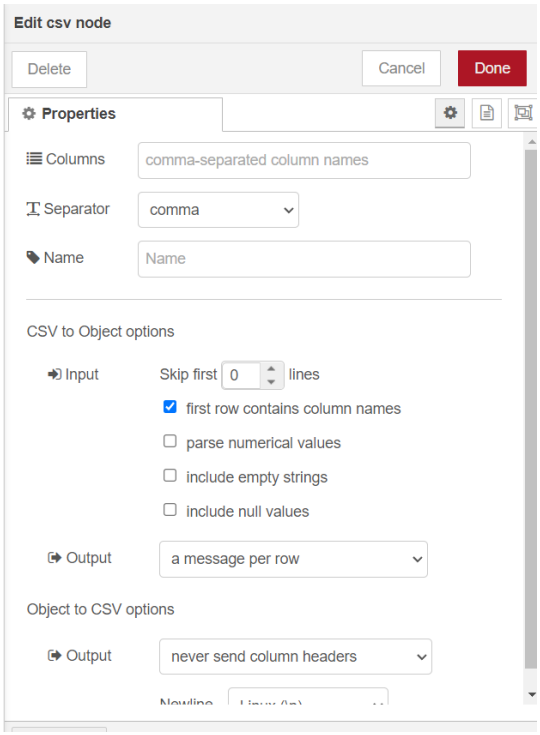
*input node:



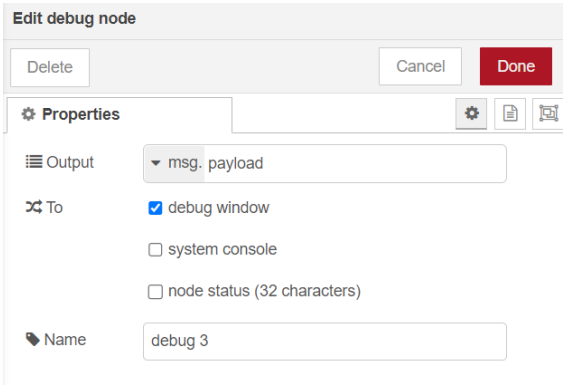
*http request node:



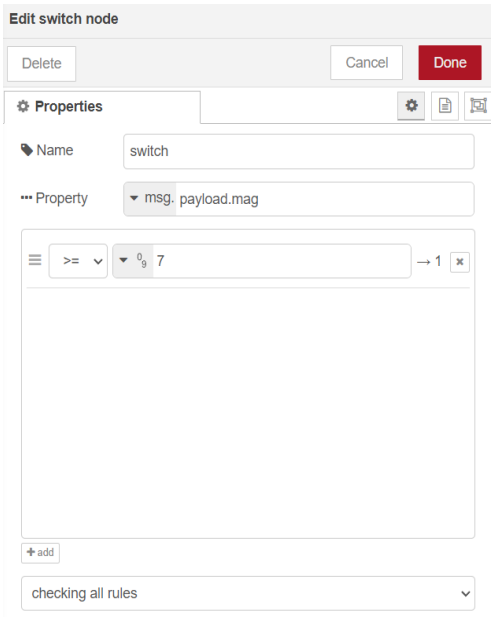
*csv node:



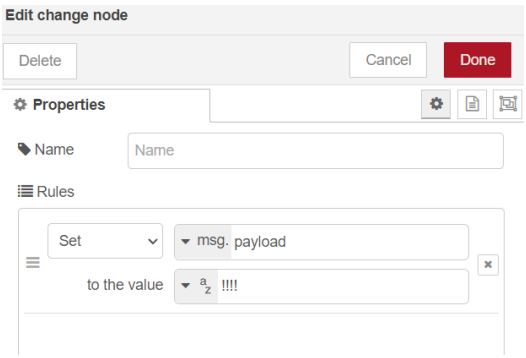
*debug3 node:



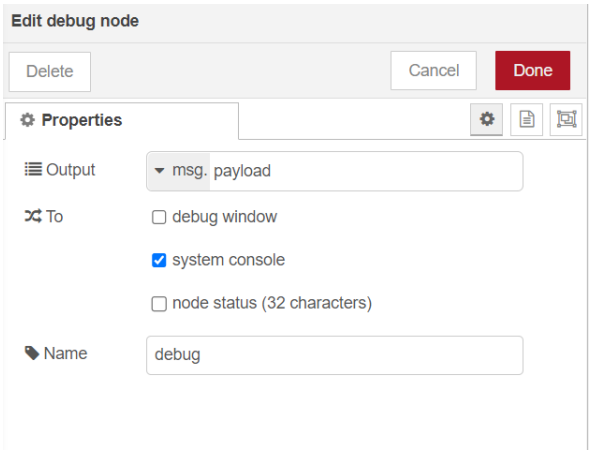
*Switch node :



*set msg.payload node:



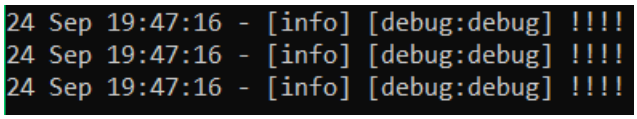
*debug node:



Output:

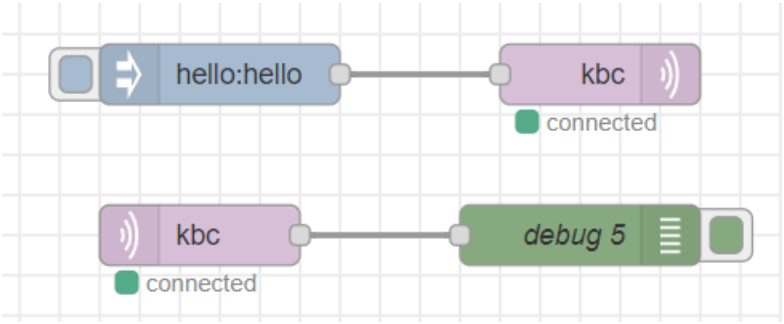
*All values:

*Triggered value:

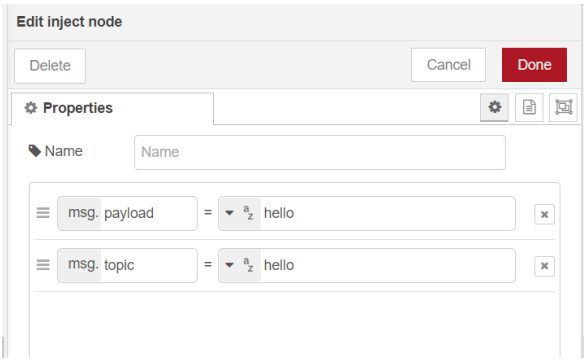


3. Build a Fourth flow: Demo of MQTT using Node-RED

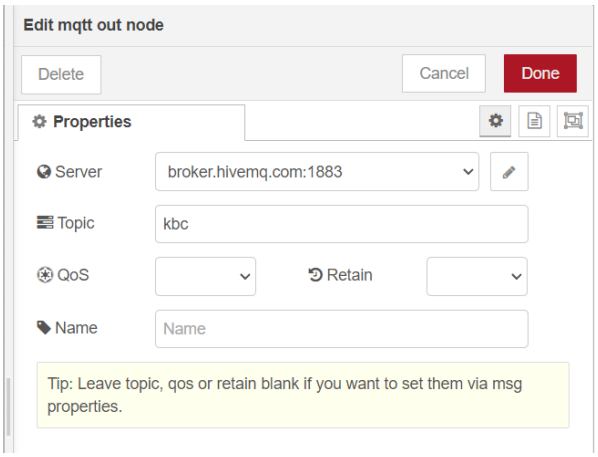
Flow:



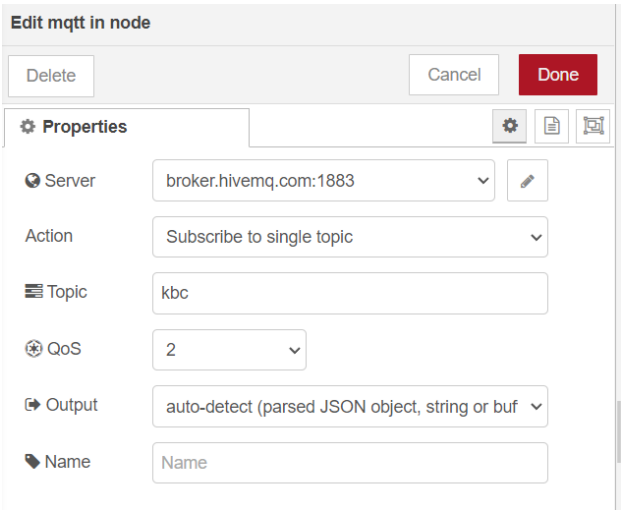
*Input node:



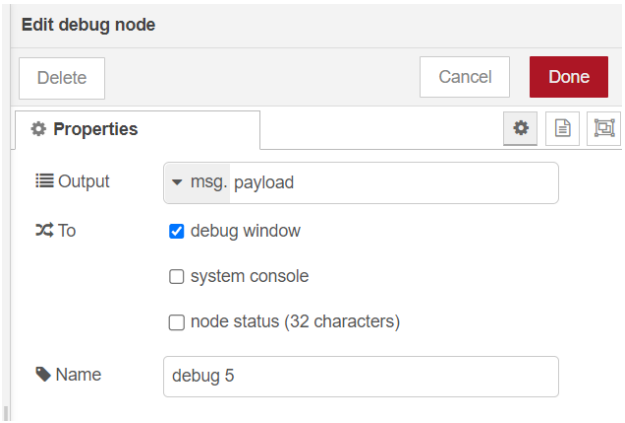
*mqtt in node:



*mqtt out node:



*Debug node



Output:

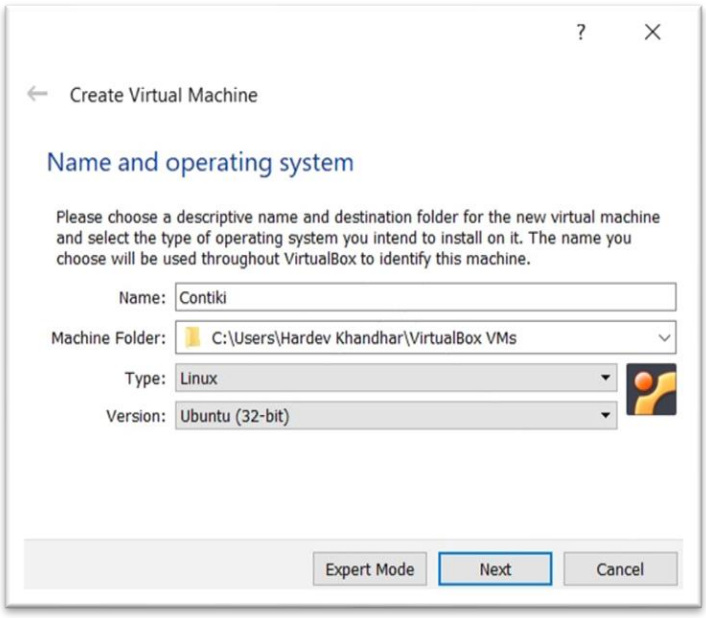
Part 3

Program 1

Aim: Installation and configuration of Instant Contiki OS with Cooja. Perform basic program of Hello World in Cooja Simulator.

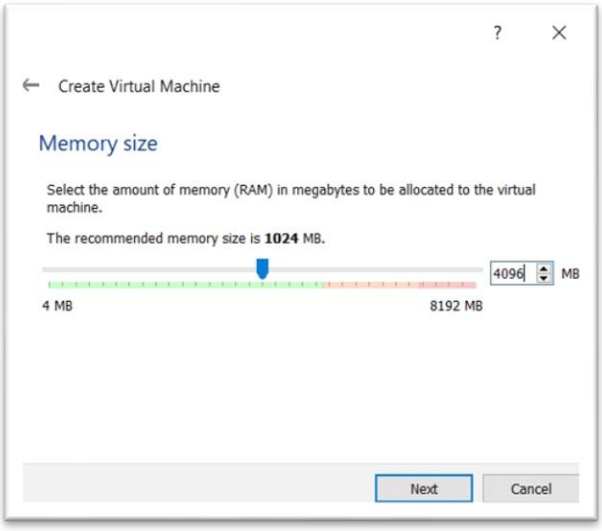
Implementation:

- Open VirtualBox & create a new Virtual Machine (VM)
- Give any preferable name
- Select type as *Linux*
- Select version as *Ubuntu (32-bit)*



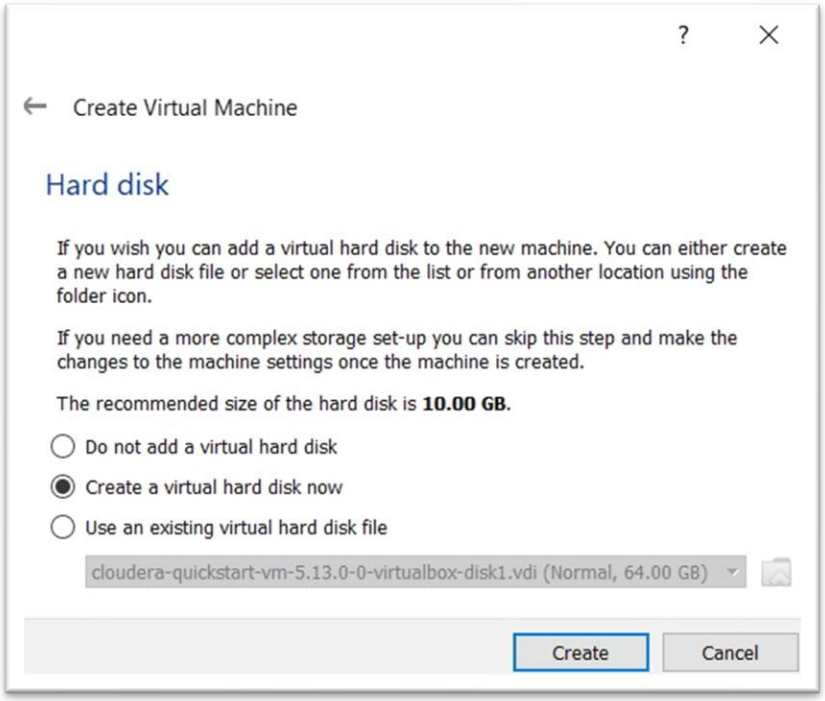
Setup a new VM in VirtualBox

- Click Next
- Select Memory Size
- More than 2GB is preferable, here 4096 MB (4GB) memory size is allocated



Allocating memory size to VM

The next step is to select the Hard Disk

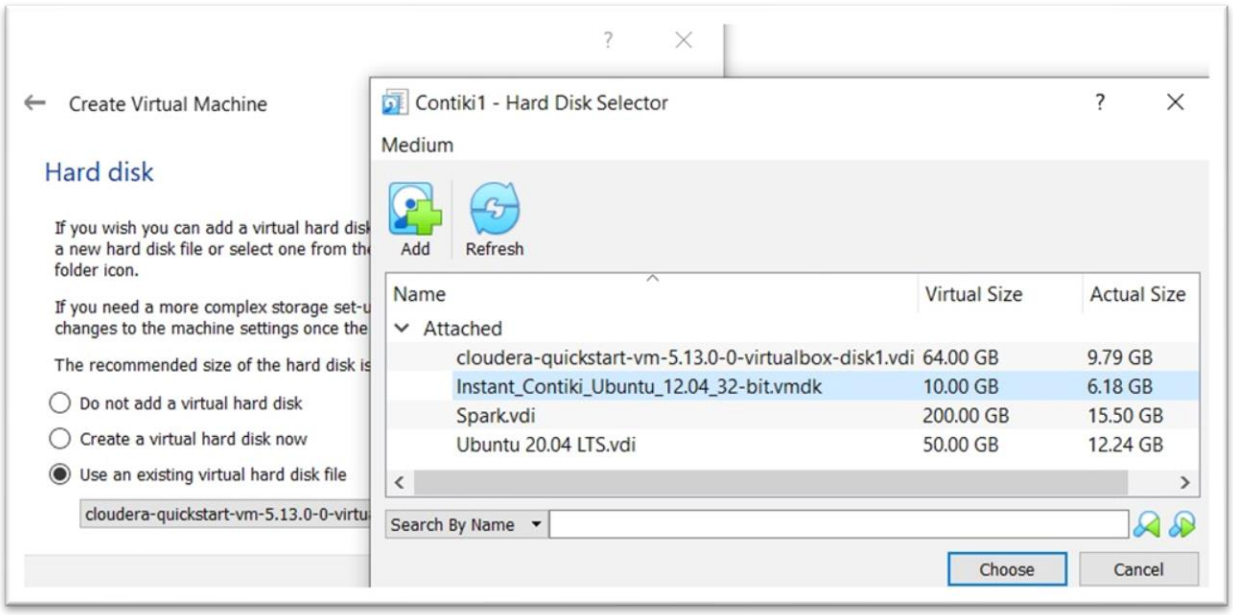


Hard Disk Selection Menu

Select the option *Use an existing virtual hard disk file*

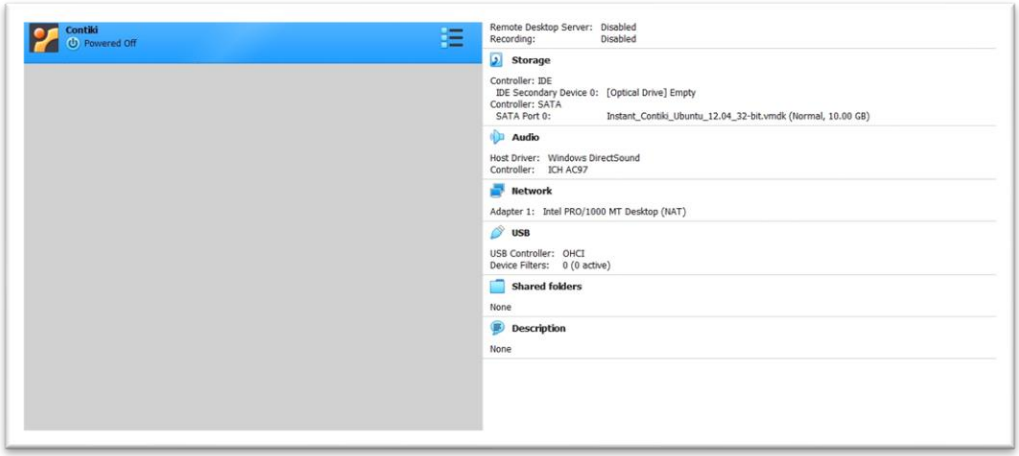
This will open a hard disk selector

- Choose the Instant Contiki download & select the vmdk file that does NOT have –s00* trailing



Selecting Instant Contiki Hard Disk vmdk file

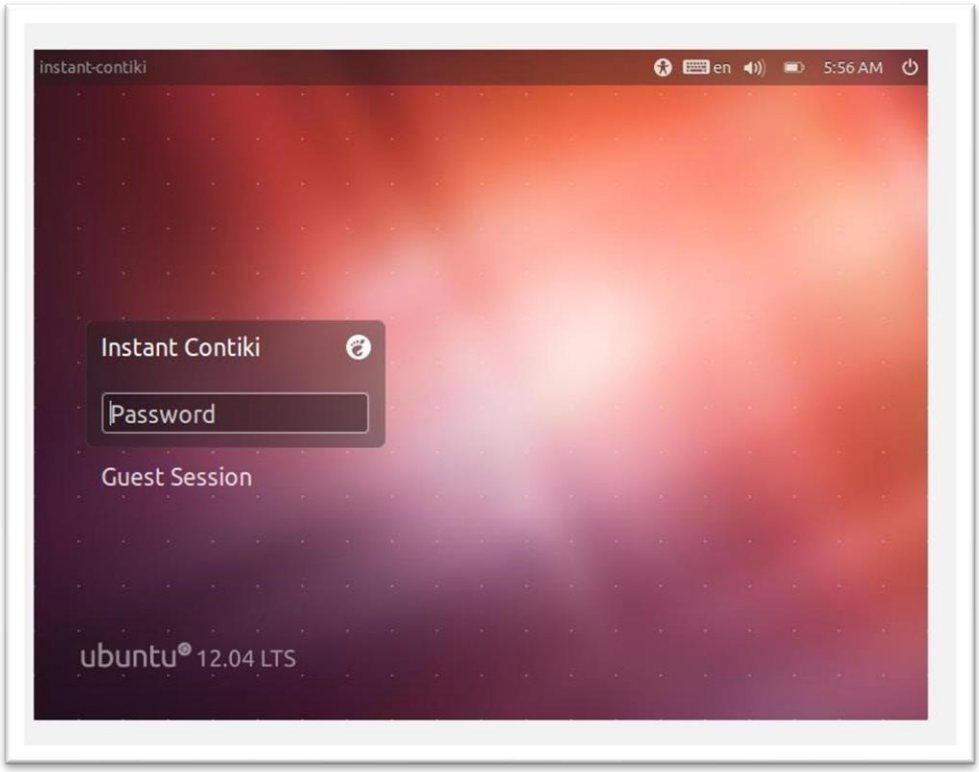
- Click on *Choose* & then click on *Create Virtual Machine* ⍰ This creates our Virtual Machine with Contiki OS.



Newly created VM is configured & ready to use

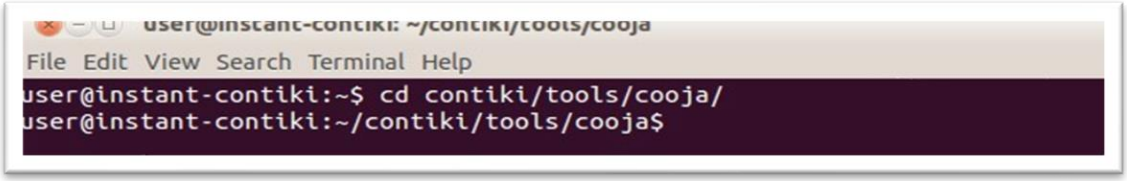
- Select the Contiki VM & click on *Start*
- This will start the Virtual Machine

Once, the login screen appears, enter the password as ***user***

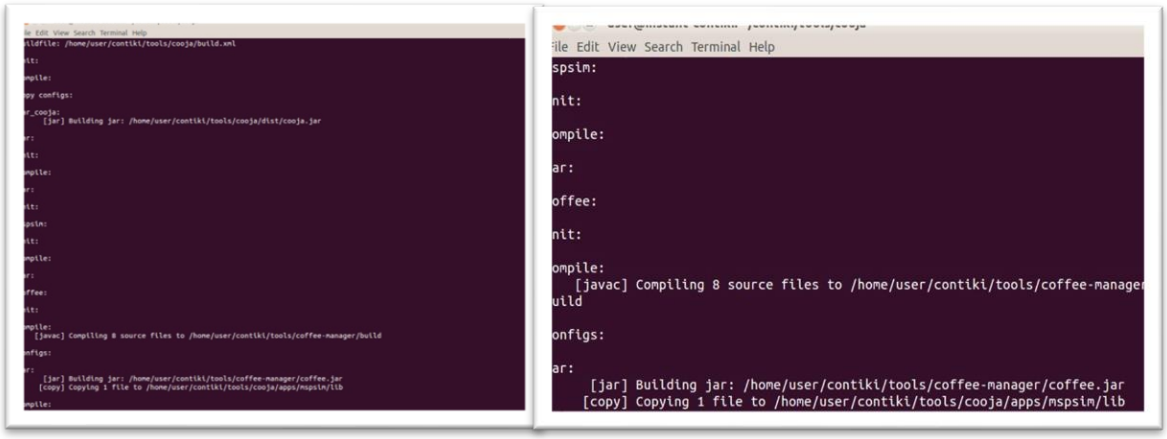


Login Screen for Instant Contiki OS

- Now, to launch Cooja in the Contiki OS, first open a terminal
- Navigate to *Cooja* directory which is located at */Contiki/tools/Cooja* Execute the command *ant run*

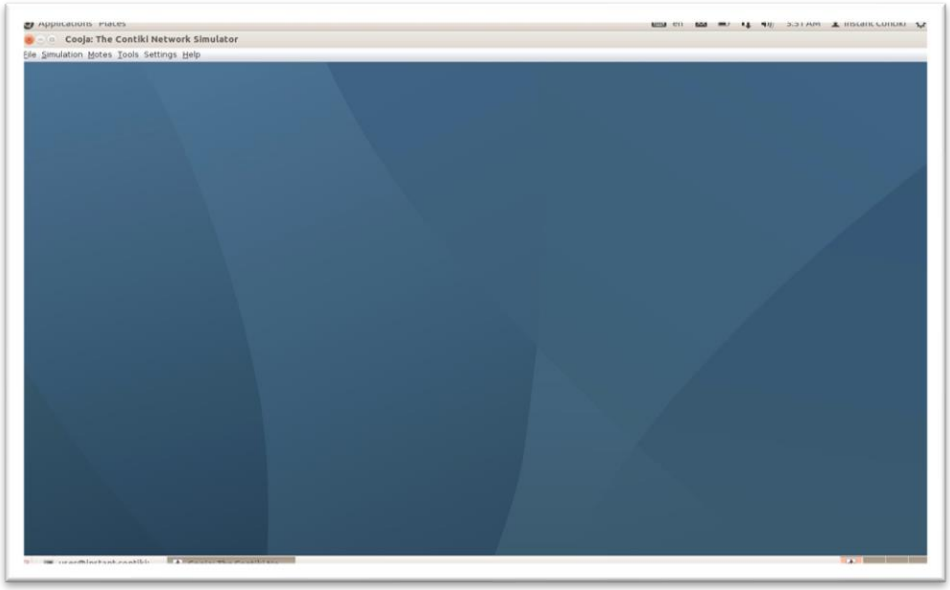


Terminal in Contiki OS – Navigate to Cooja directory



Outputs after executing the *ant run* command

Once the command is successfully executed, it will open a simulator in a new window



Contiki Network Simulator

- With this, we can conclude that we have successfully installed and configured Instant Contiki OS with Cooja.

➢ **Additional Troubleshooting:**

- If your keyboard is not behaving as expected, the keyboard may be using the Swedish layout. You can change this by clicking on the keyboard icon at the top right of the screen.

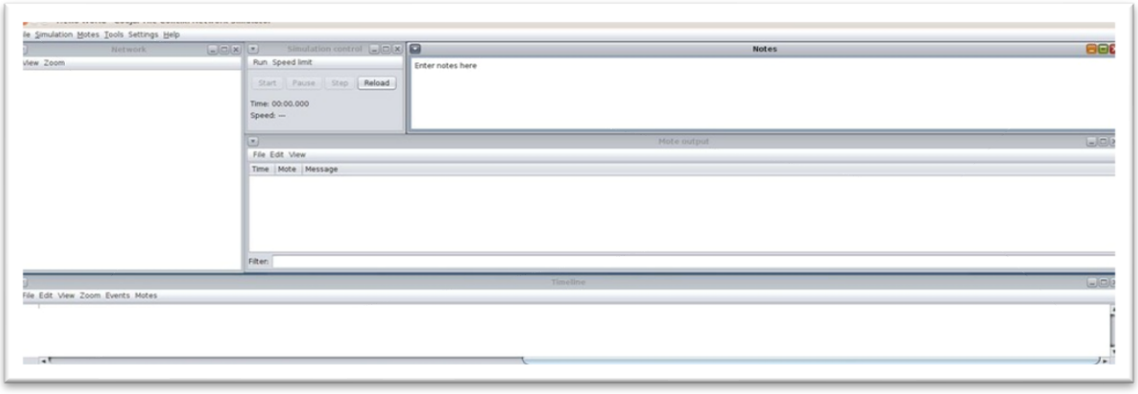
- If your virtual machine screen is stuck at the same resolution install VirtualBox Guest Additions by clicking on the devices menu while in the virtual machine and select "*Insert Guest Additions CD image…*" and follow the instructions. Restart the virtual machine and after it is restarted go to the view menu and make sure full screen mode and scaled mode are unchecked and auto-resize display is checked.

Step 1: Open Cooja simulator.

Step 2: Press Ctrl + N to open up a new simulation tab. Name the simulation *Hello World*. Keep rest of the settings as it is. We will understand about the different Radio Mediums later.

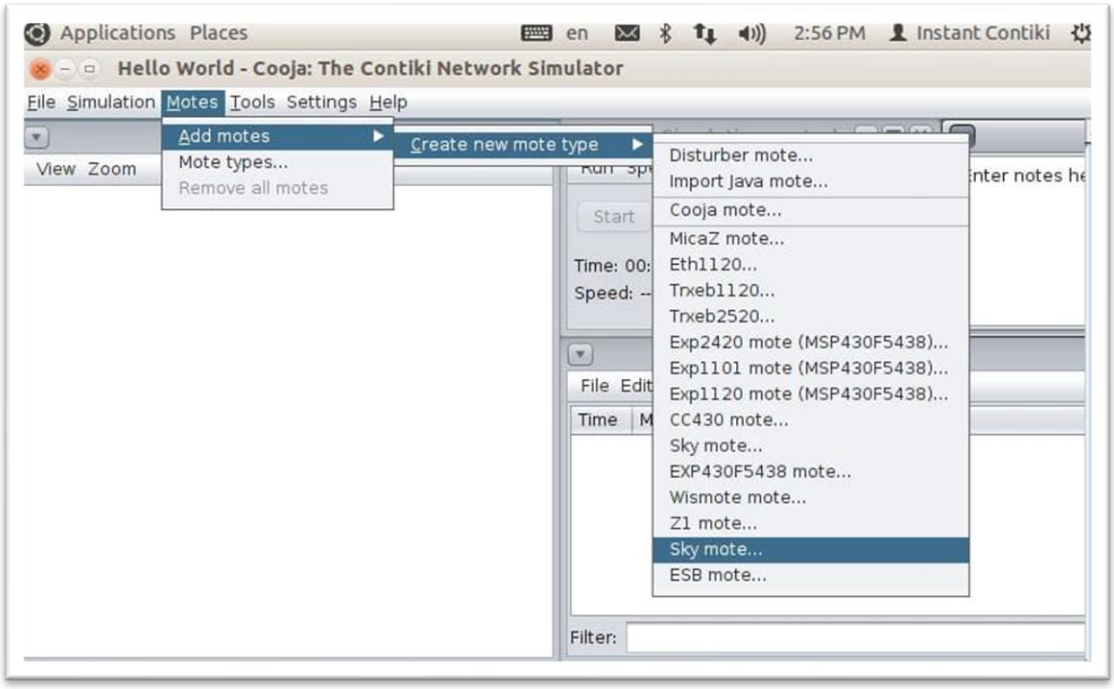
Create new Simulation

Step 3: You will get an output like shown below



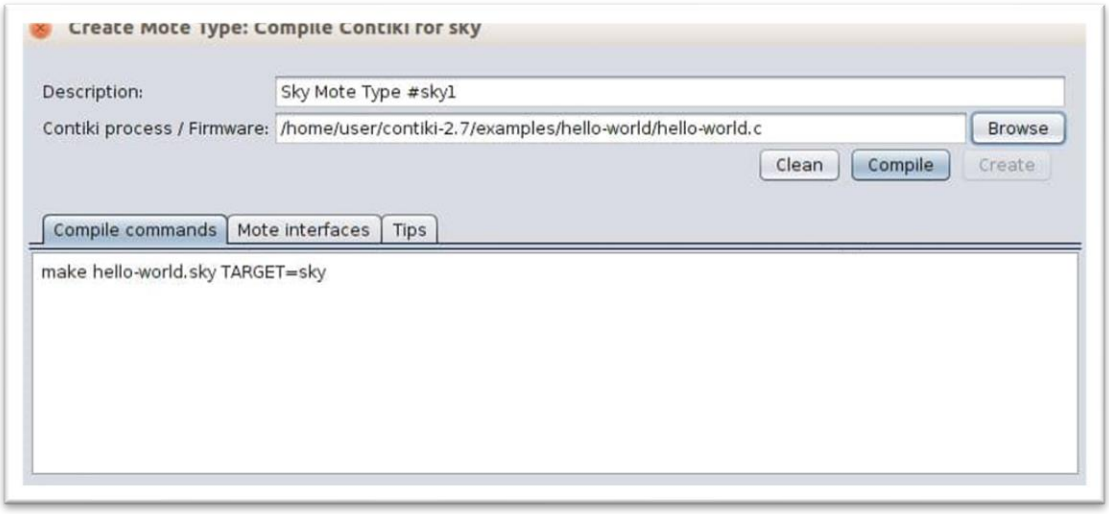
New Simulation Panel

Step 4: Navigate to Motes > Add motes > Sky mote. Sky motes are also known as Tmote Sky or Telos B.



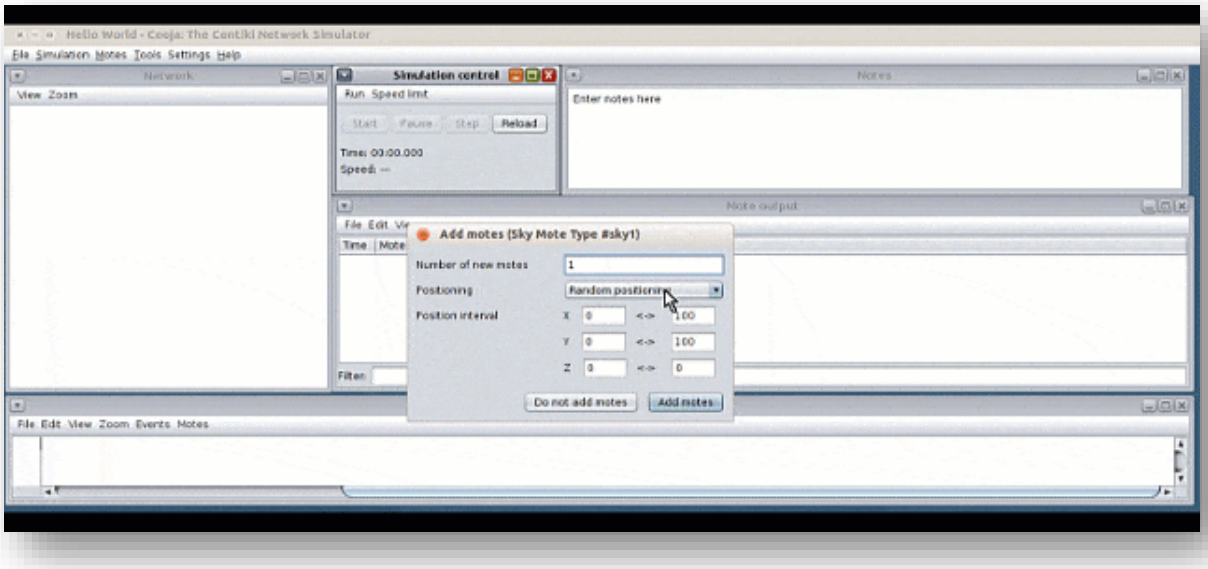
Sky mote

Step 5: Now you will get a new panel as shown below. Click browse and navigate to *contiki-2.7/examples/hello-world/hello-world.c* and press Compile

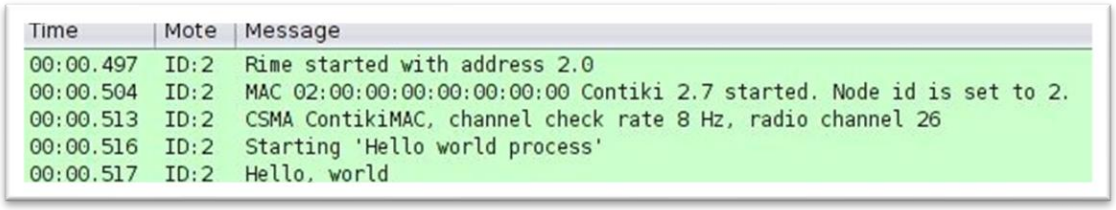


Sky Mote #1

Step 6: After compilation press the Create button. You will get a option to add motes. Add 2 Motes and in Network Box click on View and enable Mote Type and 10m background (Optional) . In simulation control press Start.



Step 7: Check out the output of Mote 2.



Time	Mote	Message
00:00.497	ID:2	Rime started with address 2.0
00:00.504	ID:2	MAC 02:00:00:00:00:00:00:00 Contiki 2.7 started. Node id is set to 2.
00:00.513	ID:2	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.516	ID:2	Starting 'Hello world process'
00:00.517	ID:2	Hello, world

Mote 2 Output

The Mote 2 Output denotes the Main components that: -

Node Id	2
Network Stack	Rime
MAC Protocol	CSMA
Radio Duty Cycle (RDC) Protocol	ContkiMAC
Sleep Cycle	8 Hz
Channel	26
Transmitted Message	Hello World

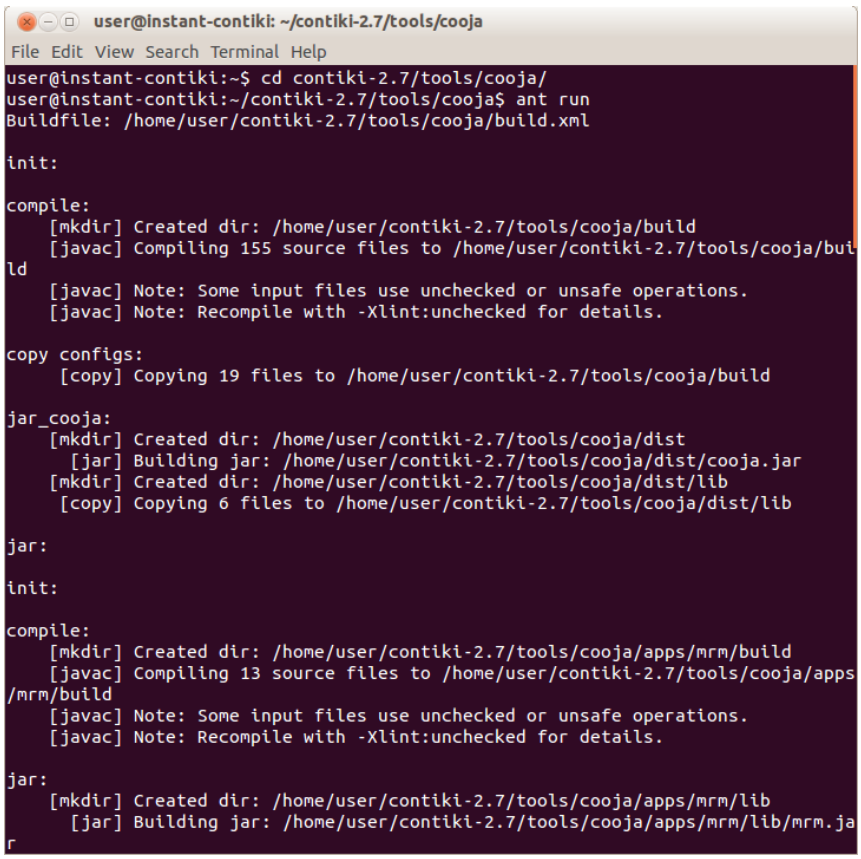
Part 3

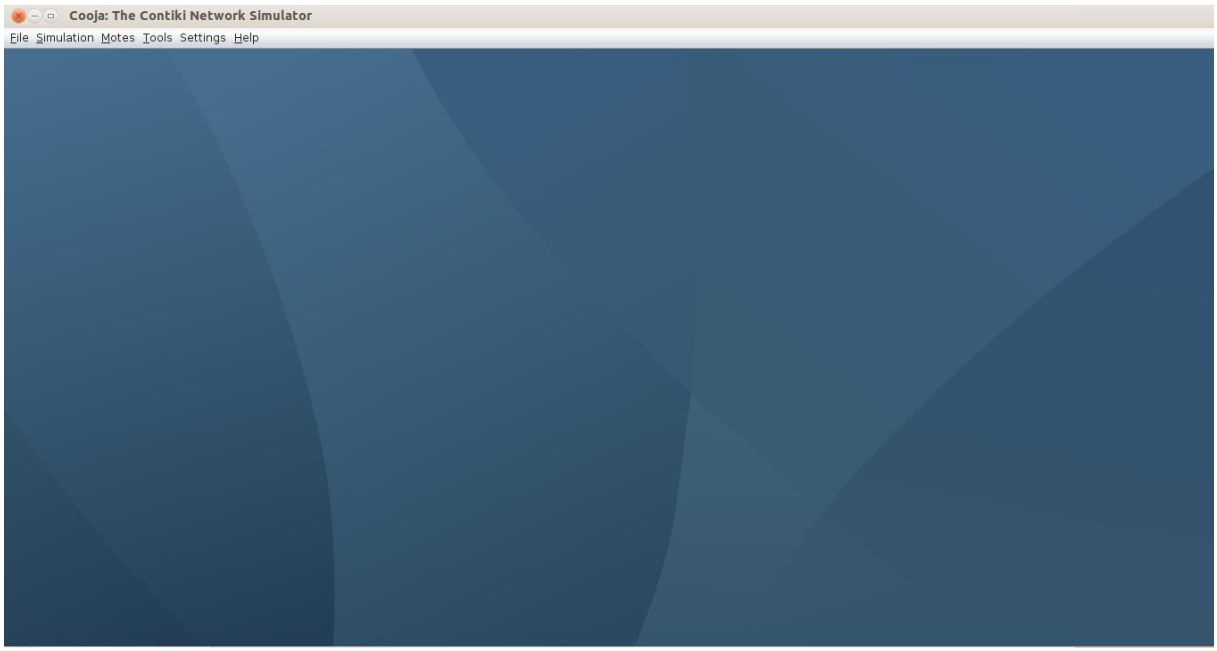
Program 2

Aim: Perform RPL-6LowPAN Demo in Contiki and analyse pcap file.

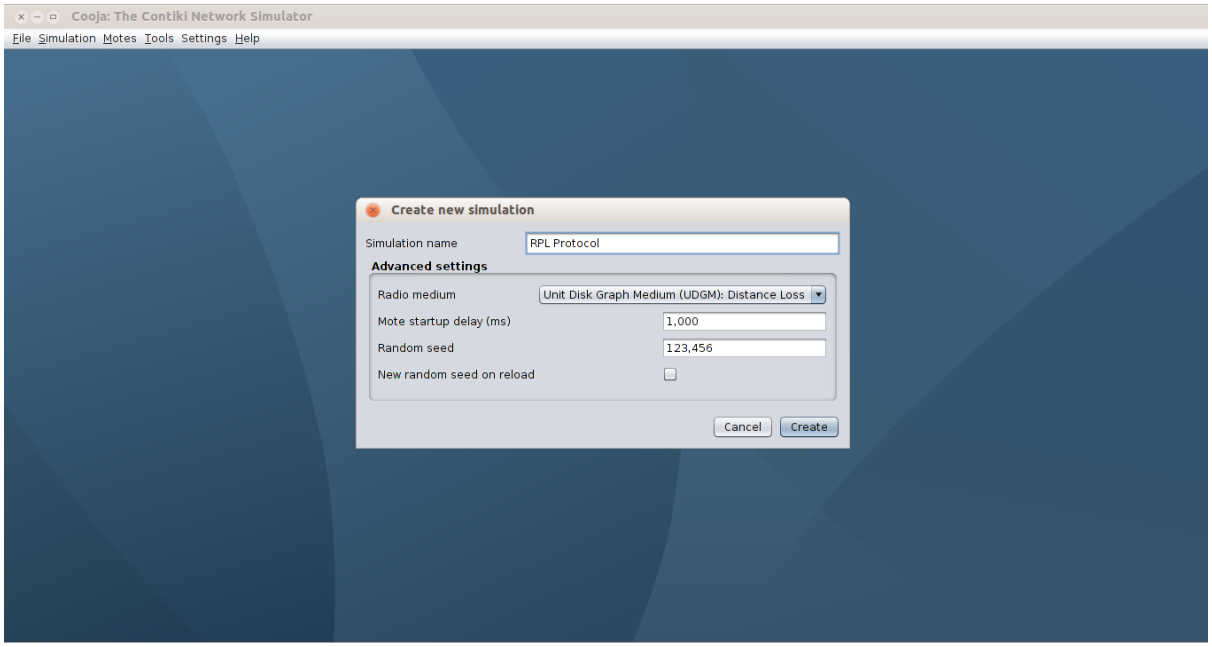
Implementation:

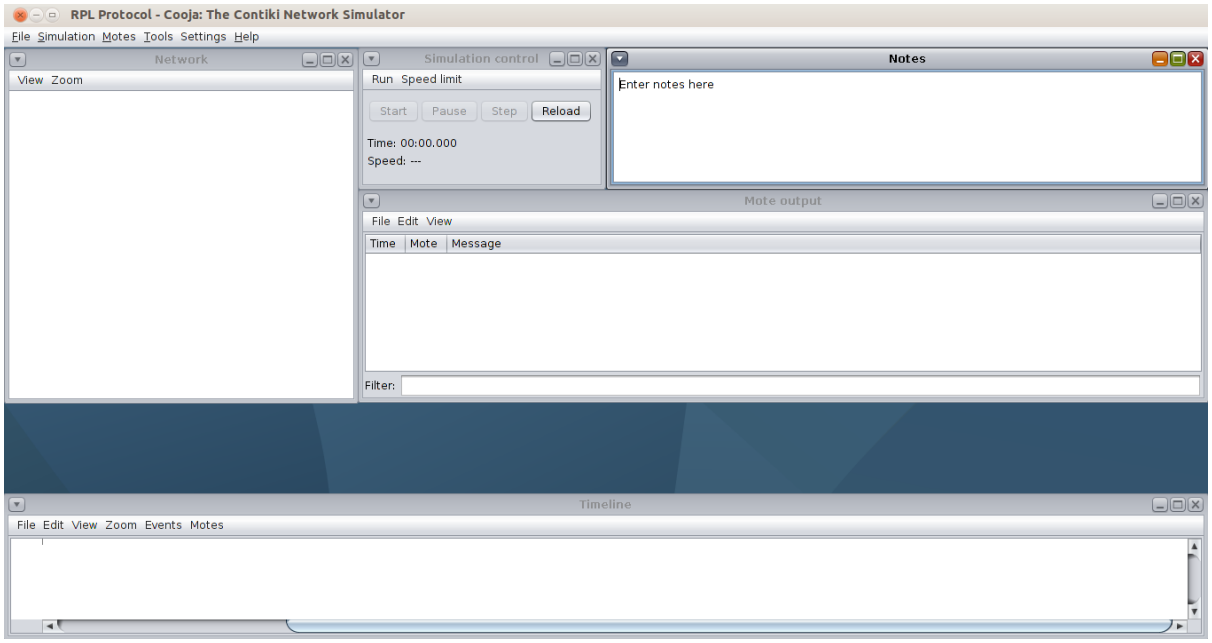
1. **Open the terminal, route to cooja path and write "ant run" to Start the Cooja Simulator.**



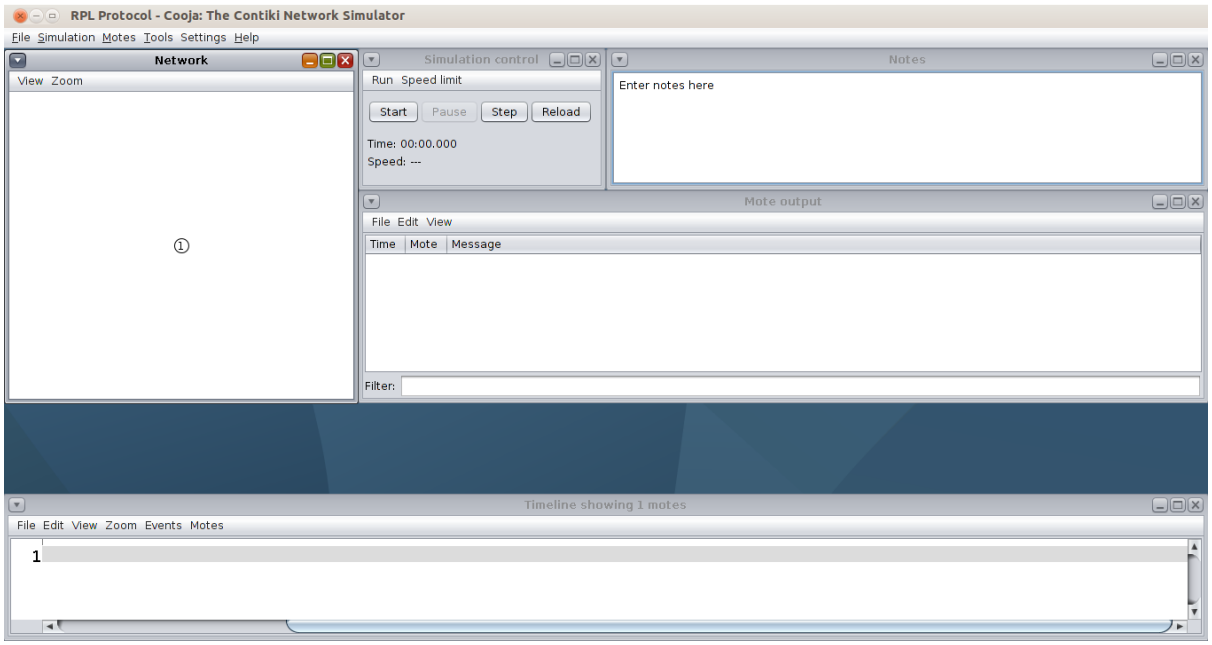


2. **Create New Simulation as shown below.**

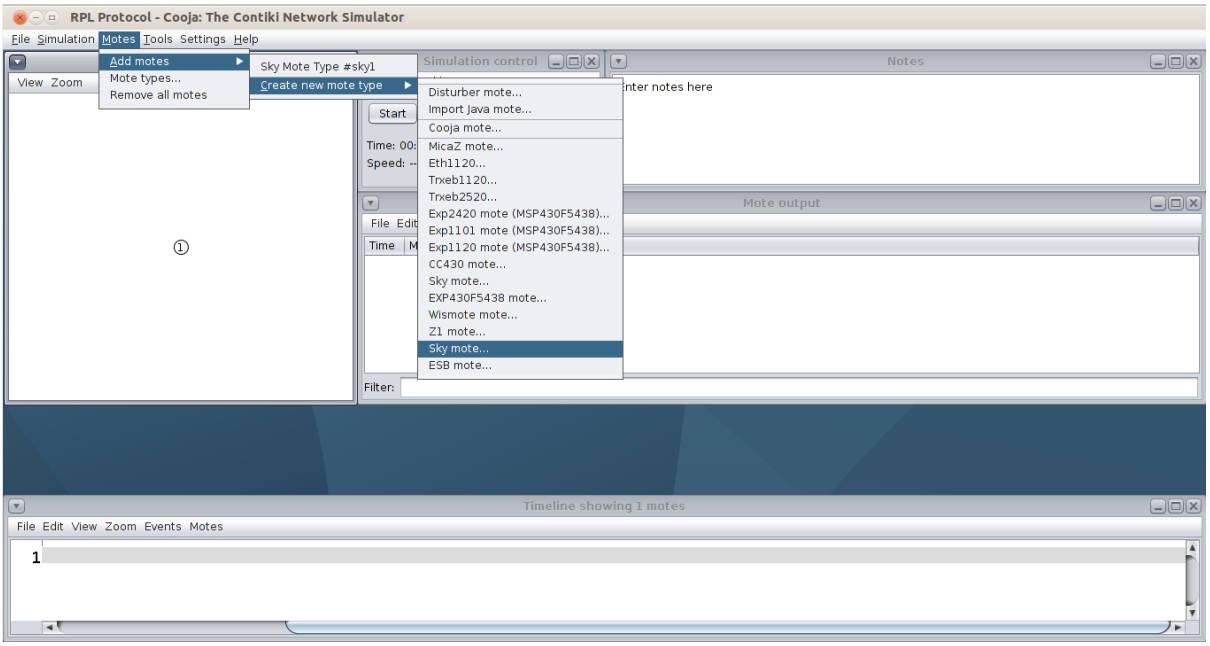

Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Create new simulation
Simulation name
RPL Protocol
Advanced settings
Radio medium
Unit Disk Graph Medium (UDGM): Distance Loss
Mote startup delay (ms)
1,000
Random seed
123,456
New random seed on reload
Cancel
Create

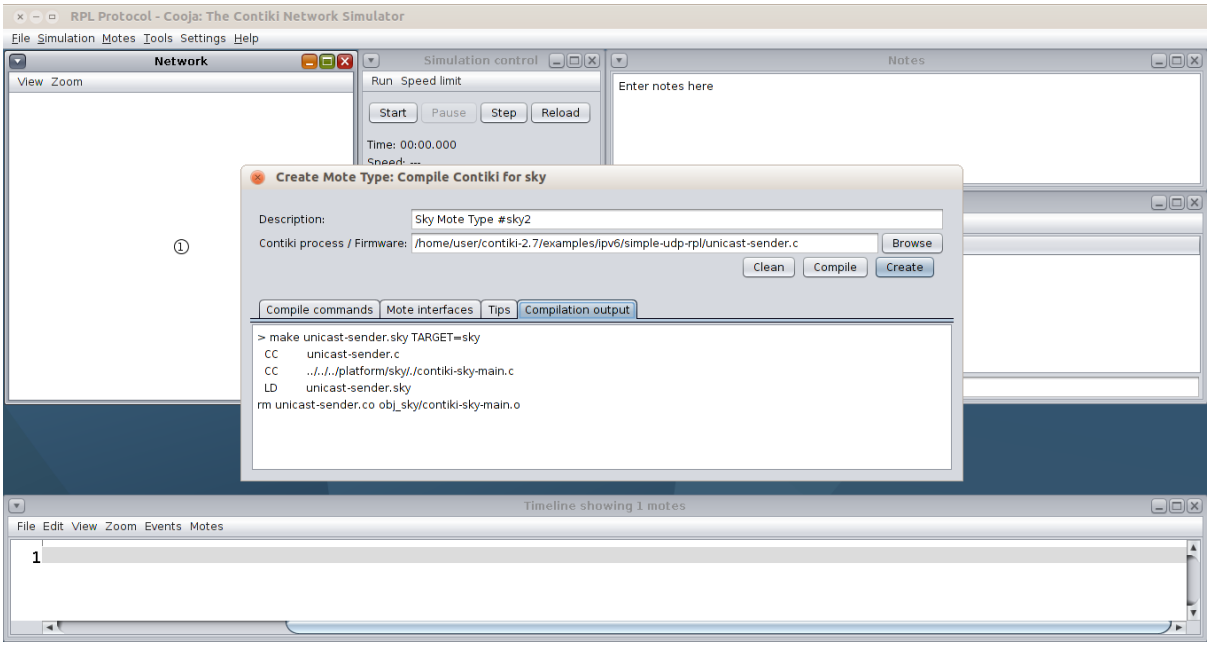

RPL Protocol - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Network
View Zoom
Simulation control
Run Speed limit
Start
Pause
Step
Reload
Time: 00:00.000
Speed: ---
Notes
Enter notes here
Mote output
File Edit View
Time
Mote
Message
Filter:
Timeline
File Edit View Zoom Events Motes

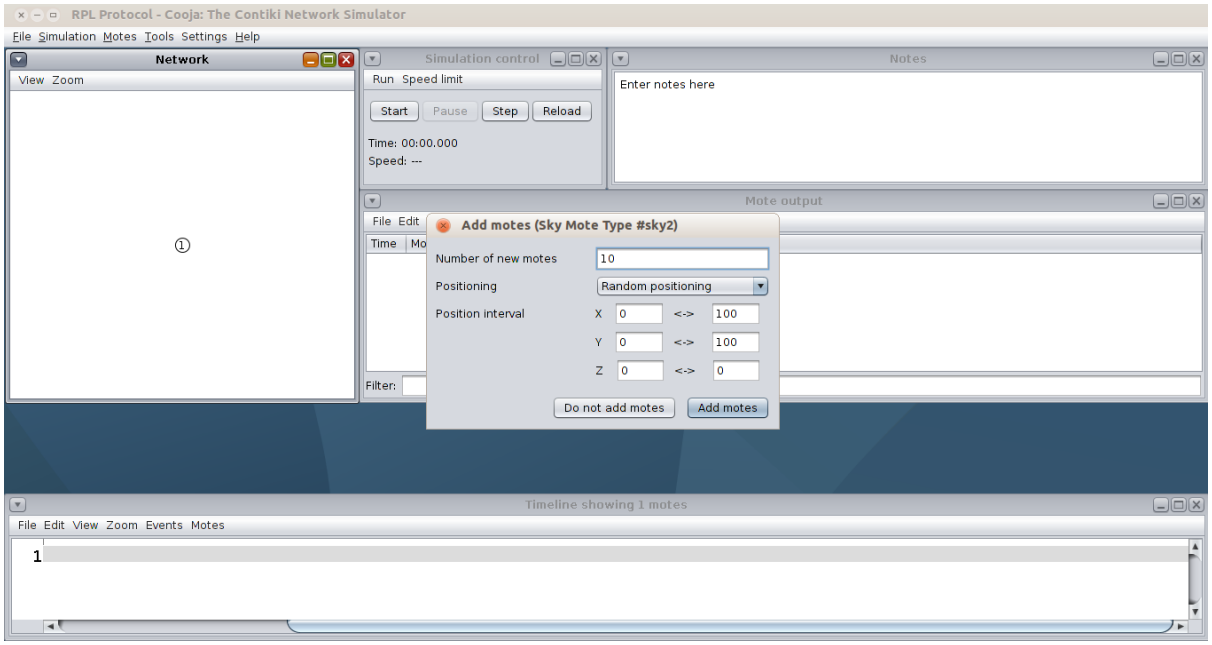
3. **Now add 1 receiver sky mote and for Contiki Process/firmware browse to the "unicast-receiver.c" file path. Then compile it, create it and add 1 mote.**



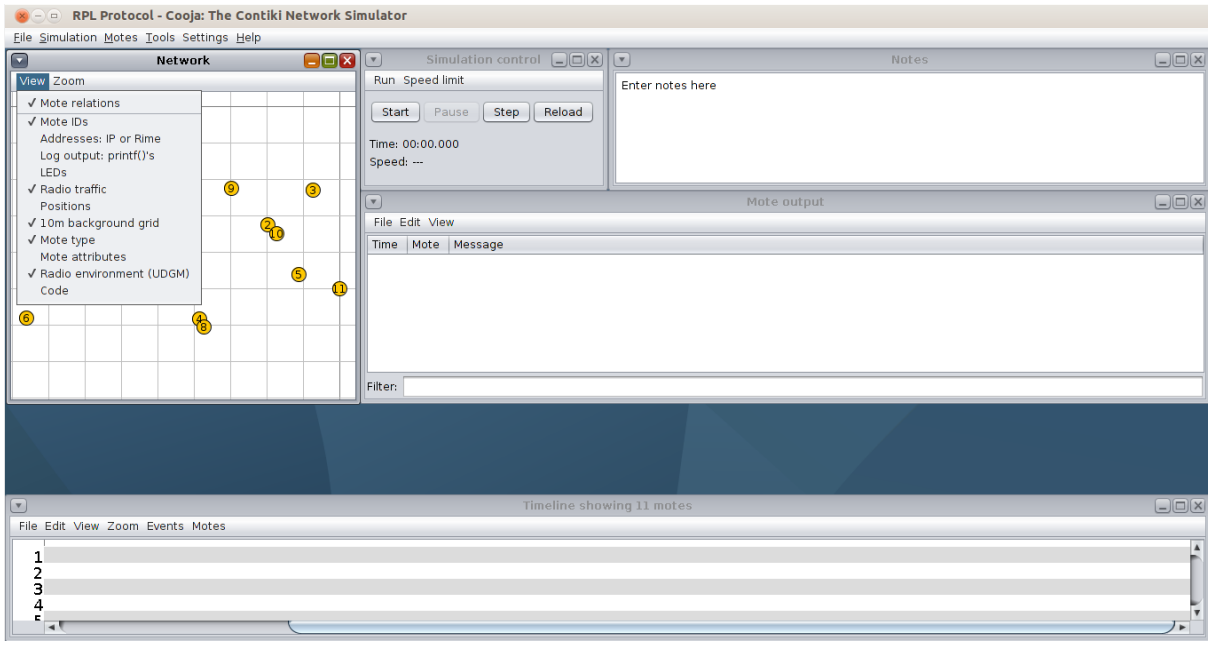
4. **Similarly Add 10 sender sky motes, but now for Contiki Process/firmware browse to the "unicast-sender.c" file path. Then compile it, create it and add 10 motes.**

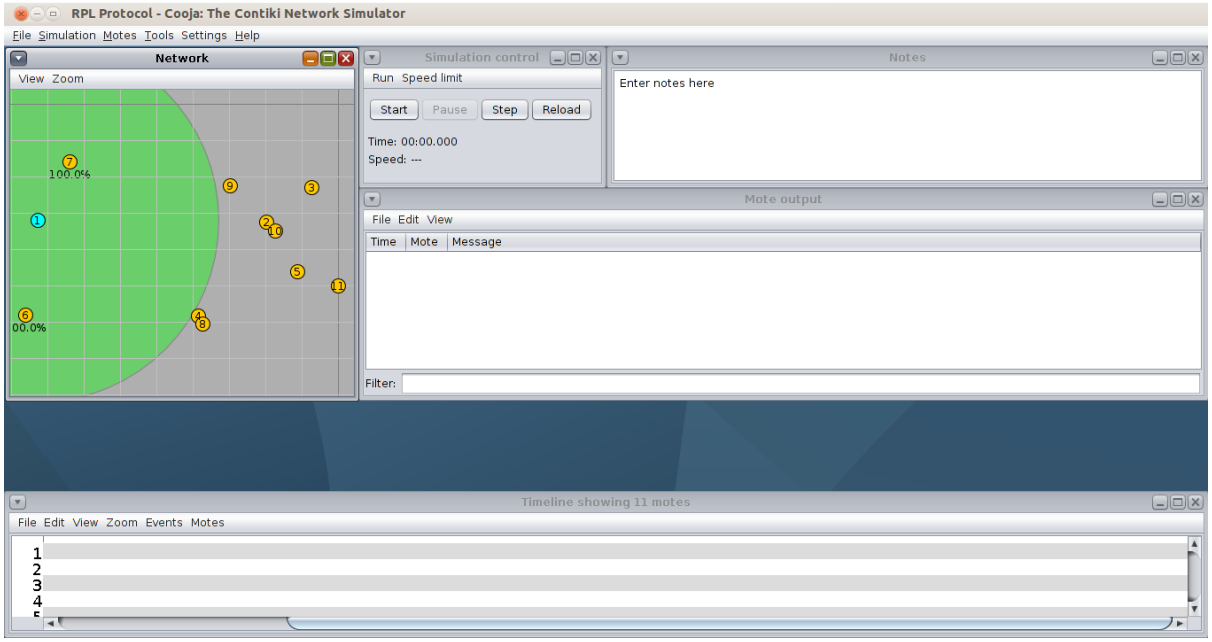




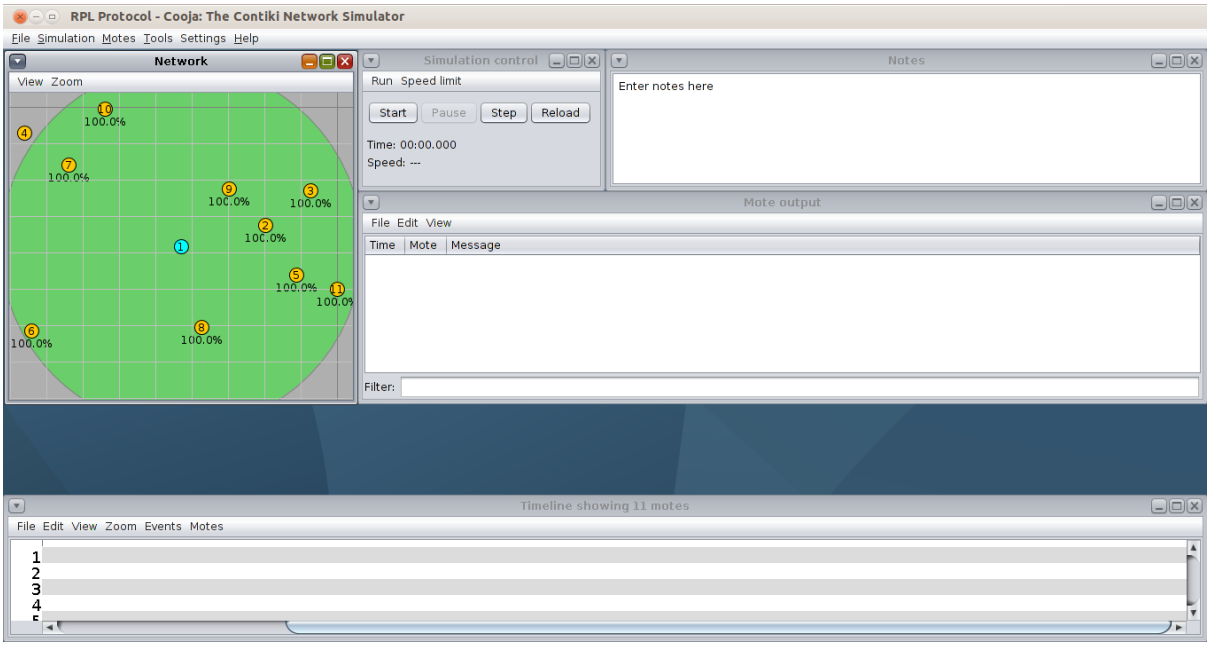


Also Enable some options from “View” menu for detailed and better visualization as shown below.

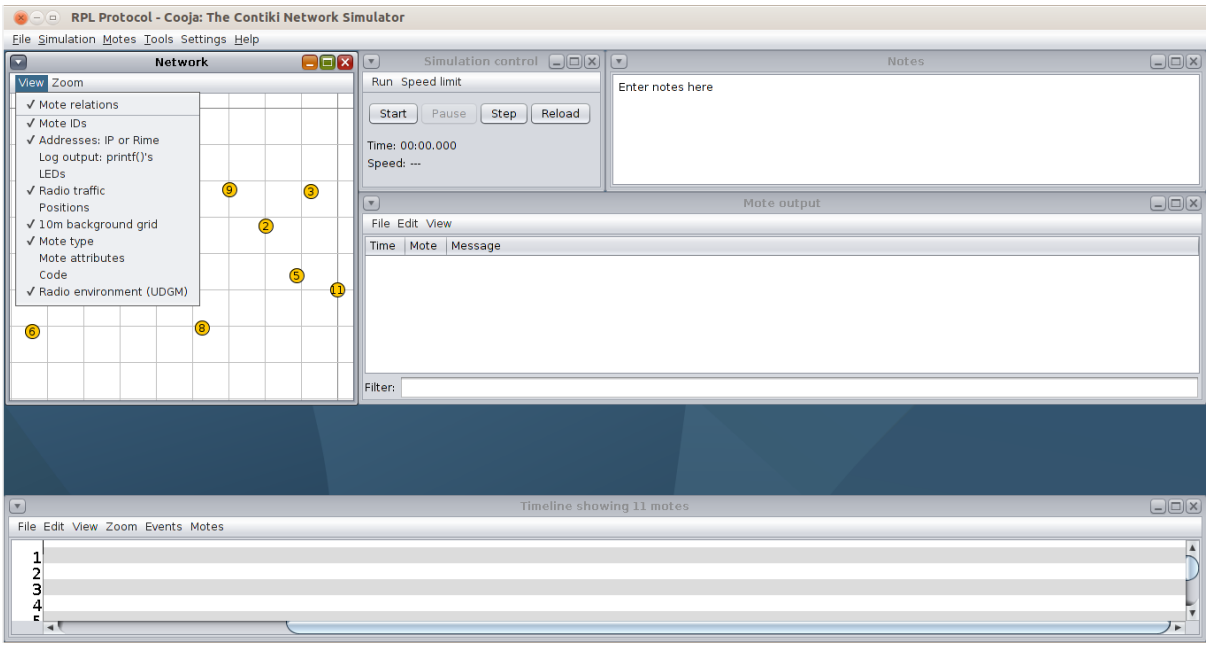




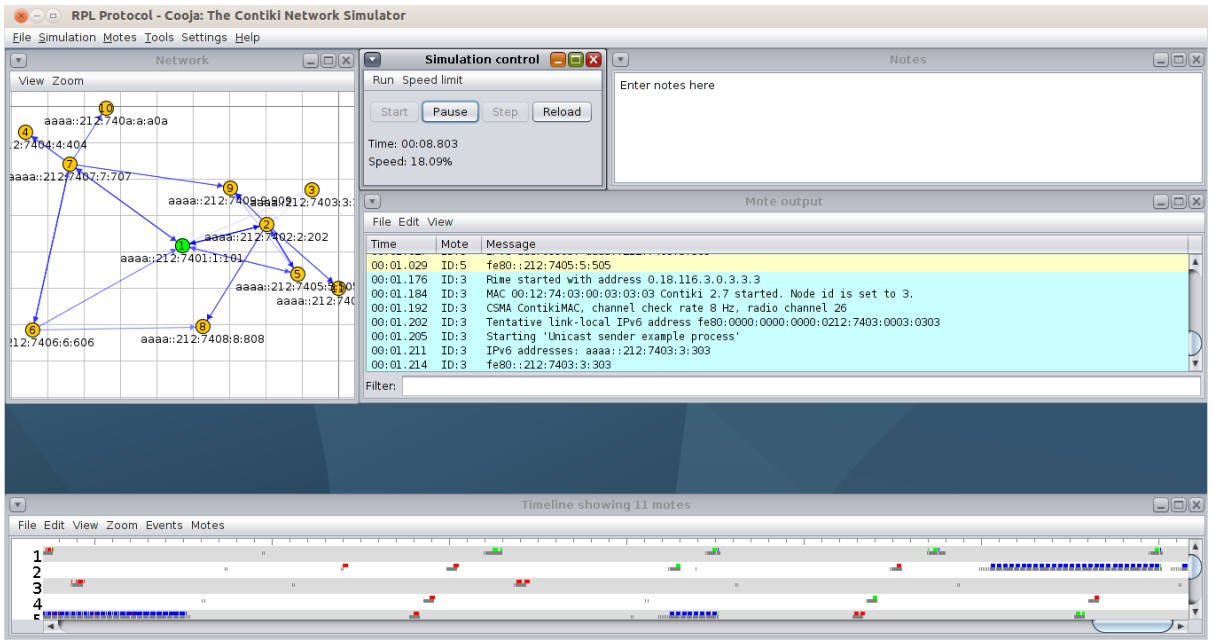
5. Arrange the motes as shown below.



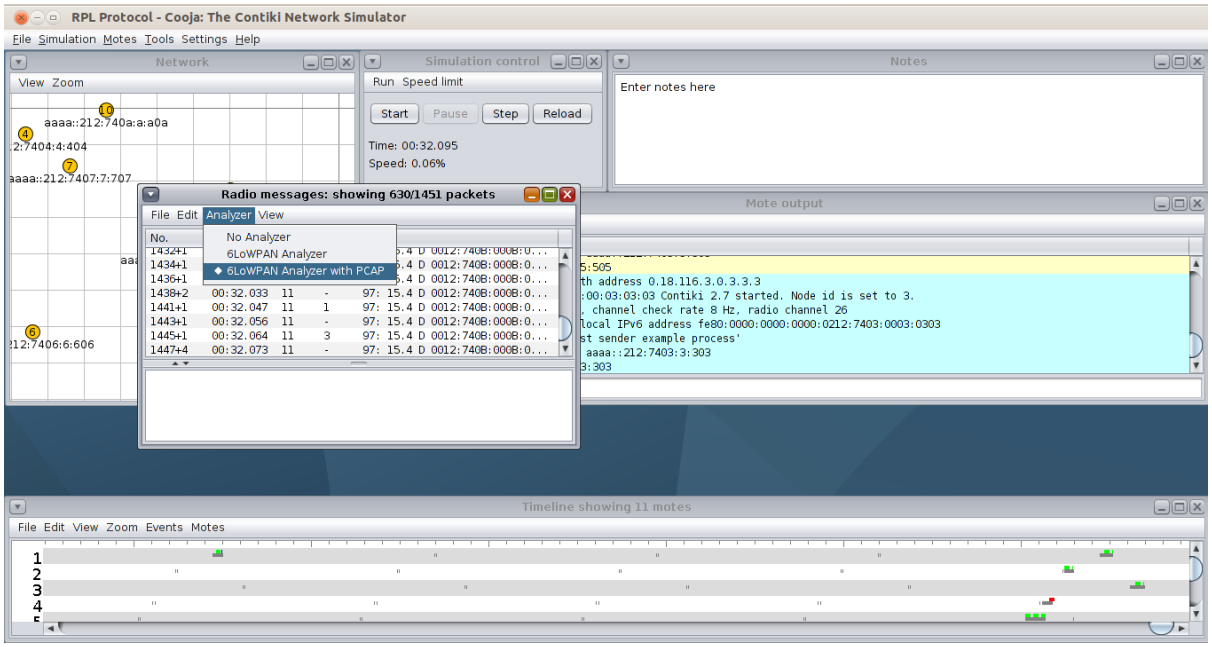
Enable “Addresses” option from “View” menu for detailed visualization as shown below.



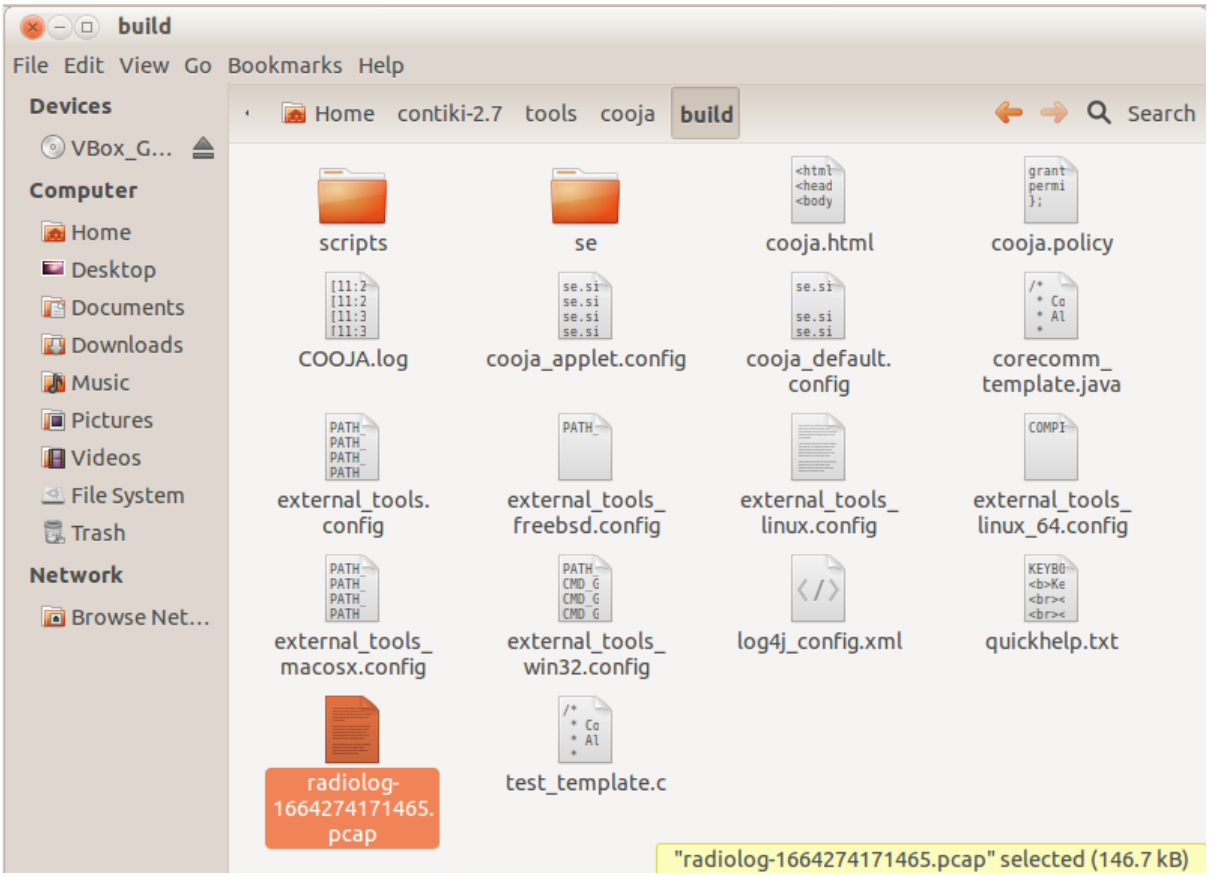
6. Click on “Start” button to Start the simulation.



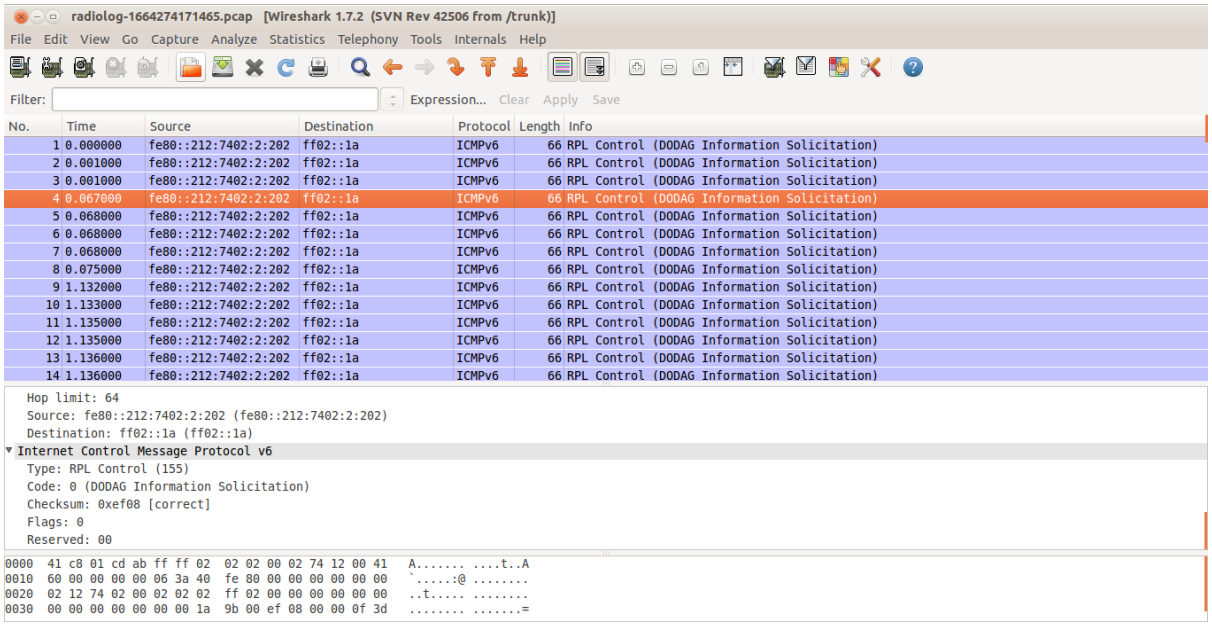
7. **Now click on “Tools” menu, select “Radio Messages” and select “6LowPan Analyzer with PCAP” to analyze packets using wireshark.**



8. **Browse to the "build" directory of cooja tool, we should be able to see the saved PCAP file.**



9. Open Wireshark and open saved PCAP file to analyze.



Part 3

Program 3

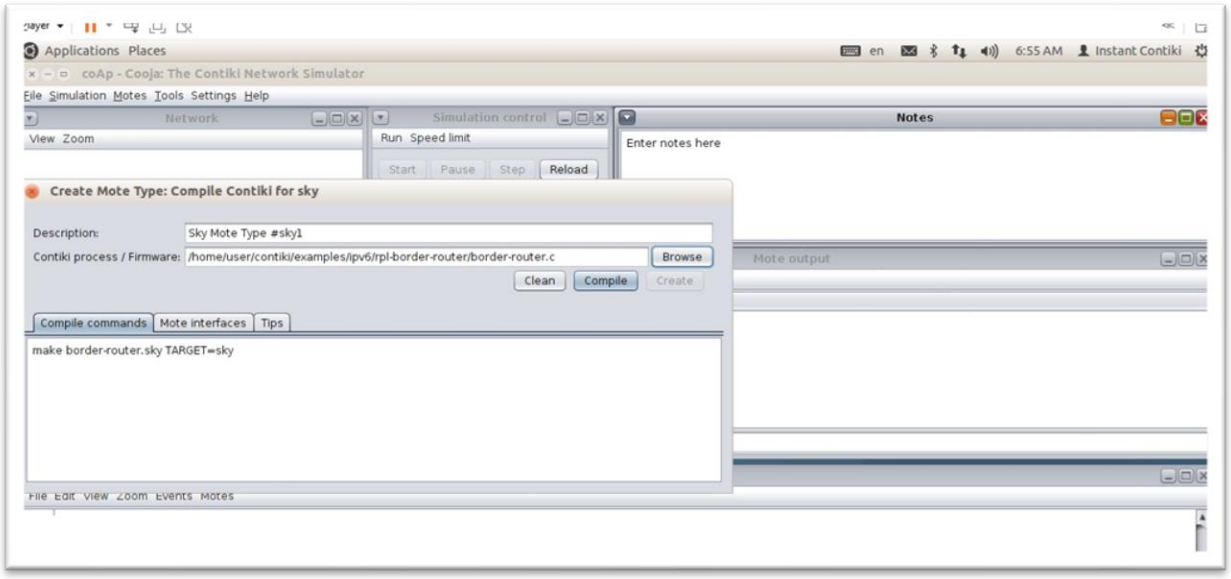
Aim: Perform CoAP Demo in Contiki.

Implementation:

- For this practical, we will add a sky mote
- First of all, we will open Cooja simulator and create new simulator
- Then, we will first add the sky mote

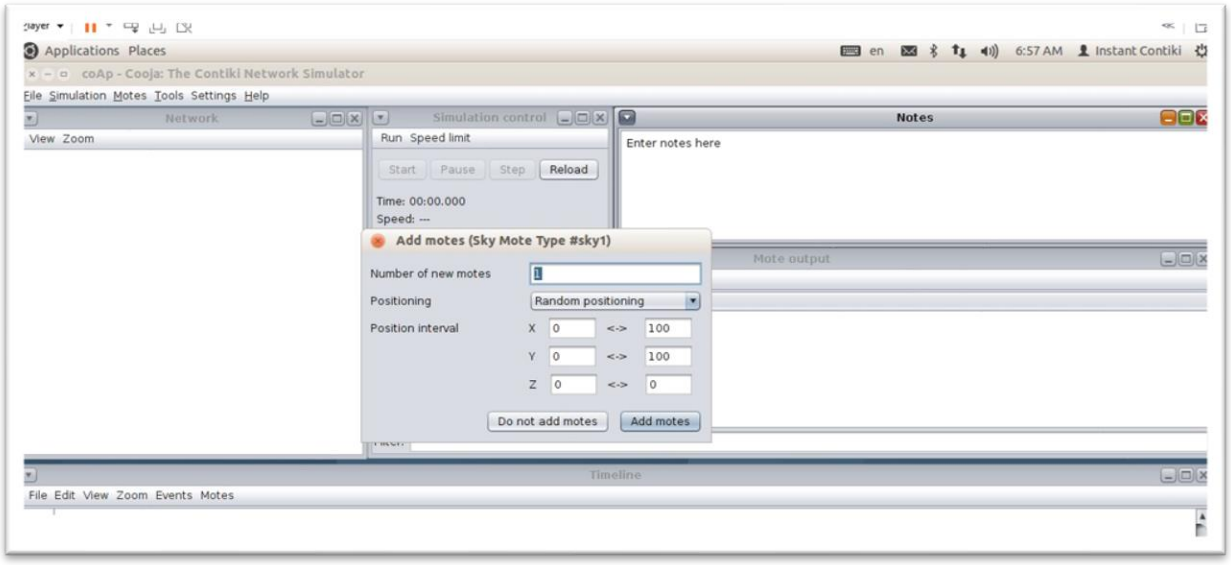
So, we will go to Motes → Create new mote type → Sky mote

- Next, we will browse through example folder to find *example* → *ipv6* → *border-router* → *border-router.c*



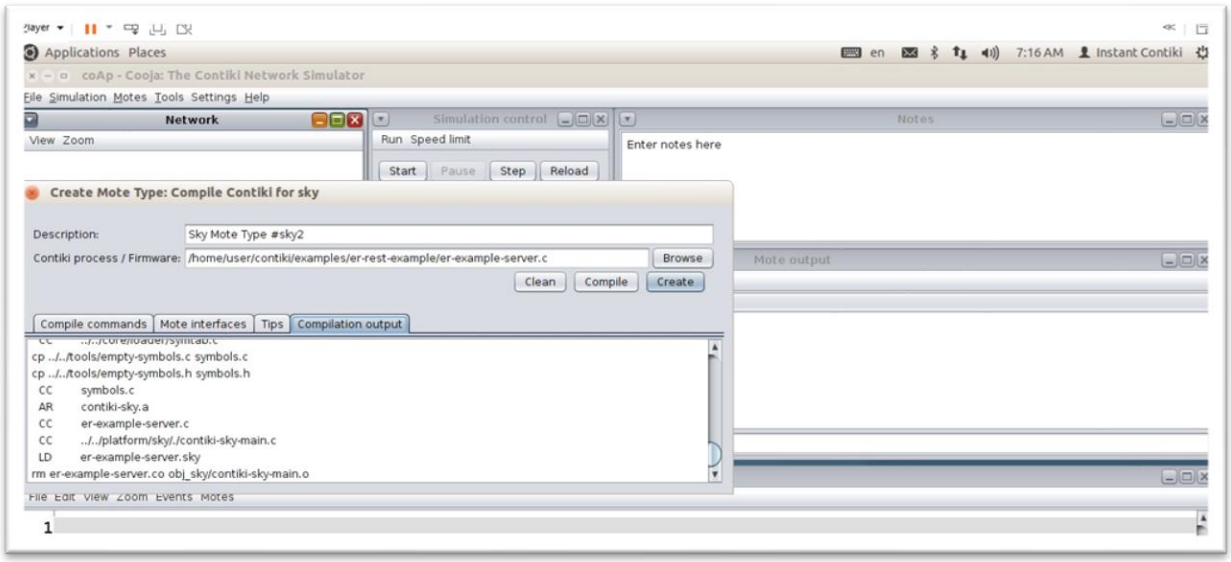
Opening a new simulator & creating a sky mote

- We will compile it and press create
- We will need only 1 mote so number of new motes will be 1 only



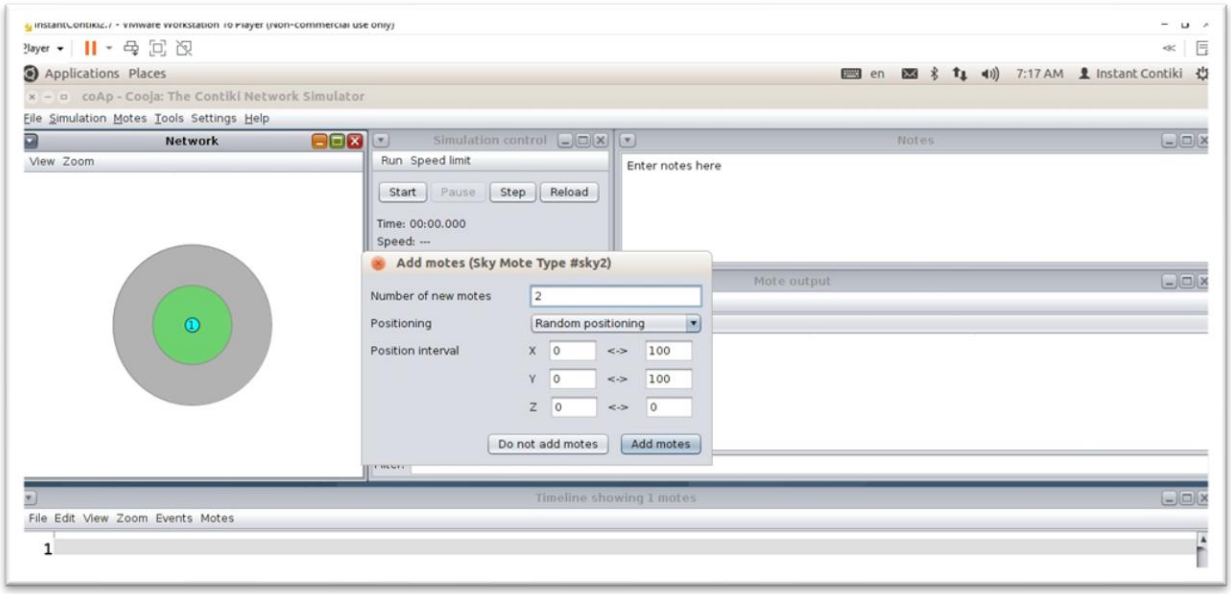
Adding new mote

- Then, we will add the sky mote, so we will go to *Motes* → *Create new mote type* → *Sky mote*
- We will browse through example folder to find *example* → *er-rest-example-server.c*



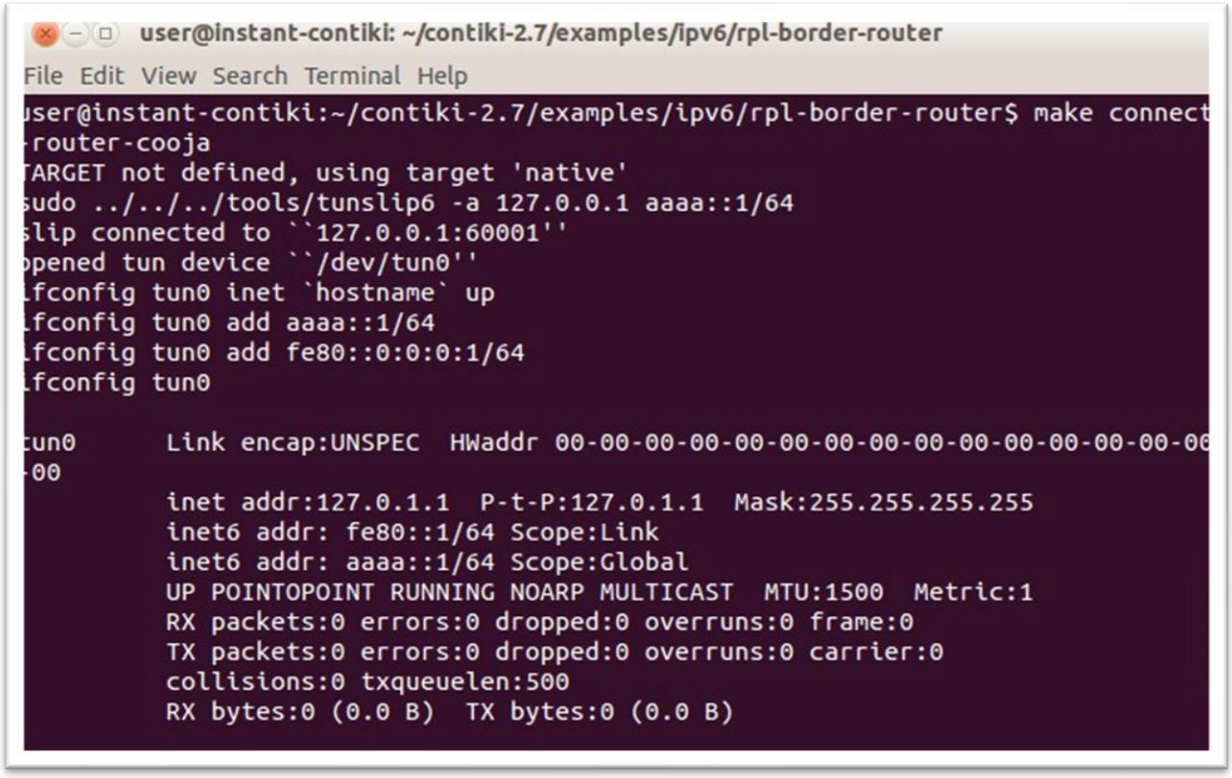
Creating new mote type i.e., sky mote

In total, we would need only 2 motes, so number of new motes will be 2 only



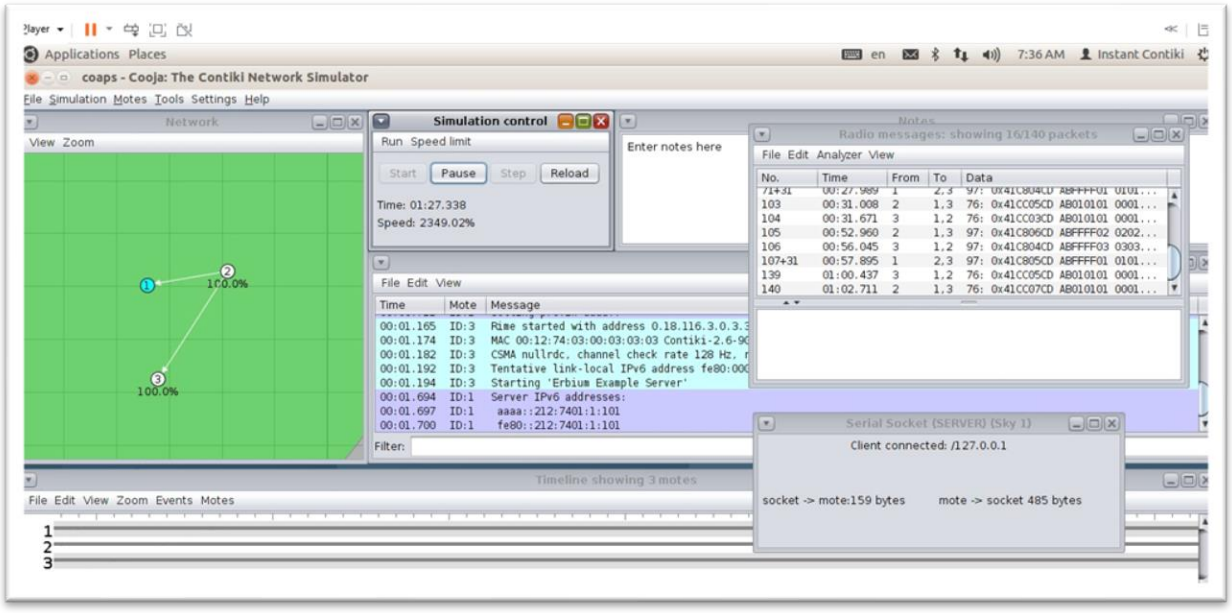
Adding new motes to #sky2

- We will now connect the router with Cooja



Connecting router with Cooja

- When we add new motes, we will be able to see the screen similar to Figure 4.6
- By clicking on a mote, we will also be able to see its range
- We can change the view options as per our convenience
- After this, finally, we can start the simulator



Simulation screen

Part 3

Program 5

Aim: Introduction of Thingspeak for data collection on cloud and show the steps "How to use Thingspeak". Use the sensors data from other public channel, process the data from any one channel and prepare visualization graphs in Thingspeak.

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

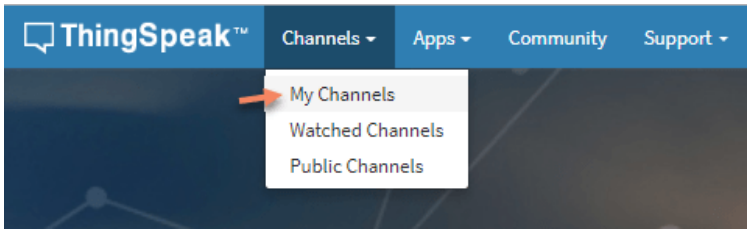
Implementation:

Collect Data in a New Channel

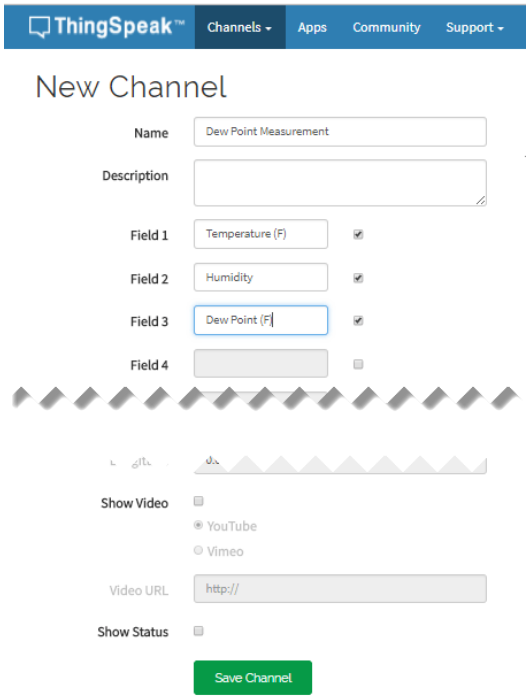
This example shows how to create a new channel to collect analyzed data. You read data from the public ThingSpeak channel 12397 - Weather Station, and write it into your new channel. To learn how to post data to a channel from devices, see see Write Data to Channel and the API Reference.

Create a Channel

1. Sign In to ThingSpeak™ using your MathWorks® Account credentials, or create a new account.
2. Click **Channels** > **MyChannels**.



3. On the Channels page, click **New Channel**.
4. Check the boxes next to Fields 1–3. Enter these channel setting values:
    - **Name**: Dew Point Measurement
    - **Field 1**: Temperature (F)
    - **Field 2**: Humidity
    - **Field 3**: Dew Point



5. Click **Save Channel** at the bottom of the settings.

   You now see these tabs:

   - **Private View**: This tab displays information about your channel that only you can see.
   - **Public View**: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.
   - **Channel Settings**: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.
   - **Sharing**: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.
   - **API Keys**: This tab displays your channel API keys. Use the keys to read from and write to your channel.
   - **Data Import/Export**: This tab enables you to import and export channel data.

Next Steps

Your channel is available for future use by clicking **Channels** > **My Channels**.

Analyze Your Data

This example shows how to read temperature and humidity data from ThingSpeak channel 12397, which collects weather-related data from an Arduino® device. You write the temperature and humidity data into your Dew Point Measurement channel, along with the calculated dew point data. Then use ThingSpeak™ to visualize the results on your channel.

Prerequisite Steps

This example requires that you have already performed these steps:

- Sign In either to your MathWorks® Account or ThingSpeak account, or create a new MathWorks account.
- Create a Channel as your Dew Point Measurement channel.

Read Data from a Channel

Read the humidity and temperature from the public WeatherStation channel Fields 3 and 4, and write that data to Fields 2 and 1, respectively, of your Dew Point Measurement channel. Dew point is calculated and written to Field 3.

Use MATLAB® Analysis to read, calculate, and write your data.

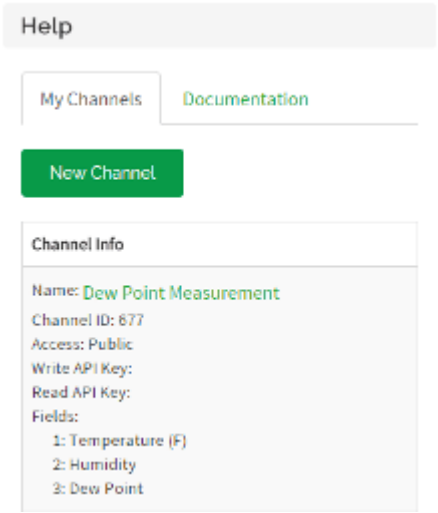
1. Go to the **Apps** tab, and click **MATLAB Analysis**.
2. Click **New**. Select the **Custom** template, and click **Create**.
3. In the **Name** field, enter Dew Point Calculation.
4. In the MATLAB Code field, enter the following lines of code.
    a. Save the public Weather Station channel ID and your Dew Point Measurement channel ID to variables.
    b. readChId = 12397;

writeChId = 671; % replace with your channel number

    c. Save your Write API Key to a variable.

writeKey = 'F6CSCVKX42WFZN9Y'; % Replace with your channel write key

To find your Channel ID and Write API Key, refer to Channel Info on the **My Channels** tab.



    d. Read the latest 20 points of temperature data with timestamps and humidity data from the public Weather Station channel into variables.
    e. [temp,time] = thingSpeakRead(readChId,'Fields',4,'NumPoints',20);

humidity = thingSpeakRead(readChId,'Fields',3,'NumPoints',20);

Calculate the Dew Point

Add the following MATLAB code to calculate the dew point using temperature and humidity readings:

1. Convert the temperature from Fahrenheit to Celsius.

tempC = (5/9)*(temp-32);

2. Specify the constants for water vapor (b) and barometric pressure (c).
3. b = 17.62;

c = 243.5;

4. Calculate the dew point in Celsius.
5. gamma = log(humidity/100) + b*tempC./(c+tempC);

dewPoint = c*gamma./(b-gamma)

6. Convert the result back to Fahrenheit.

dewPointF = (dewPoint*1.8) + 32;

7. Write data to your Dew Point Measurement channel. This code posts all the available data in one operation and includes the correct timestamps.
8. thingSpeakWrite(writeChId,[temp,humidity,dewPointF],'Fields',[1,2,3],...

'TimeStamps',time,'Writekey',writeKey);

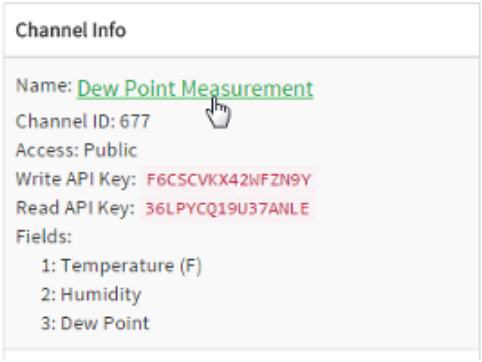
The full block of code now appears as:

```
% Enter your MATLAB Code below
readChId = 12397;
writeChId = ZZZZZ;          % Replace with your channel number
writeKey = 'XXXXXXXXXXXXXXXX';  % Replace with your channel write key
[temp,time] = thingSpeakRead(readChId,'Fields',4,'NumPoints',20);
humidity = thingSpeakRead(readChId,'Fields',3,'NumPoints',20);
tempC = (5/9)*(temp-32);
b = 17.62;
c = 243.5;
gamma = log(humidity/100) + b*tempC./(c+tempC);
dewPoint = c*gamma./(b-gamma)
dewPointF = (dewPoint*1.8) + 32;
thingSpeakWrite(writeChId,[temp,humidity,dewPointF],'Fields',[1,2,3],...
'TimeStamps',time,'Writekey',writeKey);
```

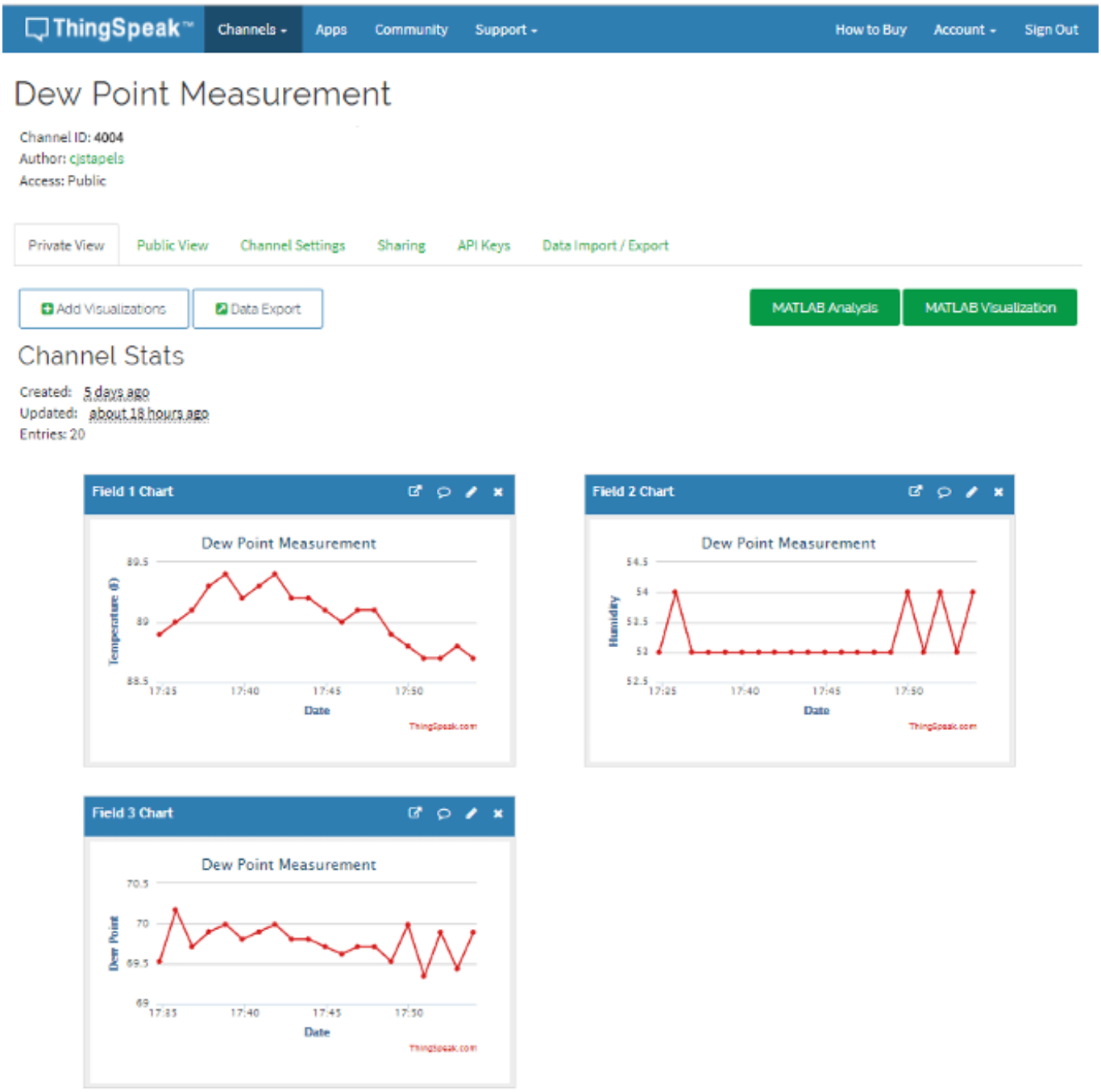
9. Click **Save and Run** to validate and process your code.

Any errors in the code are indicated in the **Output** field.

10. To see if your code ran successfully, click your **Dew Point Measurement** channel link in the **Channel Info** panel.



The Dew Point Measurement channel now shows charts with channel data from each Field.

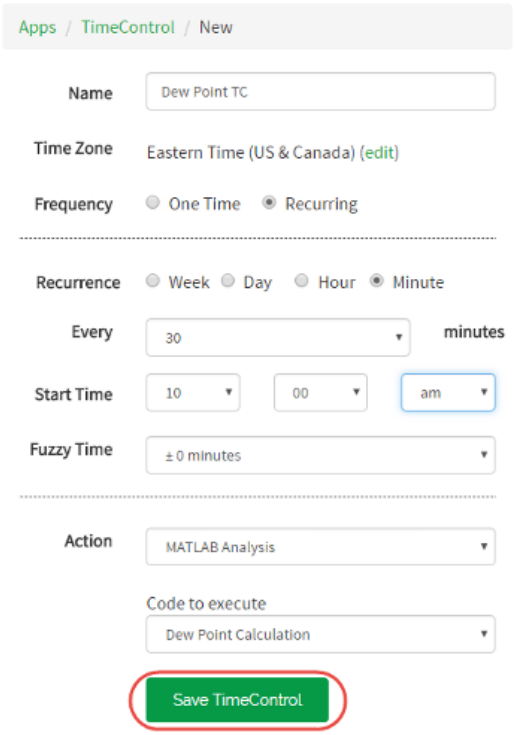


Schedule Code

Use the TimeControl app to schedule the dew point calculation in your MATLAB Analysis code. Schedule it to read data from the weather station every 30 minutes and calculate the dew point.

1. Scroll to the bottom of your MATLAB Analysis Dew Point Calculation page. Click **TimeControl** to open the app with MATLAB Analysis preselected in the **Actions** field and the Dew Point Calculation as the **Code to execute**.
2. Name your new TimeControl Dew Point TC

3. Choose **Recurring** in the **Frequency** field.
4. Choose **Minute** in the **Recurrence** field.
5. Select **30** in the **Every — minutes** field.
6. Keep the **Start Time** at the default value.
7. Verify that the **Action** is MATLAB Analysis, and the **Code to execute** is your Dew Point Calculation.
8. Click **Save TimeControl**



Note

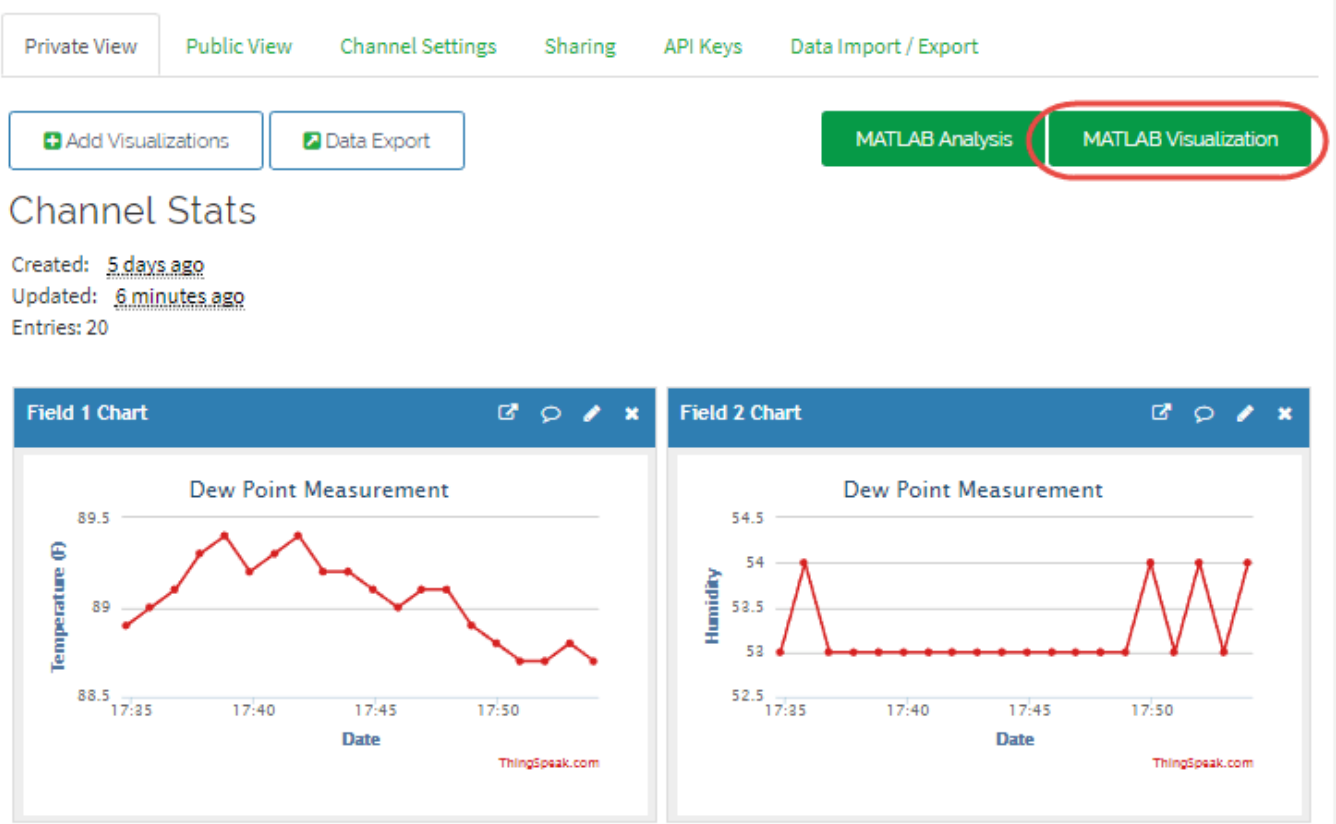
Setting up a TimeControl to write data to your channel uses available messages on your ThingSpeak account. This action can eventually exhaust available messages, which results in rejection of channel feed updates. Make sure that the data you write to a channel does not overlap in the time domain as it causes unnecessary use of messages.

Visualize Dew Point Measurement

Use the MATLAB Visualizations app to visualize the measured dew point data, temperature, and humidity from your Dew Point Measurement channel. This example uses the plot (MATLAB) function to show all three data points in a single visualization.

Go to **Apps** > **MATLAB Visualizations**, and click **New** to create a visualization.

Alternately, you can click **MATLAB Visualization** in your Dew Point Measurement channel view.



1. Select the **Custom** template, and click **Create**.
2. Name the visualization "Dew Point."
3. Create variables for your Dew Point Measurement channel ID and your Read API Key. Replace the values in the code with your channel ID and Read API Key.
4. readChId = ZZZZ

readKey = 'XXXXXXXXXXXXXXXX';

5. Read data from your channel fields, and get the last 100 points of data for:
    - Temperature: from Field 1
    - Humidity: from Field 2
    - Dew point: from Field 3
    - [dewPointData,timeStamps] = thingSpeakRead(readChId,'fields',[1,2,3],...

'NumPoints',100,'ReadKey',readKey);

6. Plot the data with x and y labels, a title, and a legend.
7. plot(timeStamps,dewPointData);
8. xlabel('TimeStamps');
9. ylabel('Measured Values');
10. title('Dew Point Measurement');
11. legend({'Temperature','Humidity','Dew Point'});

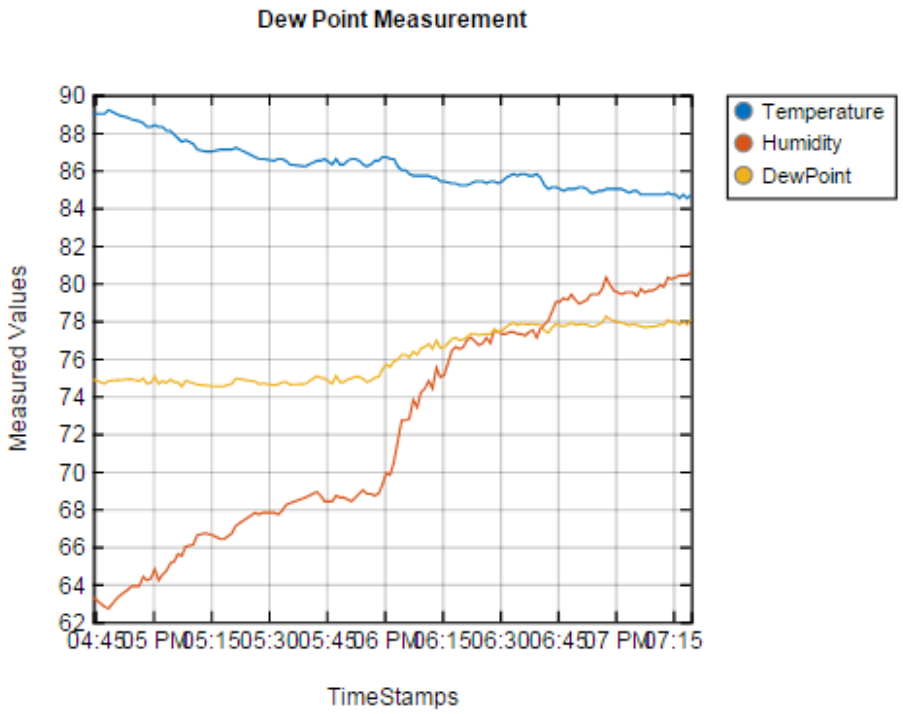
grid on;

Your code will look similar to this code:

```
% Enter your MATLAB code below

readChId = ZZZZZ            % Your Channel ID

readKey = 'XXXXXXXXXXXXXXXX' %Your Read API Key

[dewPointData,timeStamps] = thingSpeakRead(readChId,'fields',[1,2,3],...

   'NumPoints',100,'ReadKey',readKey);


plot(timeStamps,dewPointData);

xlabel('TimeStamps');

ylabel('Measured Values');

title('Dew Point Measurement');

legend({'Temperature','Humidity','Dew Point'});

grid on;
```

12. Click **Save and Run**. If your MATLAB code has no errors, the plot output looks similar to the plot shown here:

MATLAB Plot Output

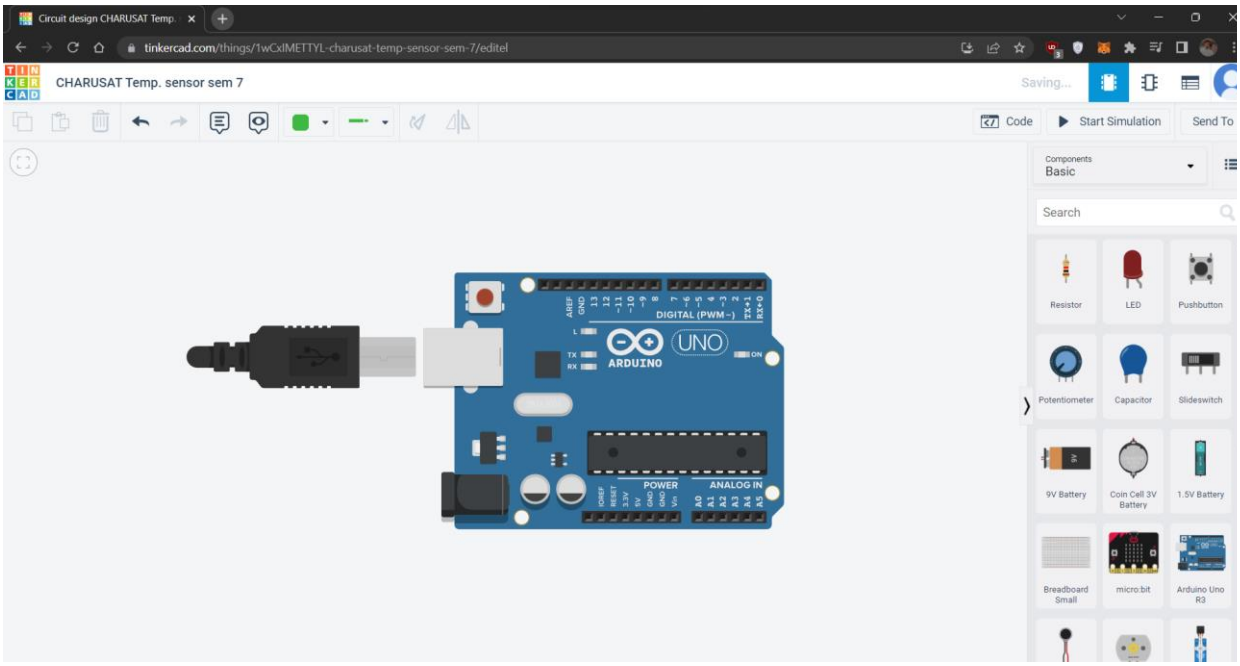


Part 3

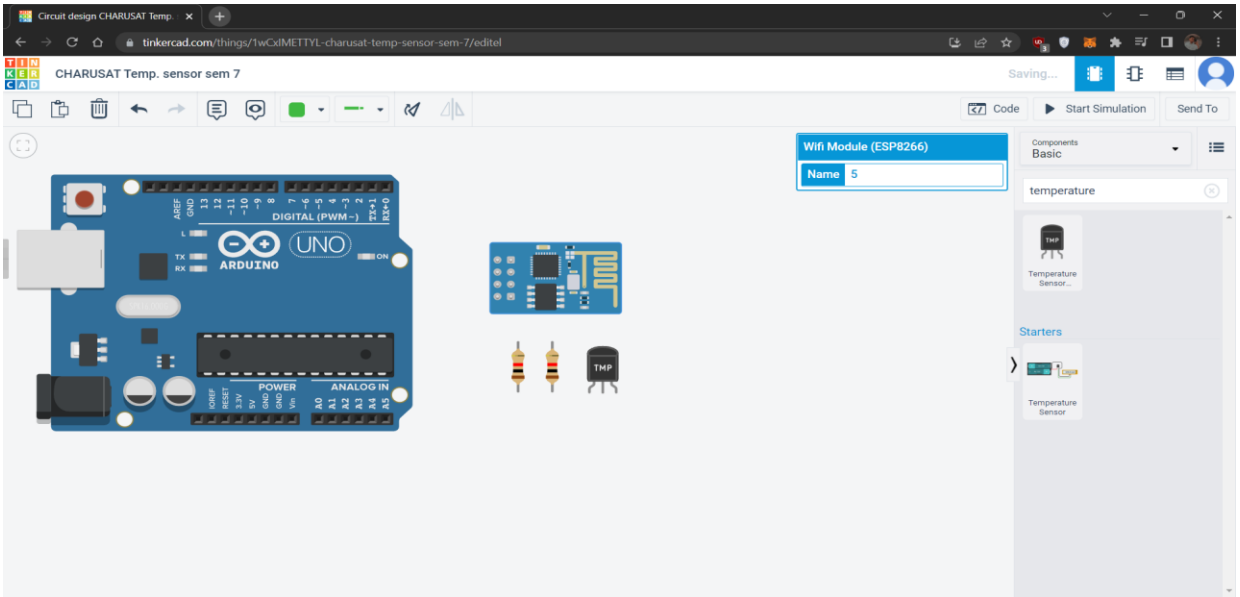
Program 6

AIM: Track and visualize temperature data using Tinkar cad and Thingspeak.

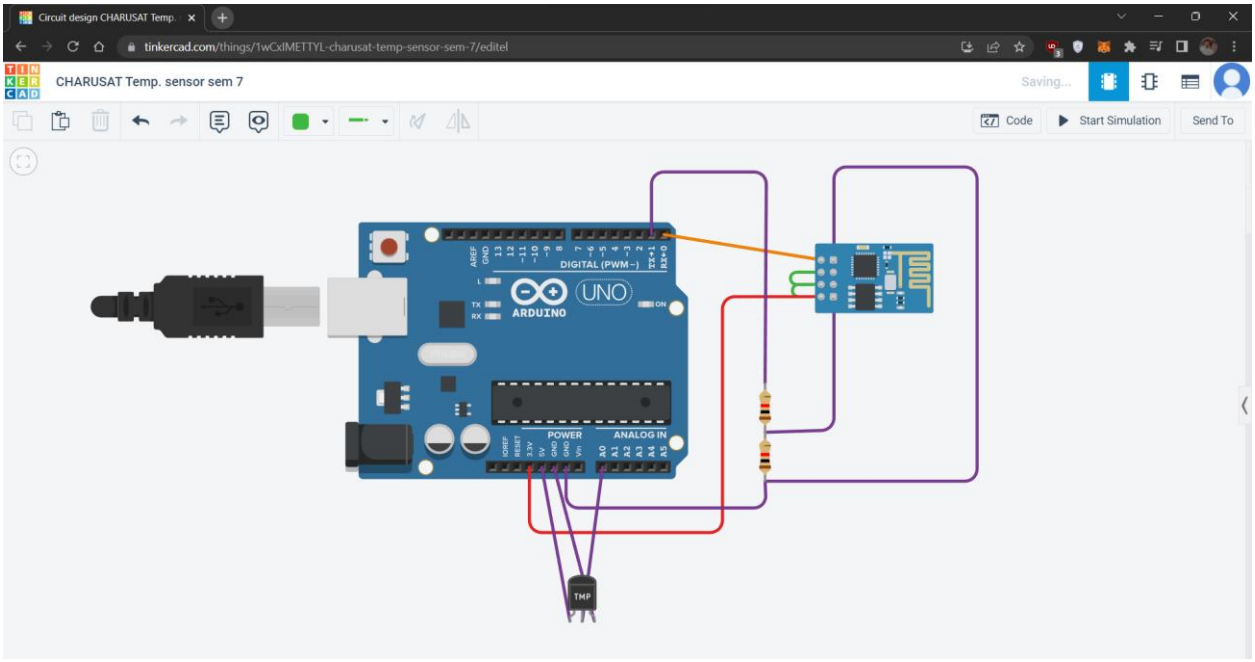
Step 1: Take Arduino Uno R3



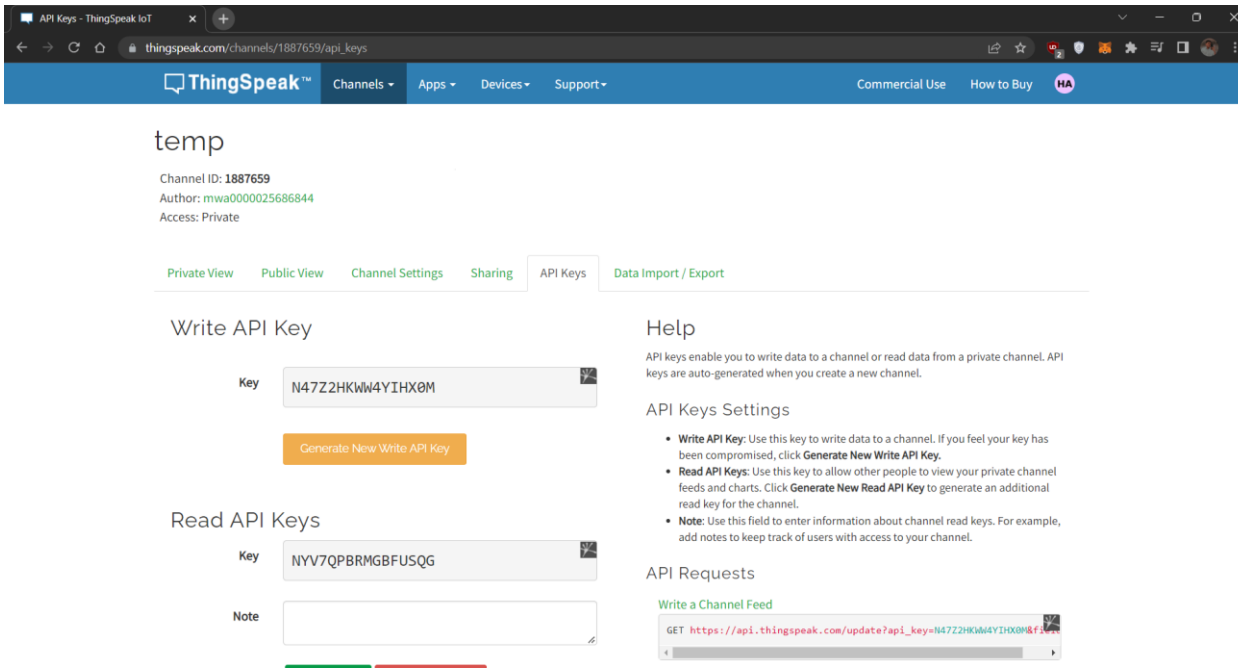
Step 2: Then take one Wifi module(ESP8266), 2 Resistor of 1kΩ and one temp sensor(TMP36).



Step 3: connect pins as shown in diagram.



Step 4: create channel in thingspeak.com and take write key in code.



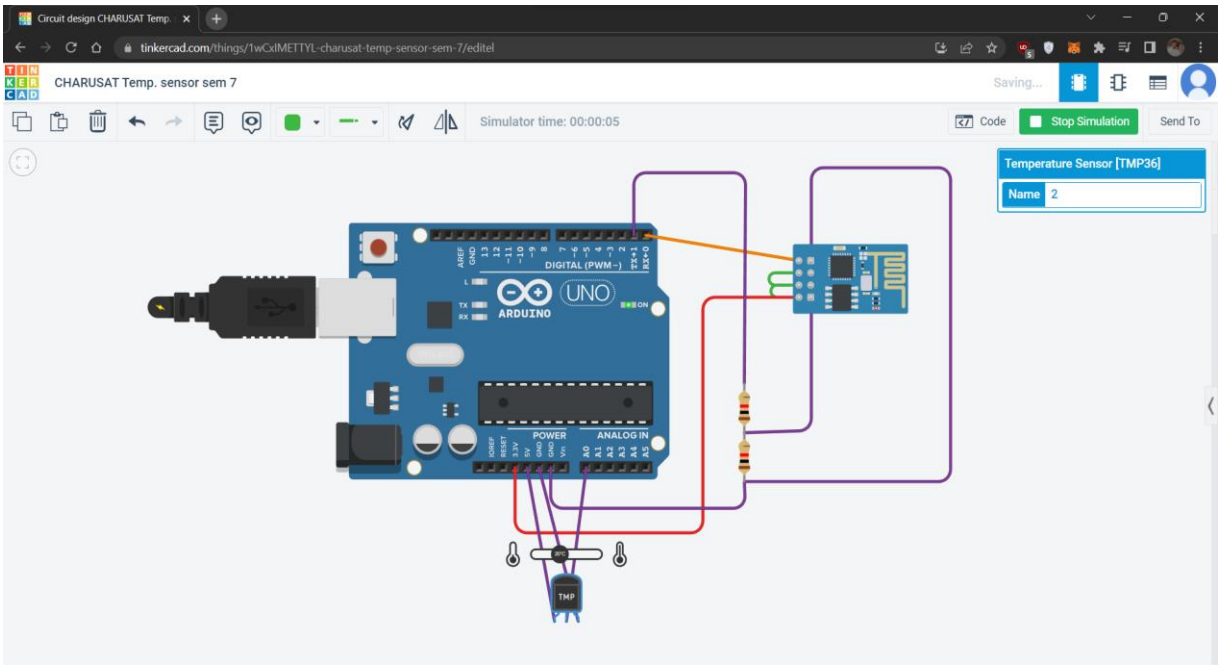
Step 5: Then write given code in thinkercad.

API KEY= N47Z2HKWW4YIHX0M

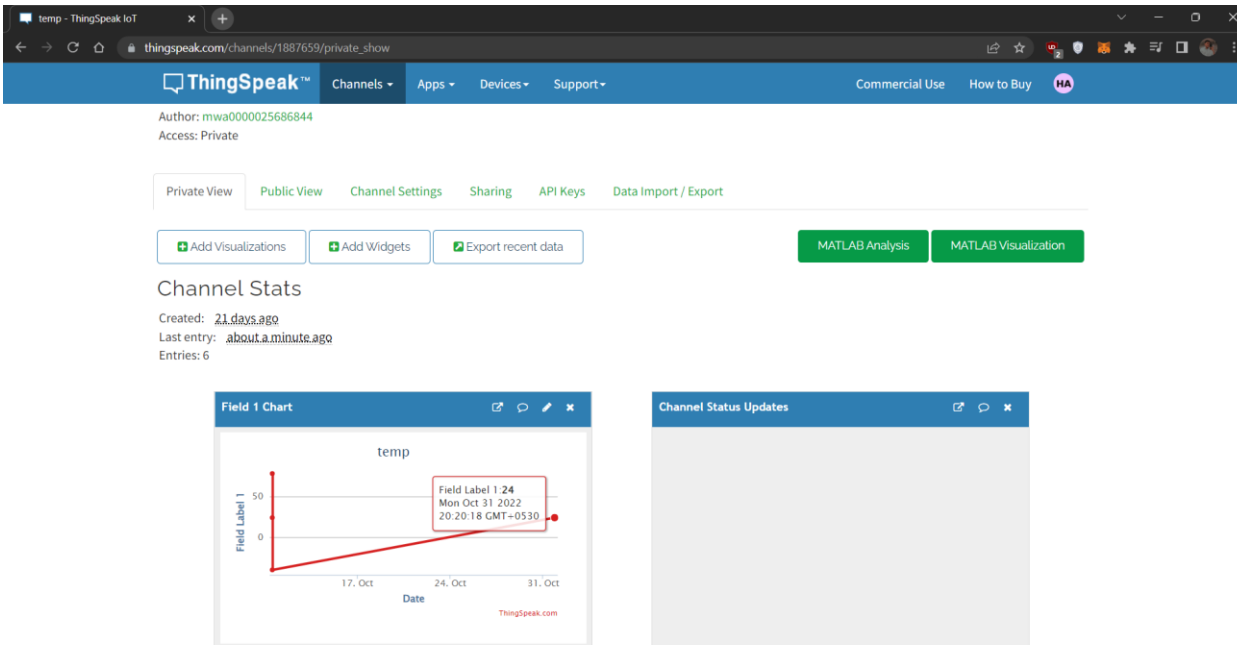
```
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = ""; // Our virtual wifi has no password
String host     = "api.thingspeak.com"; // Open Weather Map API
const int httpPort   = 80;
String uri     = "/update?api_key=N47Z2HKWW4YIHX0M&field1=";

int setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200);   // Serial connection over USB to computer
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);       // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 1;

  // Connect to 123D Circuits Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
  delay(10);       // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 2;

  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);       // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 3;

  return 0;
}

void anydata(void) {

  int temp = map(analogRead(A0),20,358,-40,125);

  // Construct our HTTP call
  String httpPacket = "GET " + uri + String(temp) + " HTTP/1.1\r\nHost: " +
host + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
```

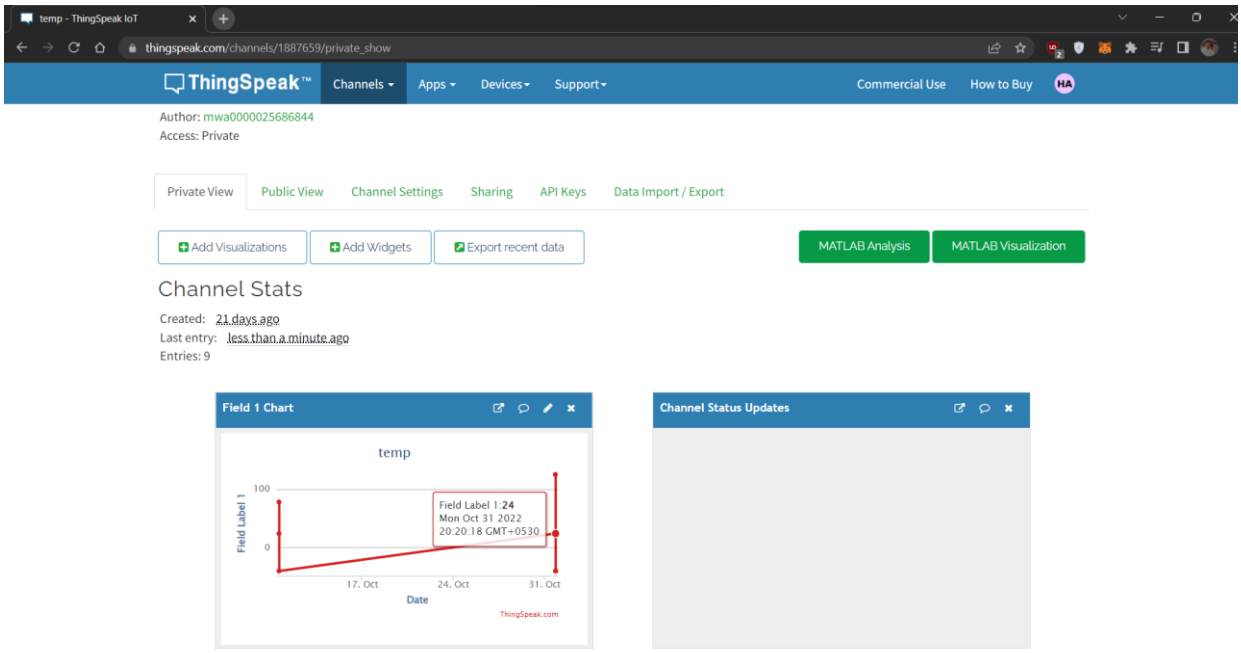
```
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return
-1;

  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("SEND OK\r\n")) return;
}
void setup() {
  setupESP8266();
}
void loop() {
 anydata();
  delay(10000);
}
```

Step 6: start simulate.



Step 7: move the temp sensor bar and see the output in thing speak.




temp - ThingSpeak IoT
thingspeak.com/channels/1887659/private_show
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
HA
Author: mwa0000025686844
Access: Private
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: 21 days ago
Last entry: less than a minute ago
Entries: 9
Field 1 Chart
temp
Field Label 1
100
0
Field Label 1: 24
Mon Oct 31 2022
20:20:18 GMT+0530
17. Oct
24. Oct
31. Oct
Date
ThingSpeak.com
Channel Status Updates