Department of Electronics & Communication Engineering

LAB MANUAL

SUBJECT: Internet of Things **[ECE-3062]**

B.Tech Year: 3rd Semester: VI
(Branch: CCE)

Version: 1(Jan, 2019)

LNMIIT
The LNM Institute of Information Technology

The LNM Institute of Information Technology
Jaipur, Rajasthan-302031

TABLE OF CONTENTS

Internet of Things Lab [ECE-3062]

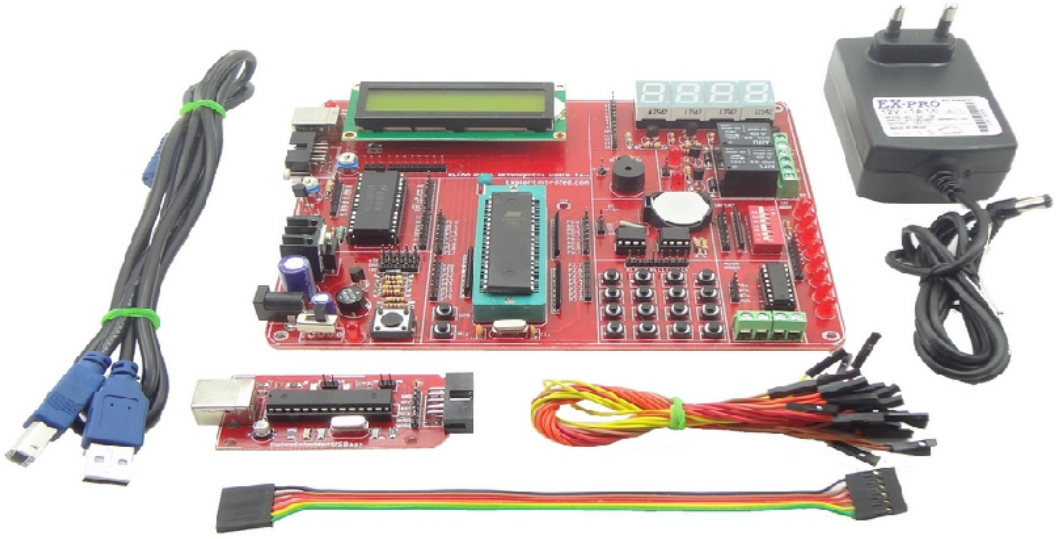
EXPERIMENT No. : 01

A. Aim: (a) To study of AT89S52 Ultra Development Kit with Development Tool /Environment of Kiel software for Microcontroller programming.

B. Apparatus Used: AT89S52 Ultra Development Kit, µ Vision Keil AT89S52 Microcontroller etc.

C. Theory:-

Our top seller is back with a revision, the 8051 Ultra Development board has got a revision with enhanced features. A onboard USB to serial converter to connect directly to your laptop/computer, we have done away with the RS232. Also the board is more spacious with clear division among various peripherals.We have designed this 8051 kit, so that a beginner can gradually build up from basics like LED blinking up to standard peripherals and communication protocols without having to change boards. We are shipping a USB AT89S series programmer with this board, for easy programming. The entire kit is very affordable and we have solid 22 video tutorials to help you get started from architecture to interfacing all the on-board peripherals



AT89S52 Ultra Development Kit

Kit Contens:

1. 8051 Development Board with following Modules, ICs and Interfaces

- USB Communication with CP2102
- Eight 5mm LEDs
- Eight Switches in DIP package
- LCD 16x2
- Four Seven Segment Displays
- ADC 0808 with 555 timer for sampling frequency

- Two Relays for Switching AC devices.
- 4x4 Keypad
- L293D Motor Driver for DC and Stepper Motors
- DS1307 based Real Time Clock
- AT2404 EEPROM
- Light Sensor(LDR)
- Temperature sensor(LM35)

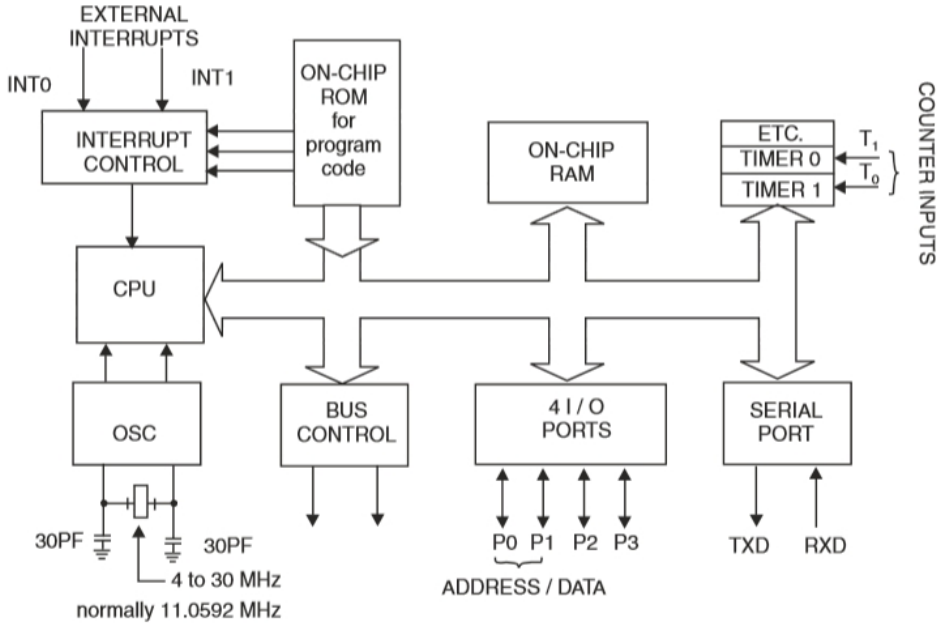
USB Programmer for AT89S series microcontrollers
3. USB cable.
4. Power Adapter
5. Single wire female connectors.

MICROCONTROLLER 8051 ARCHITECTURE

It is 8-bit microcontroller, means MC 8051 can Read, Write and Process 8 bit data. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries. Mostly used blocks in the architecture of 8051 are as follows:

- **4KB ROM**

In 8051, 4KB read only memory (ROM) is available for program storage.This is used for permanent data storage. Or the data which is not changed during the processing like the program or algorithm for specific applications.



8051 architecture

- **128 Byte RAM for Data Storage**

MC 8051 has 128 byte Random Access memory for data storage. Random access Memory is nonvolatile memory. During execution for storing the data the RAM is used. RAM consists of the register banks, stack for temporary data storage. Normally microcontroller has 256 byte RAM in which 128 byte is used for user space which is normally Register banks and stack.

We know that 128 byte = 2^7byte

- **Timers and Counters**

  In MC8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode.We can count external pulses at these pins if we use these pins in counter mode.16 bits timers are available. Means we can generate delay between 0000H to FFFFH.

If we want to load T0 with 16 bit data then we can load separate lower 8 bit in TL0 and higher 8 bit in TH0. In the same way for T1. TMOD, TCON registers are used for controlling timer operation.

- **Serial Port**

There are two pins available for serial communication TXD and RXD. Normally TXD is used for transmitting serial data which is in SBUF register,RXD is used for receiving the serial data.

- **Input Output Ports**

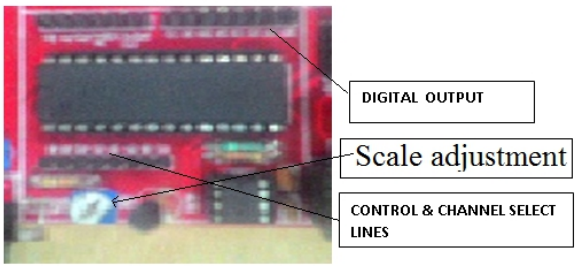
There are four input output ports available P0, P1, P2, P3. Each port is 8 bit wide and has special function register P0, P1, P2, P3 which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.Port 2 can be used as I/O port as well as higher order address bus A8 to A15. Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function

P3.0 – RXD – Serial I / P for Asynchronous communication
Serial O / P for synchronous communication.
P3.1 – TXD – Serial data transmit.
P3.2 – INT0– External Interrupt 0.
P3.3 – INT1 – External Interrupt 1.
P3.4 – T0 – Clock input for counter 0.
P3.5 – T1 – Clock input for counter 1.
P3.6 – WR – Signal for writing to external memory.
P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

- **Oscillator**
  We use crystal which frequency vary from 4MHz to 30 MHz, normally we use 11.0592 MHz frequency.
- **Interrupts**
  Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged.

ANALOG TO DIGITAL CONVERTER (ADC0809)



Features:

- Easy interface to all microprocessors
- Operates ratio metrically or with 5 VDC or analog span
- adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

- Resolution 8 Bits
- Total Unadjusted Error ±1⁄2 LSB and ±1 LSB
- Single Supply 5 VDC

- Low Power 15 mW
- Conversion Time 100

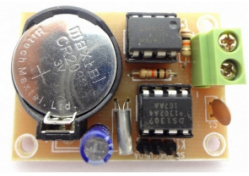
LCD 16x2 Disply

Eight Switches in DIP Packag

DC MOTOR SECTION

RTC & EEPROM SECTION

The RTC EEPROM uses DS1307 and AT24C16 ICs respectively. The EEPROM is used to store data permanently. It can be quite helpful if you want to store data from sensors or other sources in real-time. The Real time clock keeps ticking with the help of CR2032 battery.



A real time clock is basically just like a watch - it runs on a battery and keeps time for you even when there is a power outage! Using an RTC, you can keep track of long timelines, even if you reprogram your microcontroller or disconnect it from USB or a power plug.

DS1307 (64 x 8 Serial Real-Time Clock)
DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus.The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

AT24C16

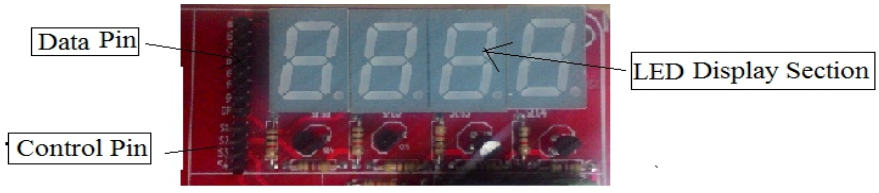
Features

• Low-voltage and Standard-voltage Operation– 2.7 (V_{CC}= 2.7V to 5.5V)– 1.8 (V_{CC}= 1.8V to 5.5V)
•Internally Organized 128 x 8 (1K), 256 x 8 (2K), 512 x 8 (4K),1024 x 8 (8K) or 2048 x 8 (16K)
•Two-wire Serial Interface
•Schmitt Trigger, Filtered Inputs for Noise Suppression
•Bidirectional Data Transfer Protocol
•100 kHz (1.8V) and 400 kHz (2.7V, 5V) Compatibility
•Write Protect Pin for Hardware Data Protection
•8-byte Page (1K, 2K), 16-byte Page (4K, 8K, 16K) Write Modes
•Partial Page Writes Allowed
•Self-timed Write Cycle (5 ms max)
•High-reliability
– Endurance: 1 Million Write Cycles
– Data Retention: 100 Year.

CR2032 battery

A **CR2032 battery** is a button cell lithium battery rated at 3.0 volts. It is commonly used in computers as a CMOS battery, calculators, remote controls, scientific instruments, wireless doorbells, watches, and other small devices.

CIRCUIT DIAGRAM FOR DISPLAY SECTION



CIRCUIT DIAGRAM FOR RELAY & BUZZER SECTION

RELAY

A **relay** is an electromagnetic switch operated by a relatively small electric current that can turn on or off a much larger electric current. The heart of a **relay** is an electromagnet (a coil of wire that becomes a temporary magnet when electricity flows through it).

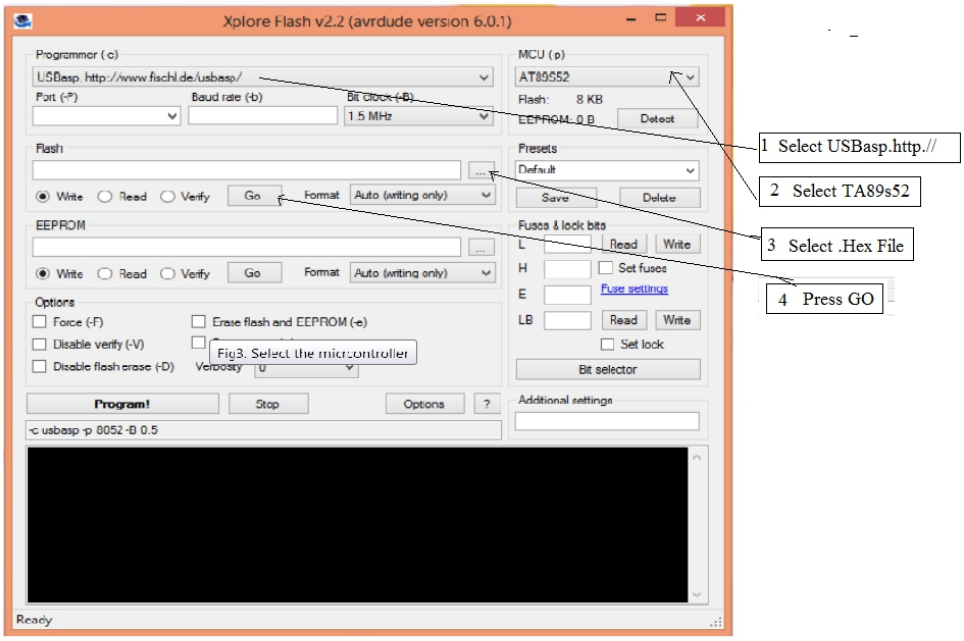
D. Procedure:-

General Procedures for Keil µ Vision 5:-

1. Create a **New folder** on the **desktop** for saving the Project.
2. Double Click on the icon **Keil µ Vision 5** present on the desktop.
3. **Keil µ Vision 5** window will appear on desktop.
4. Close the **Existing Project** if exists.
5. Go to the project menu & click on **New µ Vision project** after this **Create New Project** window will appear on desktop write the **New Project Name⟶ Select IC AT89S52⟶ OK⟶ YES.**
6. Go to **target 1** & then **source group**, Right click on there & click on the option "**Add Files to the Group 'Source Group 1'** ".
7. Select your .asm or .c file which you want to add. depending on your coding and then **Add**.
8. **Program editing window** will appear on desktop with **New File Name**.
9. Write or copy you **code** there & **save (Ctrl+S).**
10. Right click **Target 1**, go to the **Option for Target 'Target 1'** or (**Alt+F7) Option for Target 'Target 1'** window will appear on desktop **click on output** & tick on **Create HEX File** option and then **OK**.
11. Now click on "**Translate current file**", "**Build Target**" and "**Rebuilt all target files**".
12. After this **Hex file** will be created in the **New Folder**.
13. It will show you 0 errors & 0 warning on Output Window.

After performing all these steps the chip will be configured through Xplore Flash.

How to use Xplore Flash



Xplore Flash v2.2 (avrdude Version 6.0.1)

1. A window will be popped-up
2. Select **USBasp.http://www.fischl.de/usbasp/** in appear window on desktop.
3. Now select **AT89S52/AT89C51** microcontroller in appear window on desktop.
4. Now select **Hex File** to burn in chip through browse option Flash and open.
5. Now press **Go** Xplore Flash program will burn with 5-6 Seconds.

E. **Observation:** Write/ Plot Your Own With Observation Table (If Required).

F. **Analysis of Results:** (Write your own)

G. Conclusions:

The addition of two 8-bit numbers is performed using **AT8051 Ultra Development Kit** where sum is 8 – bit result display on LEDS in Binary form.

H. Precautions:

1. Properly connect the AT89S52 Ultra Development Kit with USB Cable.
2. Handle the Trainer kit carefully.

Internet of Things Lab [ECE-3062]

EXPERIMENT No. : 02

Aim: - To familiarize with Intel Galileo Gen2 board and understand the procedure of creation and compilation of C source code.

Objectives: Student should get the knowledge of Intel Galileo Board and different types of LED

Outcomes: Student will be Write program using Arduino IDE for Blink LED

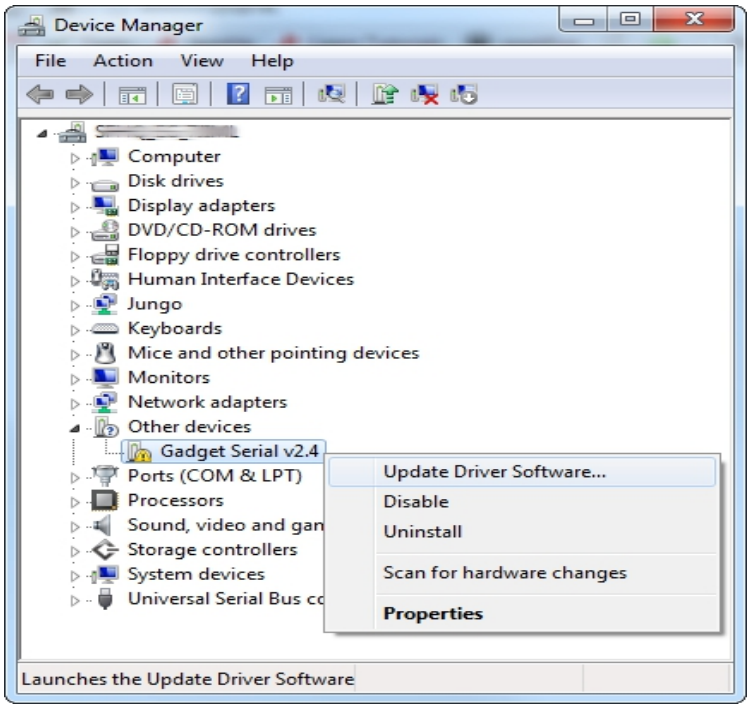
Apparatus Used -

1. Intel® Galileo Board
2. Micro SD Card
3. SD Card Reader
4. +5V Power adapter
5. USB to Serial Cable
6. Micro USB Cable

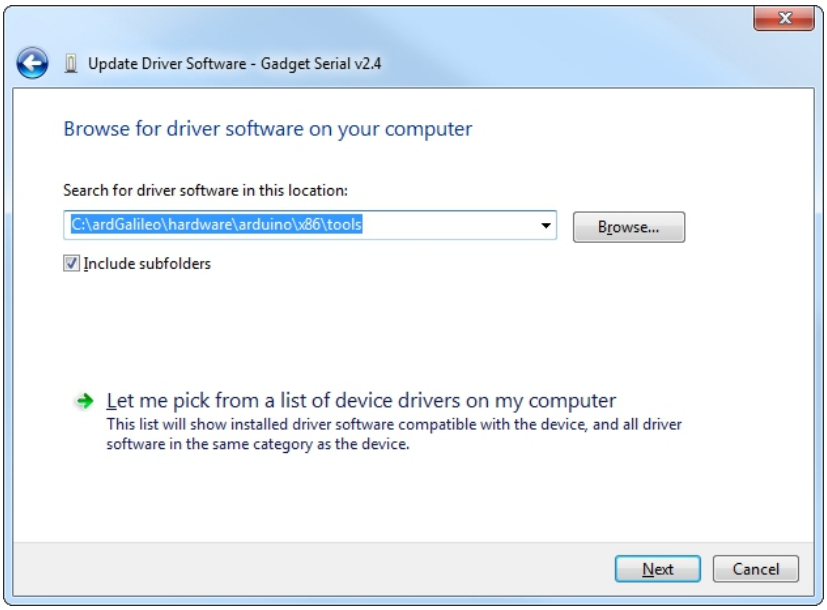
Theory:- Galileo is a microcontroller board based on the Intel® Quark SoC X1000 Application Processor, a 32-bit Intel Pentium-class system on a chip (datasheet). It's the first board based on Intel® architecture designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. Digital pins 0 to 13 (and the adjacent AREF and GND pins), Analog inputs 0 to 5, the power header, ICSP header, and the UART port pins (0 and 1), are all in the same locations as on the Arduino Uno R3. This is also known as the Arduino 1.0 pinout.
Galileo is designed to support shields that operate at either 3.3V or 5V. The core operating voltage of Galileo is 3.3V. However, a jumper on the board enables voltage translation to 5V at the I/O pins. This provides support for 5V Uno shields and is the default behavior. By switching the jumper position, the voltage translation can be disabled to provide 3.3V operation at the I/O pins.

Procedure:

1. Connect a **5V power supply** to the Galileo. (The USB port alone cannot supply enough power to run the Galileo.)
2. Connect a micro-B USB cable from the Galileo's **USB Client** port to an available USB socket on your computer.
3. Upon connecting the board, Windows will automatically attempt to install the driver and, unsurprisingly, it will fail. We'll have to manually install the driver.
4. Open up the **Device Manager**. (Either *Start* > *Run* > *devmgmt.msc*, or go to the *Control Panel*, select *System* and click *Device Manager*.)
5. Locate the *Gadget Serial v2.4* device, under the *Other devices* tree. **Right-click** that and select **Update Driver Software...**



6. On the first window that pops up, click **Browse my computer for driver software**. And on the next page select***Browse...*** and navigate to the hardware\arduino\x86\tools folder within your Arduino Galileo software installation. Then click **Next.**



7. **Click Install** on the next *Windows Security* window that pops up. And, after a number of Loading-bar-scrolls, the installation should complete and you should be greeted with a *Windows has successfully updated your driver software* window.

8. Look back at the **Device Manager**, under the *Ports* tree now. There should be an entry for **Galileo (COM #)**. Remember which COM # your Galileo is assigned, it'll be important for Arduino sketch uploading.

Uploading Blink

As always, the first program to be uploaded to a board is the "Hello, world" of microcontrollers - Blink.

To open the Blink example, go to the **File** > **Examples** > **01.Basics** > **Blink**.

```
/* Blink
Turns on an LED on for one second, then off
for one second, repeatedly. */
// Pin 13 has an LED connected on most Arduino
boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press
reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again
forever:
void loop() {
digitalWrite(led, HIGH); // turn the LED on
(HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off
by making the voltage LOW
delay(1000);
// wait for a second
}
```

Make sure the **Serial Port** and **Board** selections are still correct. Compilers supported are GCC and ICC. Then click the **Upload** button. Upload Button will Compile and run the code.

After the upload completes, you should see a tiny, green LED blinking on and off every second. This LED is connected to pin 13 of the Galileo.

Result :- Compilation and Installation of Arduino IDE & Drivers is done and process understood.

Conclusion :- Target board of such types can be used in low cost system designs using very less amount of components and can be used for many user defined applications or customizations.

Internet of Things Lab [ECE-3062]

EXPERIMENT No. : 03

Aim: - Wifi module interfacing with Intel Galileo Gen2 Board.

Apparatus Used -

9. Intel® Galileo Board
10. Micro SD Card
11. SD Card Reader
12. +5V Power adapter
13. USB to Serial Cable
14. Micro USB Cable

Theory:-

Make a bootable micro SD card

You must boot your Intel® Galileo board using a micro SD card that contains the latest Intel® IoT Developer Kit version of the Yocto*-built Linux image.

Linux is the operating system that powers the Intel® Galileo board. While there is already a version of Linux built into your board, the developer-kit version of Yocto-built Linux includes even more libraries and resources to help developers create applications in their favorite programming language. This version includes GCC, Python, Node.js, OpenCV, to name a few.

This section contains steps to download and extract the latest developer kit version of the Linux image.

1. Download and install the 7-Zip file archiver from 7-Zip.org.
2. Download the latest **Intel® Galileo Board micro SD Card Linux* Operating System Image** from the Intel® Galileo Board Downloads page.
3. Use 7-Zip to extract the .bz2 file to your system. You must use 7-Zip to extract the image, as it supports the extended file paths found in the compressed image file.

   You should now have a file called `iot-devkit-version-mmcblkp0.direct` alongside the original .bz2 file, where *`version`* is either `latest` or a date in the *`YYYYMMDDHHMM`* format.

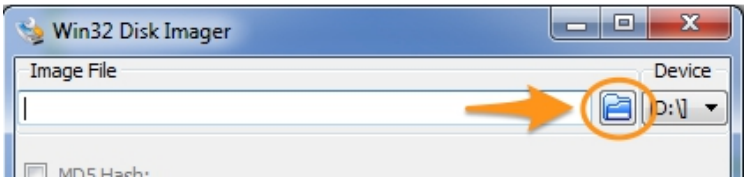
Write the image to your micro SD card

This section contains steps to write the image to your micro SD card.

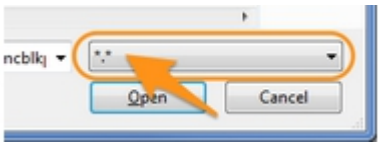
1. Visit http://sourceforge.net/projects/win32diskimager and click the **Download** button to save the latest `Win32DiskImager-version-binary.zip` file to your system.
2. Insert the micro SD card into your card reader.
3. Run Win32 Disk Imager as an Administrator, as follows:

a. Double-click **Win32DiskImager-binary.zip** to expand its contents.
b. Open the **Win32DiskImager-binary** folder.
c. Right-click **Win32DiskImager.exe**, then select **Run as administrator**.

4. In Win32 Disk Imager, click the folder icon on the top right.

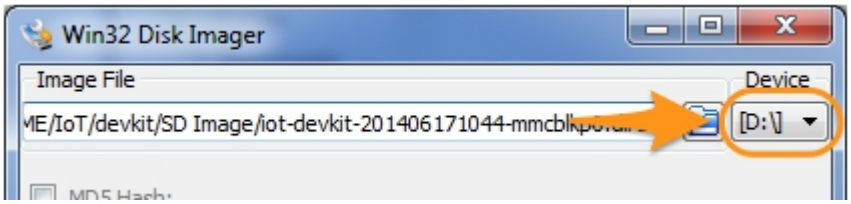


5. In the file type drop-down list, select ***.*** so you can see all files, regardless of file extension.



6. Navigate to and select the .direct file that you extracted earlier, then click **Open**.
7. From the **Device** drop-down list, select the device drive of your micro SD card.

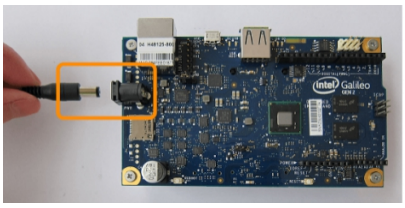
   Caution: Be sure to select the correct device drive, as the drive letter for your card may be different on your system. Selecting the wrong drive letter could result in erasing your data on the wrong drive.



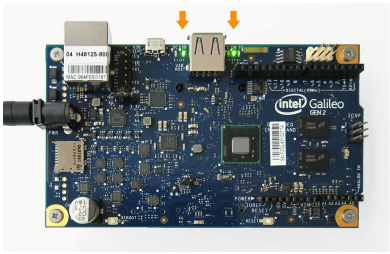
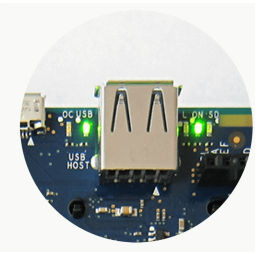
8. Click **Write**, then wait for the write process to finish. Please be patient, as this may take up to 5 minutes.
9. After completing the write process, click **Exit** to close Win32 Disk Imager. Eject and remove the micro SD card from your card reader.
10. You can now insert the card into the micro SD card slot on your Intel Galileo board. When you power up your board, the board automatically boots using the image on the card.

Set up Galileo Gen 2 board

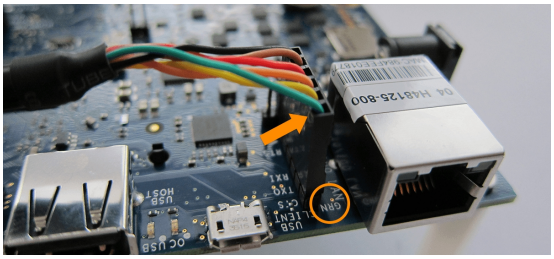
1. Confirm flashed SD card is inserted in your board properly.
2. Plug the DC power supply in to your board. Plug the other end into your wall socket.




3. Wait for two green LEDs to light up.

4. Plug in the 6 pin Serial (FTDI) to USB cable.



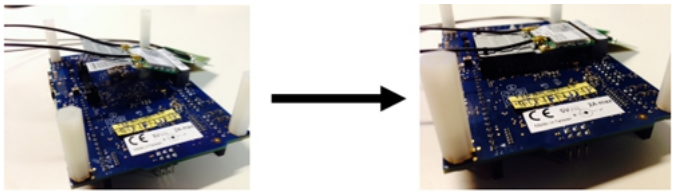
5. Plug the other end of the FTDI cable in to a USB port on your computer.

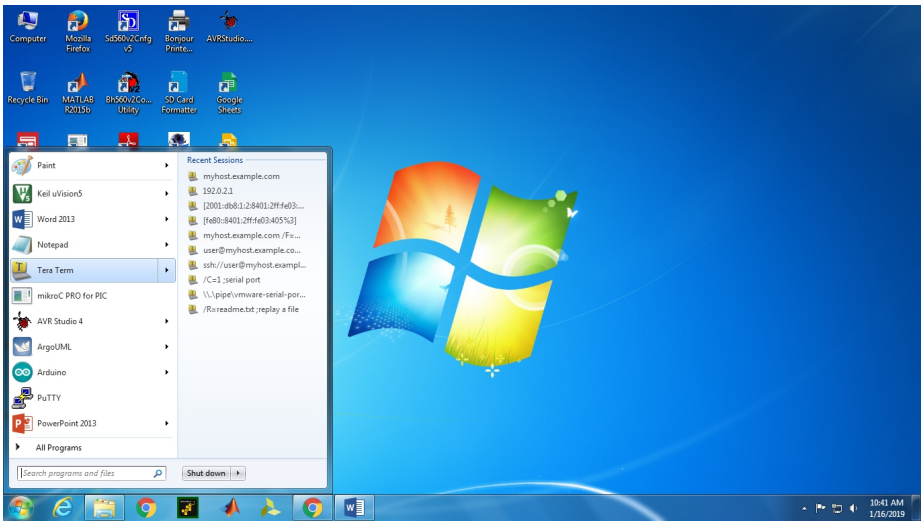

Install your Wi-Fi adaptor

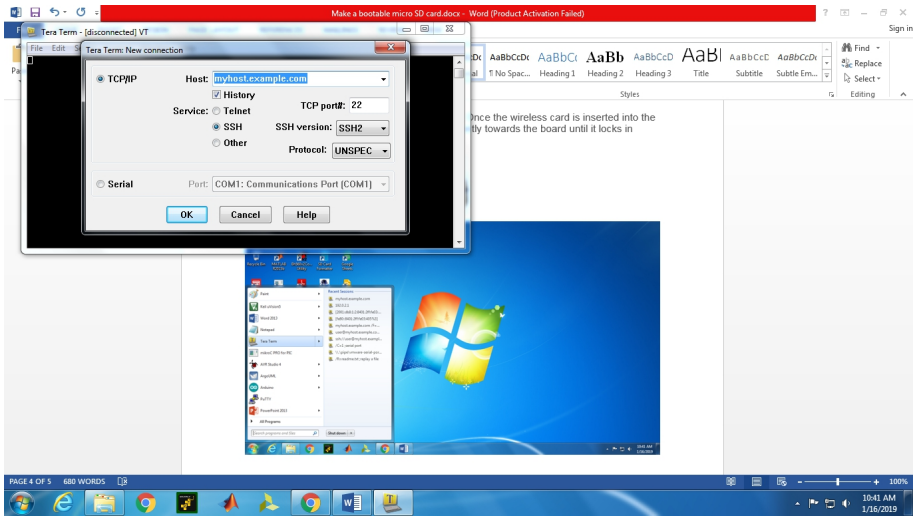
For the purpose of this tutorial we are using the Intel® Centrino® Wireless-N 2230.

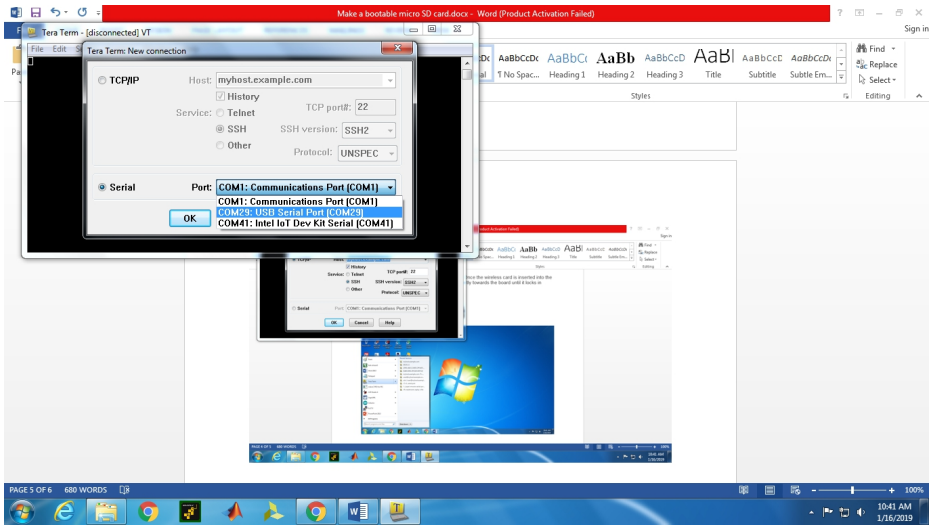
1. Assemble the wireless card with the aluminum plate, to help keep the wireless card in place on the board.
2. Connect the antennas to the wireless card.

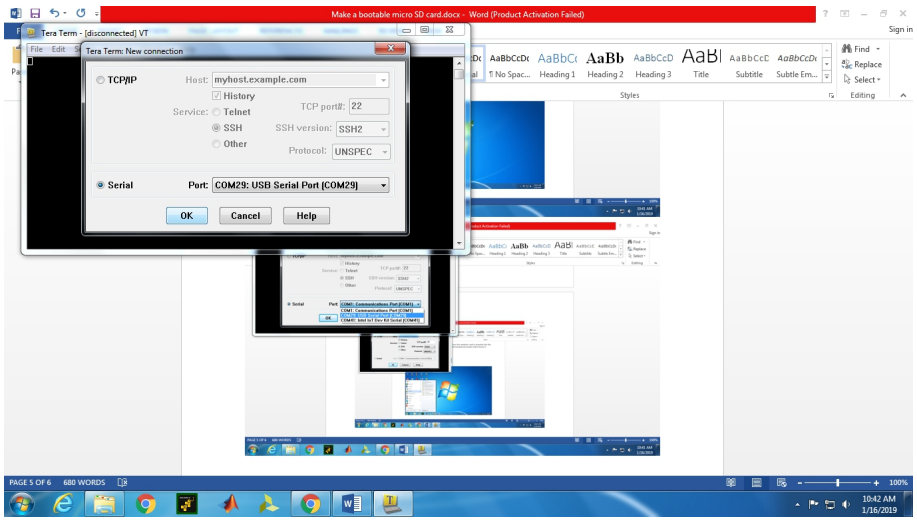
3. Insert wireless card in the mini PCI Express slot. Once the wireless card is inserted into the Mini PCI Express slot, press the wireless card gently towards the board until it locks in place.

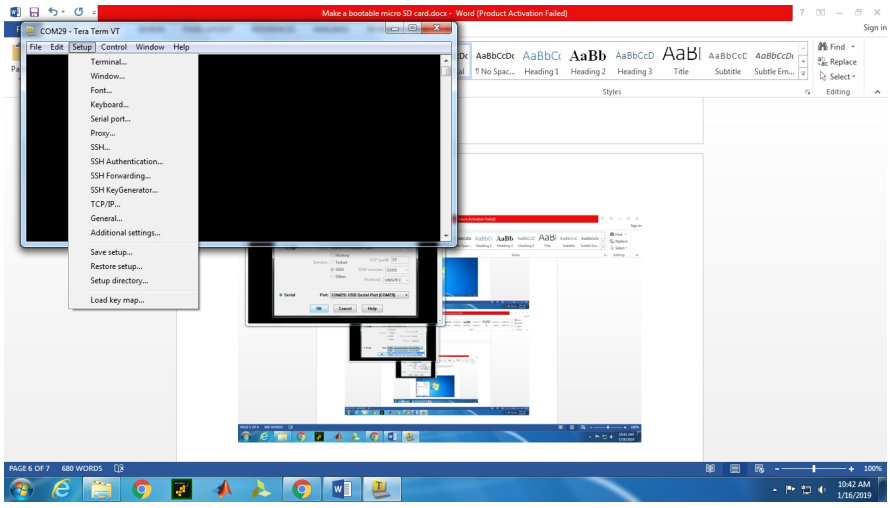


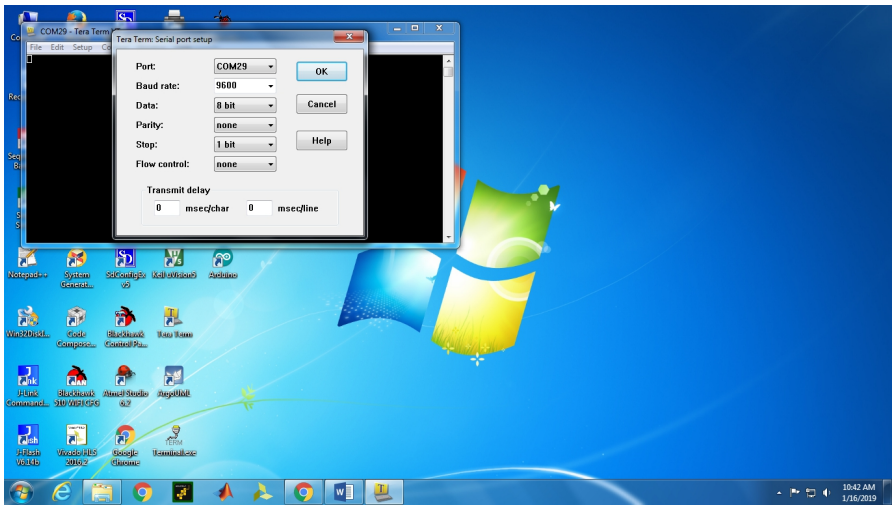


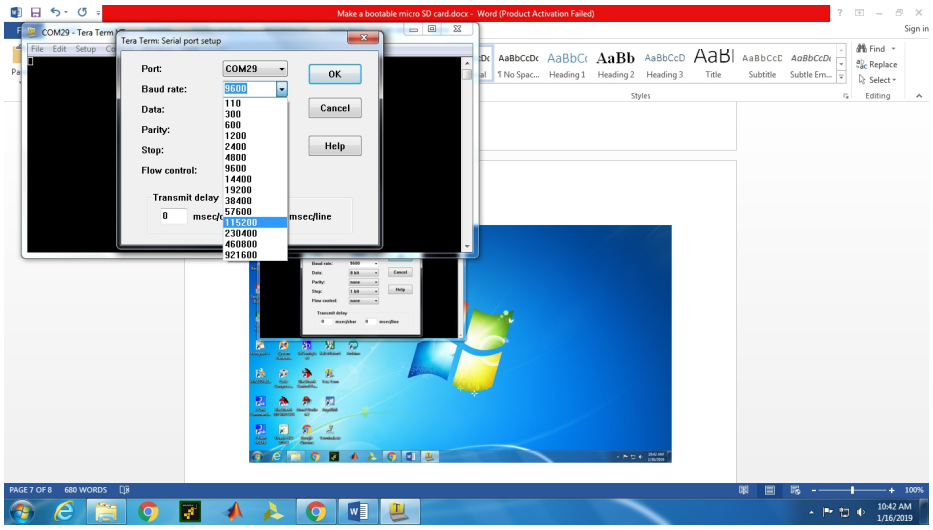


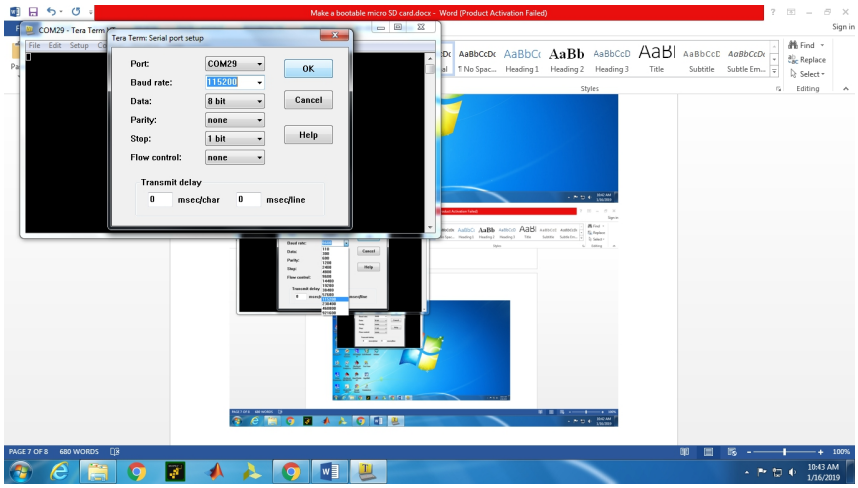

Make a bootable micro SD card.docx - Word (Product Activation Failed)
Tera Term - [disconnected] VT
Tera Term: New connection
TCP/IP
Host: myhost.example.com
History
Service: Telnet
SSH
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC
Serial
Port: COM1: Communications Port (COM1)
COM1: Communications Port (COM1)
COM29: USB Serial Port (COM29)
COM41: Intel IoT Dev Kit Serial (COM41)
OK
PAGE 5 OF 6 680 WORDS
10:41 AM
1/16/2019

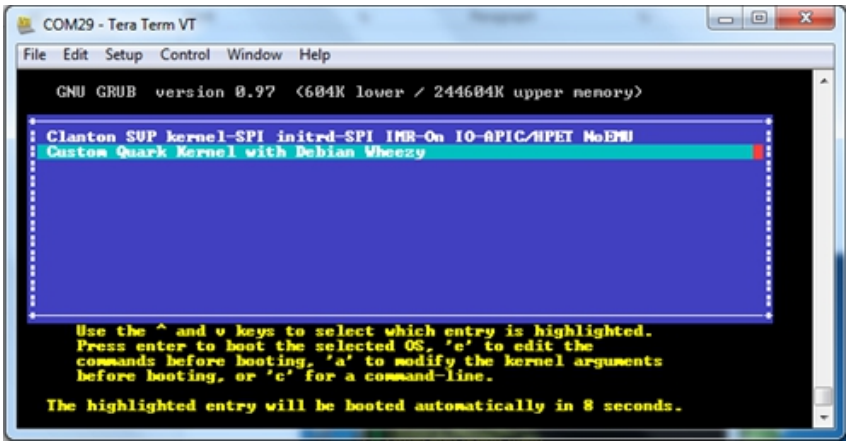

Make a bootable micro SD card.docx - Word (Product Activation Failed)
Tera Term - [disconnected] VT
Tera Term: New connection
TCP/IP
Host: myhost.example.com
History
Service: Telnet
SSH
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC
Serial
Port: COM29: USB Serial Port (COM29)
OK
Cancel
Help
PAGE 5 OF 6 680 WORDS
10:42 AM
1/16/2019

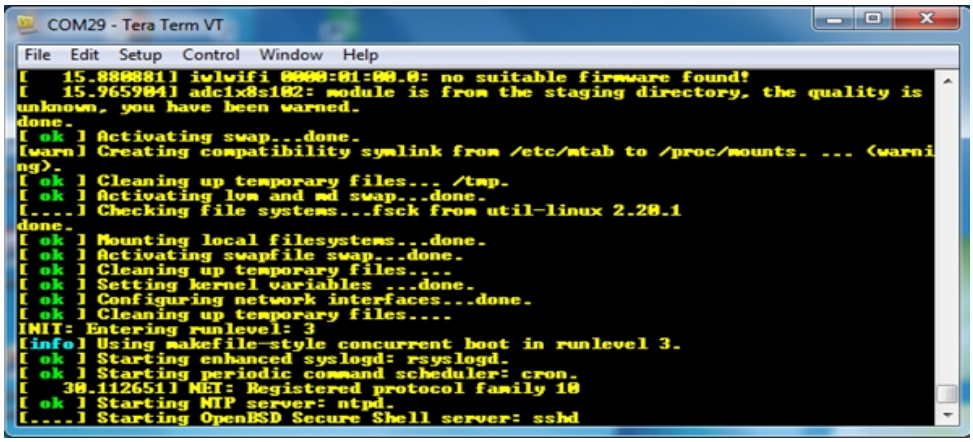

Make a bootable micro SD card.docx - Word (Product Activation Failed)
COM29 - Tera Term VT
File Edit Setup Control Window Help
Terminal...
Window...
Font...
Keyboard...
Serial port...
Proxy...
SSH...
SSH Authentication...
SSH Forwarding...
SSH KeyGenerator...
TCP/IP...
General...
Additional settings...
Save setup...
Restore setup...
Setup directory...
Load key map...
PAGE 6 OF 7 680 WORDS
10:42 AM
1/16/2019


Tera Term: Serial port setup
Port: COM29
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line
OK
Cancel
Help

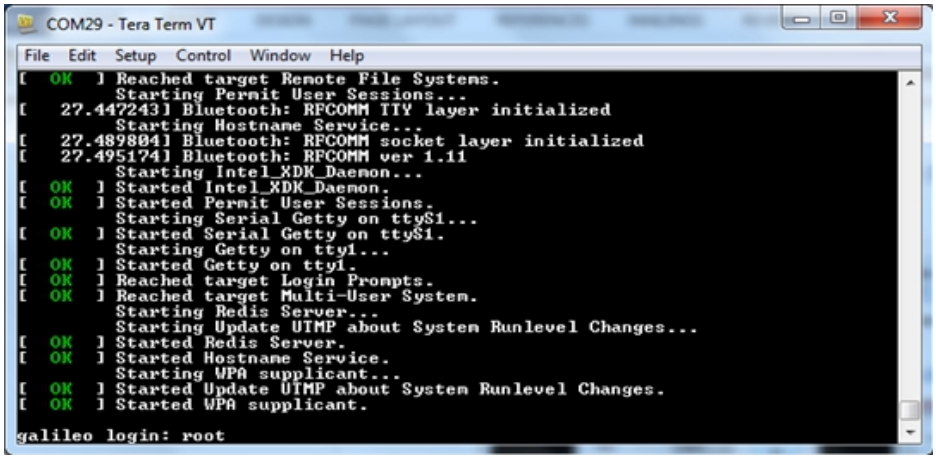

Make a bootable micro SD card.docx - Word (Product Activation Failed)
Tera Term: Serial port setup
Port: COM29
Baud rate: 9600
110
300
600
1200
2400
4800
9600
14400
19200
38400
57600
115200
230400
460800
921600
Data:
Parity:
Stop:
Flow control:
Transmit delay
msec/line
OK
Cancel
Help


Make a bootable micro SD card.docx - Word (Product Activation Failed)
Tera Term: Serial port setup
Port: COM29
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line
OK
Cancel
Help





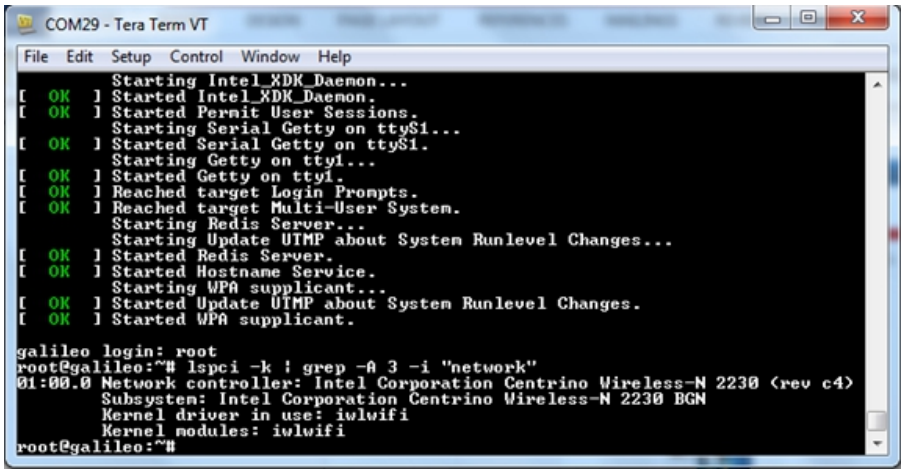
Login name root



Connecting to a Wi-Fi Network

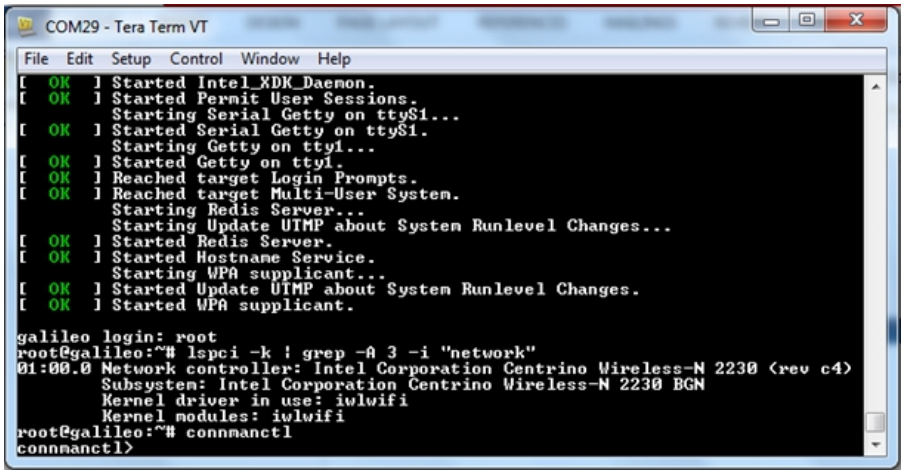
Verify the Wi-Fi adapter is connected and working

root@galileo:# **lspci -k | grep -A 3 -i "network"**



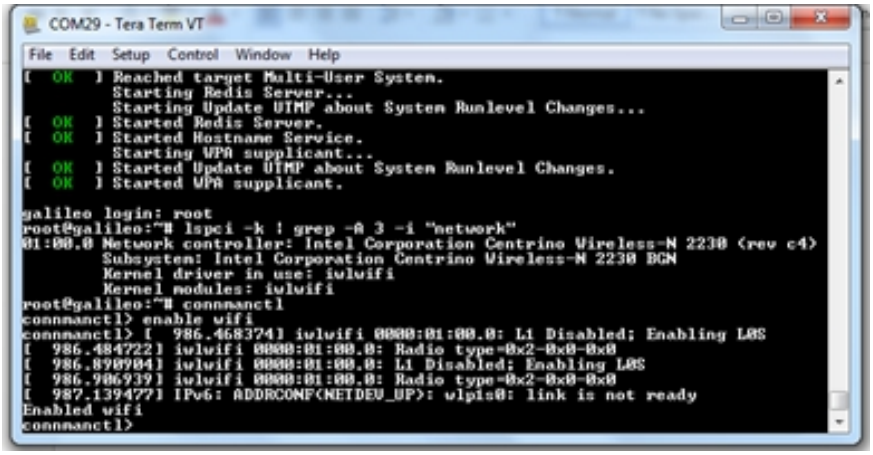
To establish the Wi-Fi connection

root@galileo:# **connmanctl**



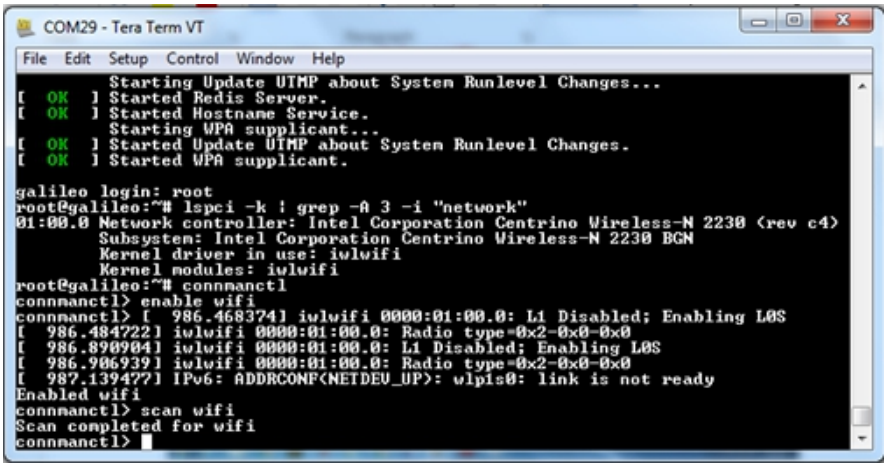
To enable Wi-Fi

connmanctl> **enable wifi**



To scan Wi-Fi

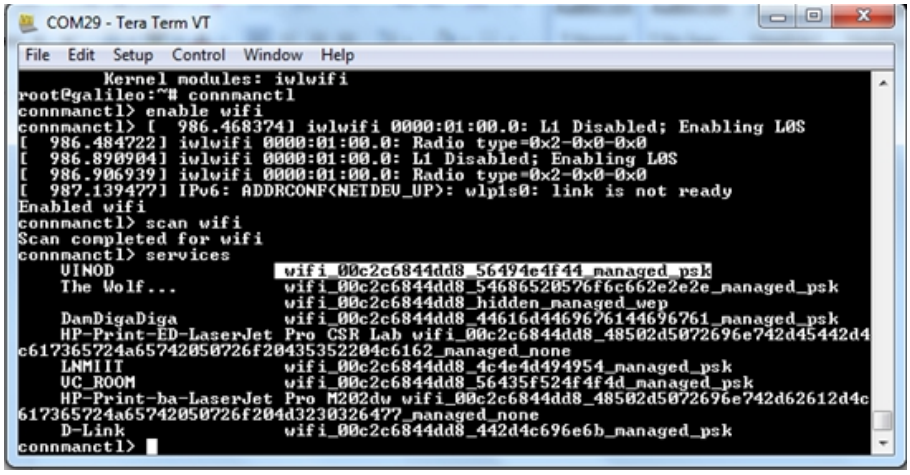
connmanctl> **scan wifi**



Connecting to a Wi-Fi Network

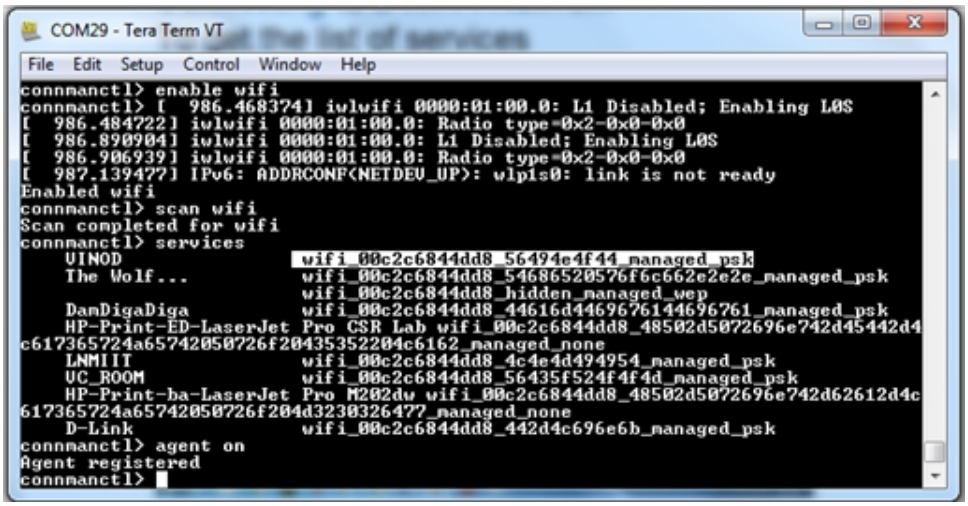
To get the list of services

connmanctl> **services**

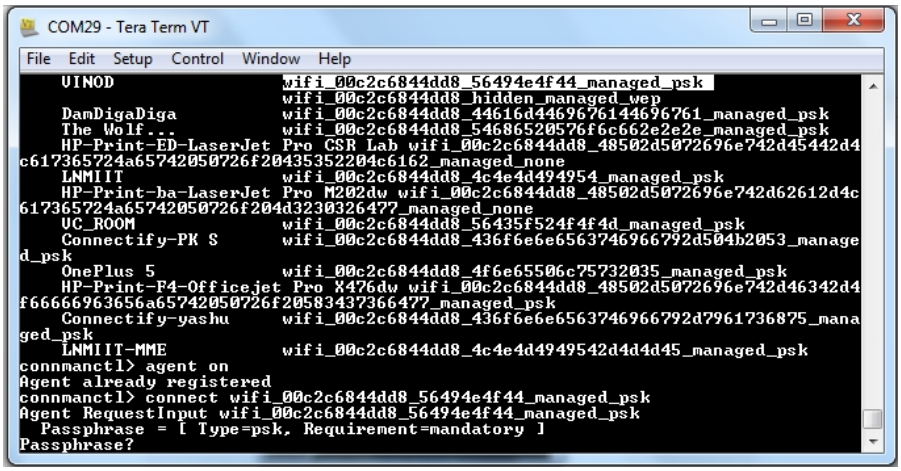


To connect to a secure Wi-Fi network

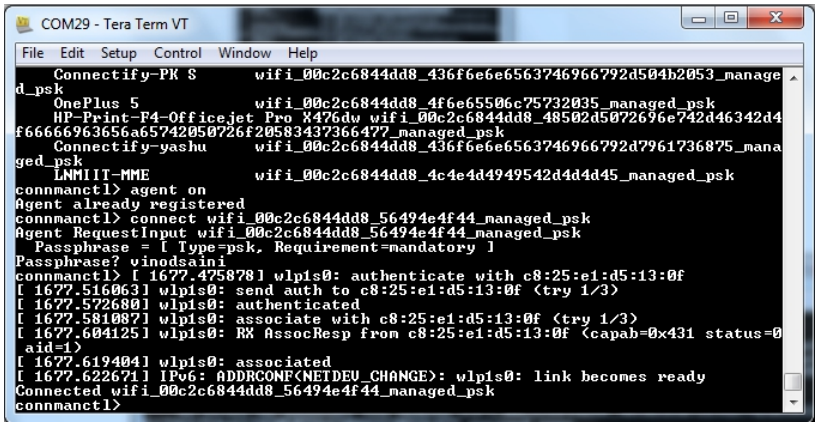
connmanctl> **agent on**



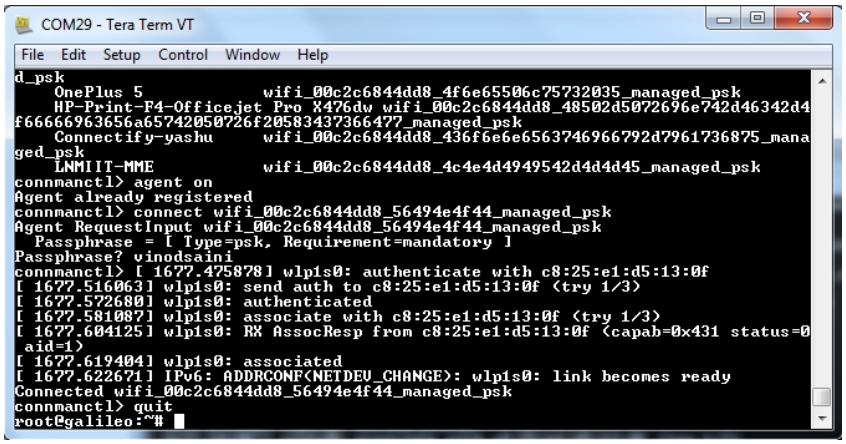
Connecting to a Wi-Fi Network



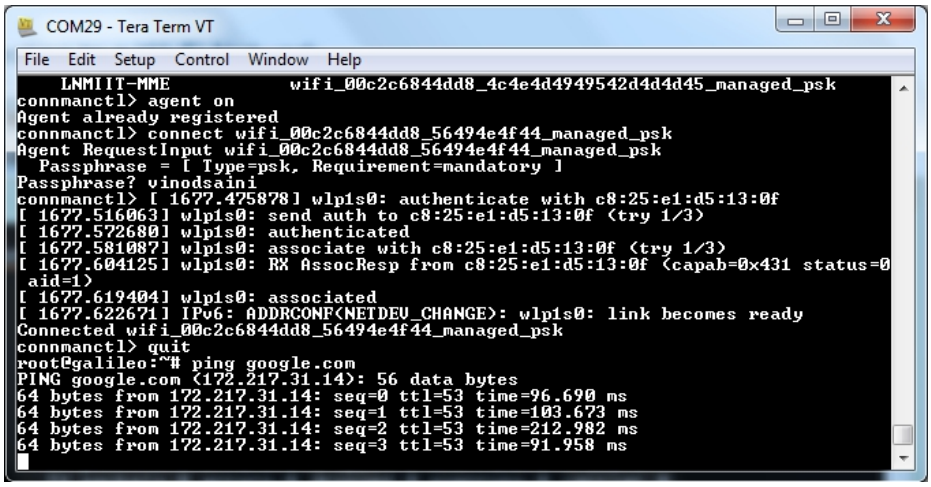
Type your password



Type “exit” to leave connmanctl, and then “ifconfig” to see the IP address associated with wifi.



Type “ping google.com” in the shell to see if Intel® Galileo Gen 2 is already connected to internet.



Analysis of Results: (Write your own)

Conclusions:

Precautions:

Internet of Things Lab [ECE-3062]

EXPERIMENT No. : 04

Aim: - To study of IoT Data Logging using Beaglebone Black and Thingspeak.

Apparatus Used -

1. Beaglebone Black
2. USB cable
3. Bread board
4. Light Dependent Resistor(LDR)
5. 10K Resistor
6. PC with internet connection and Putty installed.
7. Latest Debian distribution of Linux installed.

Theory:- The BeagleBone Black is unique in that it has quite a few pins that are available on easy to use pin headers, as well as being a fairly powerful little system. There are 2 x 46 pins available (well, not all of them are, but we'll get to that later) to use.
Some of the functionality that is available:

1. 7 Analog Pins
2. 65 Digital Pins at 3.3V
3. 2x I2C
4. 2x SPI
5. 2x CAN Bus
6. 4 Timers
7. 4x UART
8. 8x PWM
9. A/D Converter

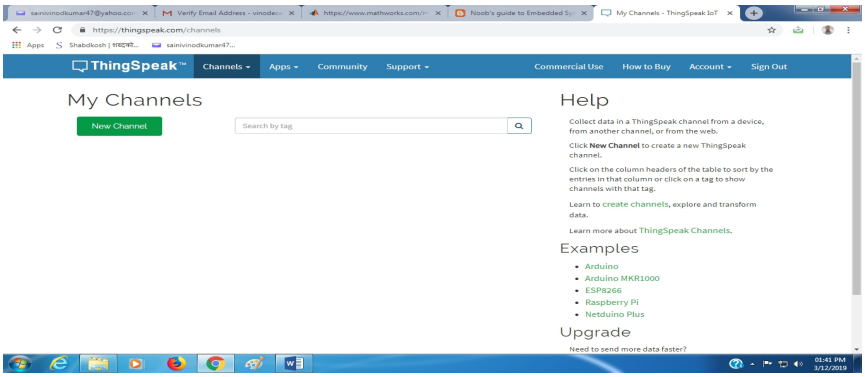
Setting up Thingspeak

Thing speak is a free online cloud service that lets it's users to create data channels which they can connect with their devices with the unique API Key generated with each channel. The connected device just need to

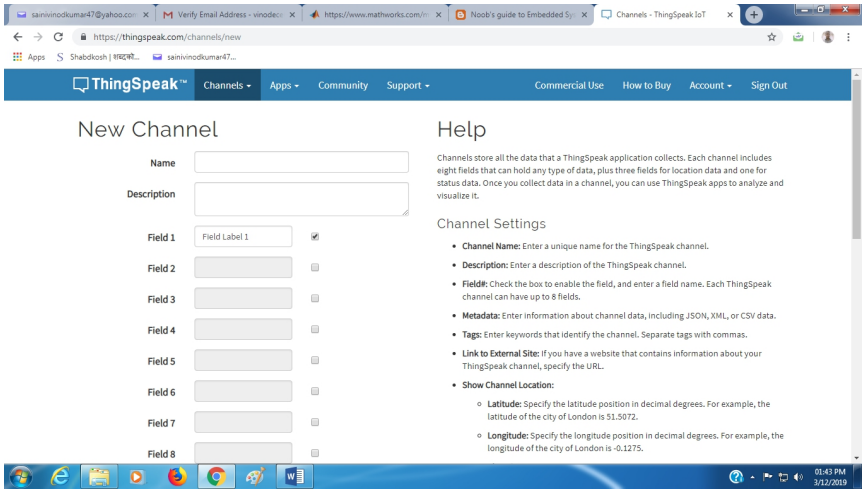
send an HTTP POST request along with the API key and the data to update the data channels. Don't worry if it looks too complicated. Let's make it simple.

Create a Thingspeak account by clicking on getting started. Now you will be asked to register yourself with thingspeak.

After you are registered you will be with a screen like this.

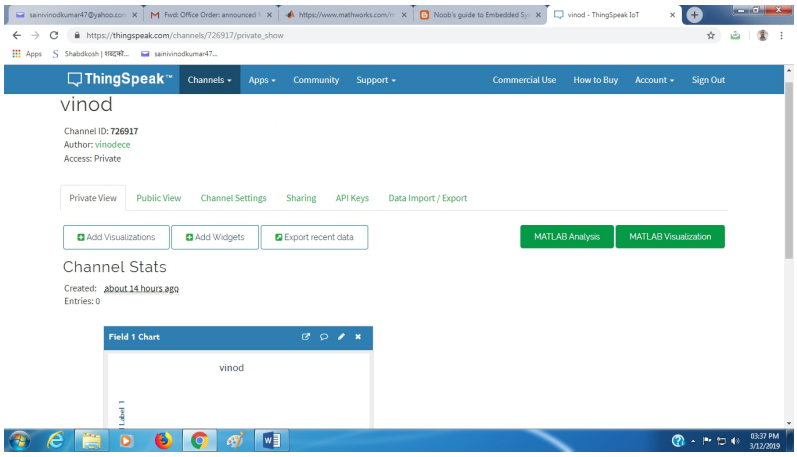


Now click on the "New Channel" button. You will have a form like this.

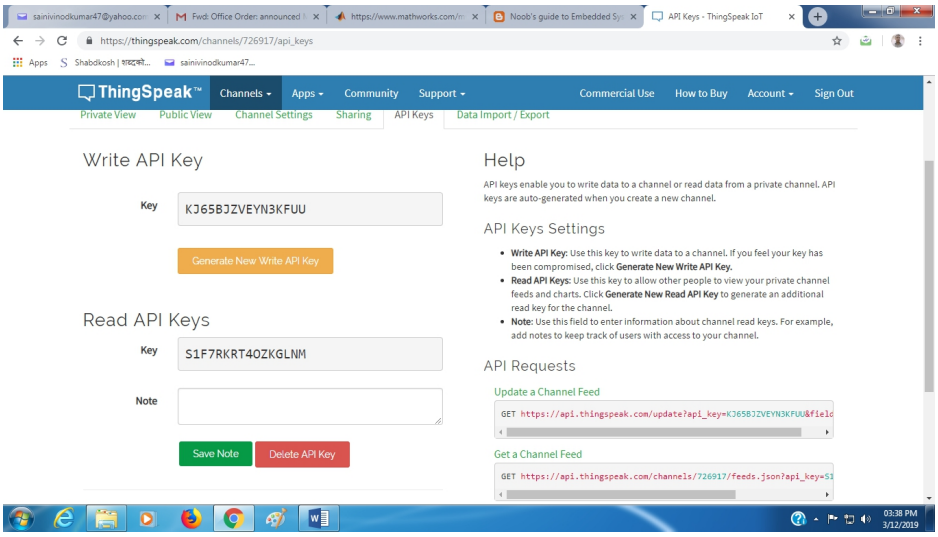


After filling up the Name you can save the channel by clicking on "Save Channel". I have given some extra parameters for understanding but it is not mandatory.

Now you will get a window like this. In this window you ca visualize your channel feed. If you want to tweak your window or add some more charts you are free to check out the documentations. For now we will proceed with the basic window.

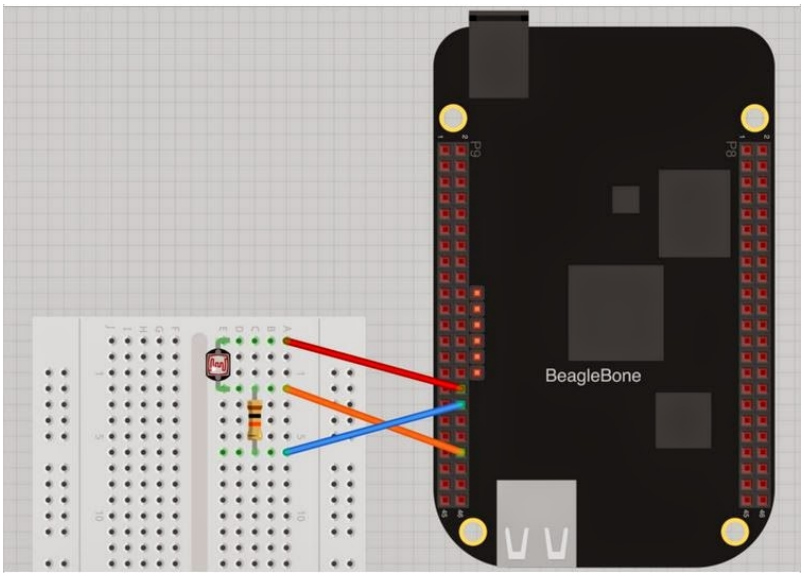


Now Click on the "API Key" tab to open a window like this



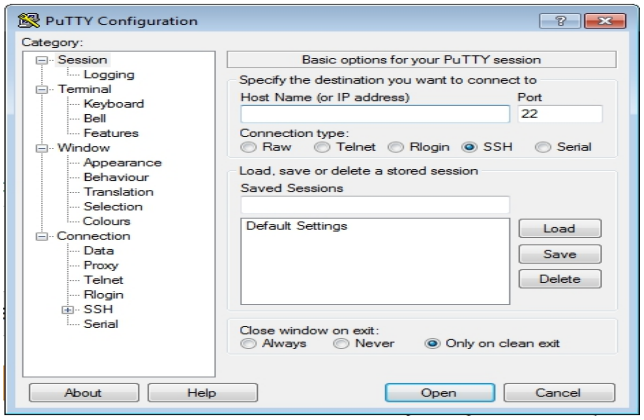
Note down the API Key. You will need this in order to connect your device to the newly created channel. I have blotted out my API Key so that it doesn't become public.

Now follow the wiring diagram as shown below

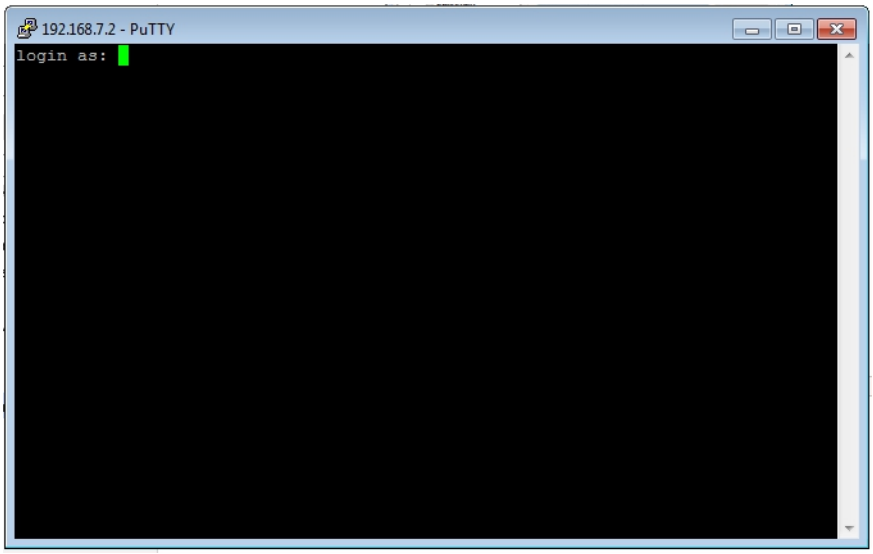


Now start up your Beaglebone Black and Log in with your ID and password. Connect your BBB to the internet.

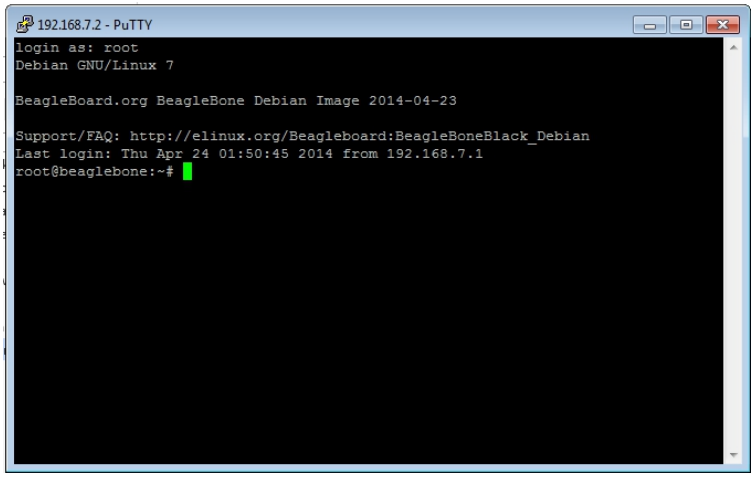
Open Putty



Host Name (IP address BBB) 192.168.7.2 type then open



Login with root



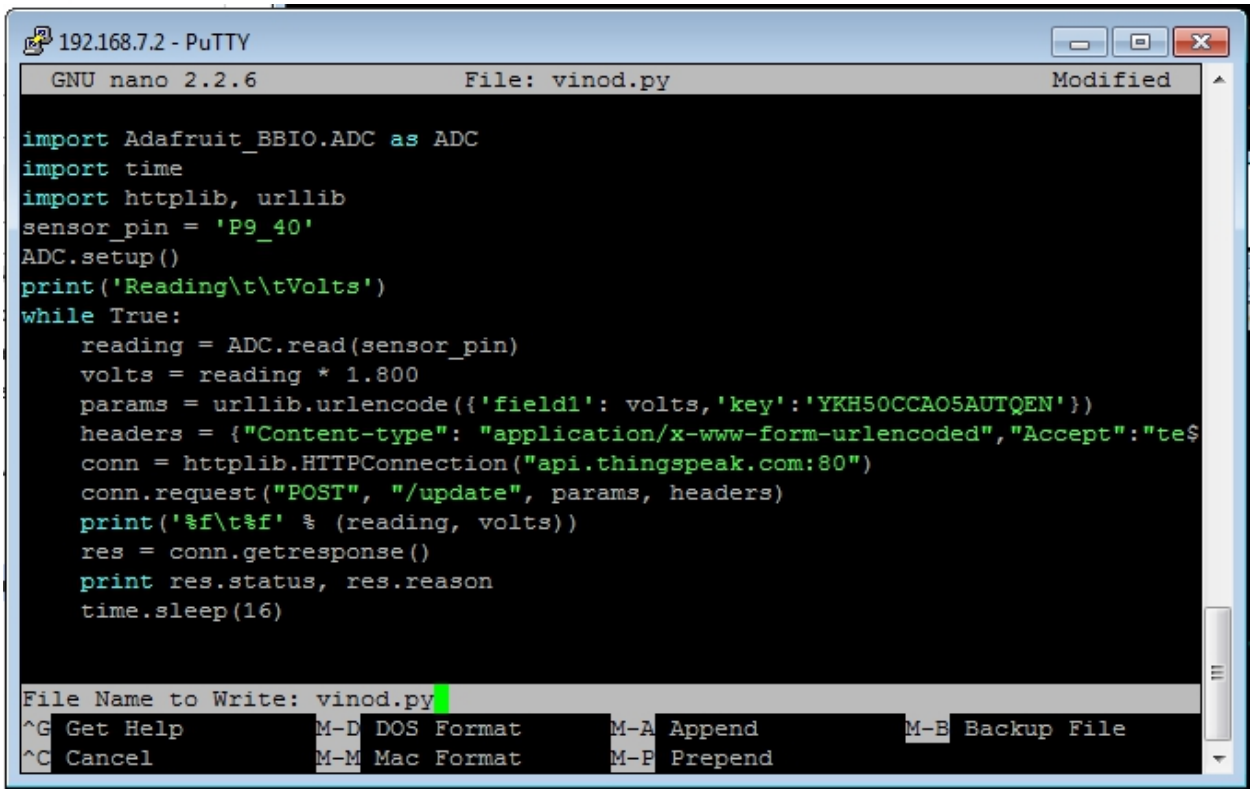
Type nano (file name).py

Now open the text editor and copy the following python script.

```
import Adafruit_BBIO.ADC as ADC
 import time
 import httplib, urllib
 sensor_pin = 'P9_40'
 ADC.setup()
 print('Reading\t\tVolts')
 while True:
    reading = ADC.read(sensor_pin)
    volts = reading * 1.800
    params = urllib.urlencode({'field1': volts,'key':'YOUR API KEY'})
    headers = {"Content-type": "application/x-www-form-urlencoded","Accept":"text/plain"}
    conn = httplib.HTTPConnection("api.thingspeak.com:80")
    conn.request("POST", "/update", params, headers)
    print('%f\t%f' % (reading, volts))
    res = conn.getresponse()
    print res.status, res.reason
    time.sleep(16)
```

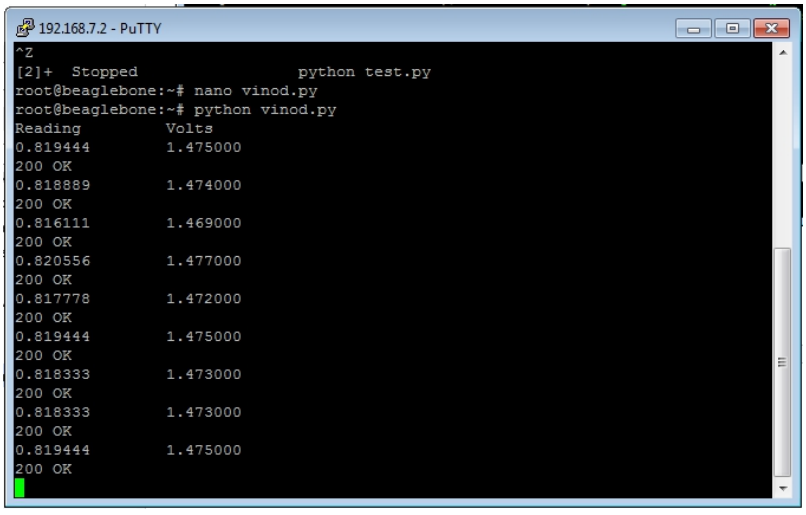
Replace 'YOUR API KEY' with the API key you noted down previously.

Now save the file with .py extension. I have named my file ldr.py



Enter, Exit (Ctrl+X) and python filename.py

Replace <filename> with the name you have given.



That's it you are done. Now check the chart on your channel Private window where the data gets updated every 15 seconds.

This is my chart getting updated here

Analysis of Results: (Write your own)

Conclusions:

Precautions:

Internet of Things Lab [ECE-3062]

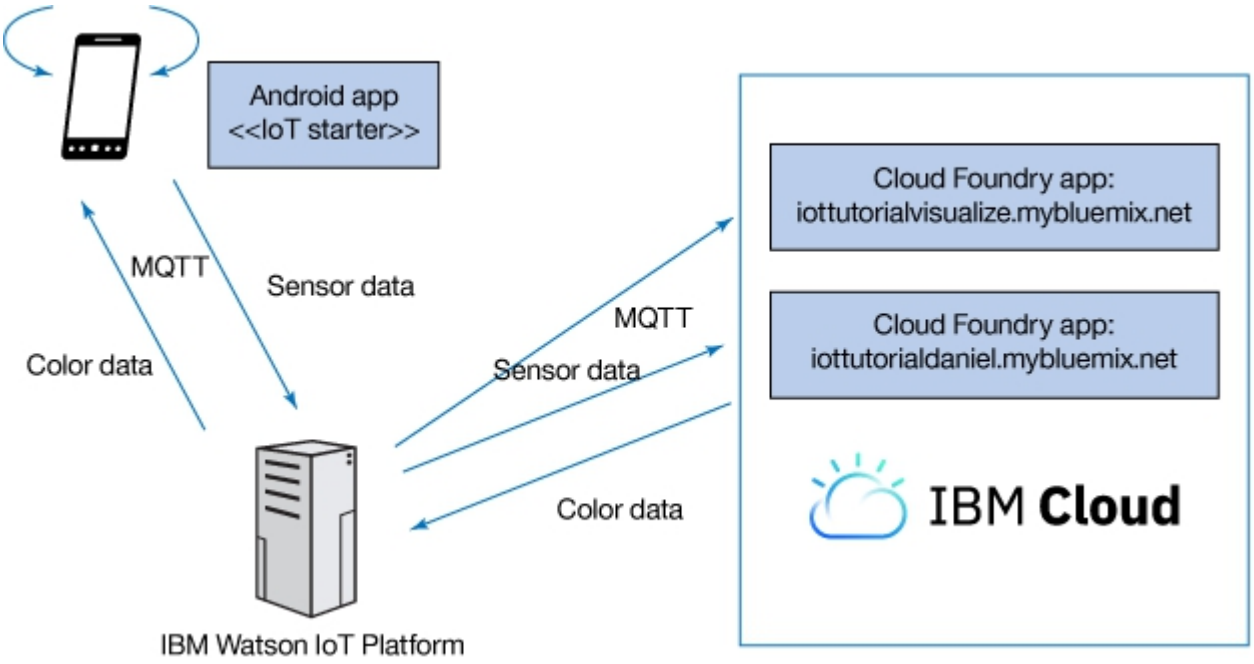
EXPERIMENT No. : 05

Aim: -Turn your smartphone into an IoT device using the IBM Watson IoT Platform cloud-hosted service.

Software Requirement: IBM Watson IoT Platform cloud-hosted service, Cloud Foundry apps, A smartphone.

Theory:-

How you can send sensor data that is generated by your smartphone to the IBM Watson IoT Platform cloud-hosted service, and then create Cloud Foundry apps on the IBM Cloud that process, visualize, and store the data. Lastly, it shows you how to create an Android application for a smartphone. Here is an overview of the architecture:

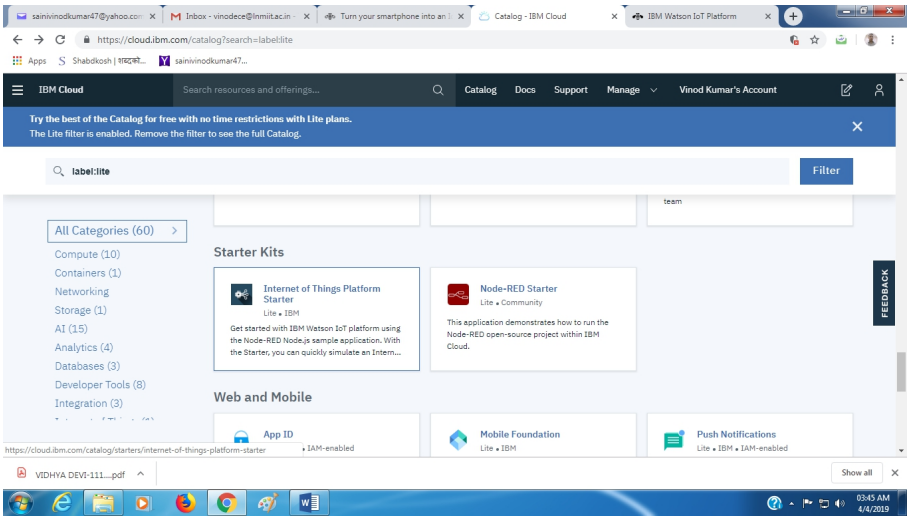


Procedure:

Create an IoT app in the IBM Cloud

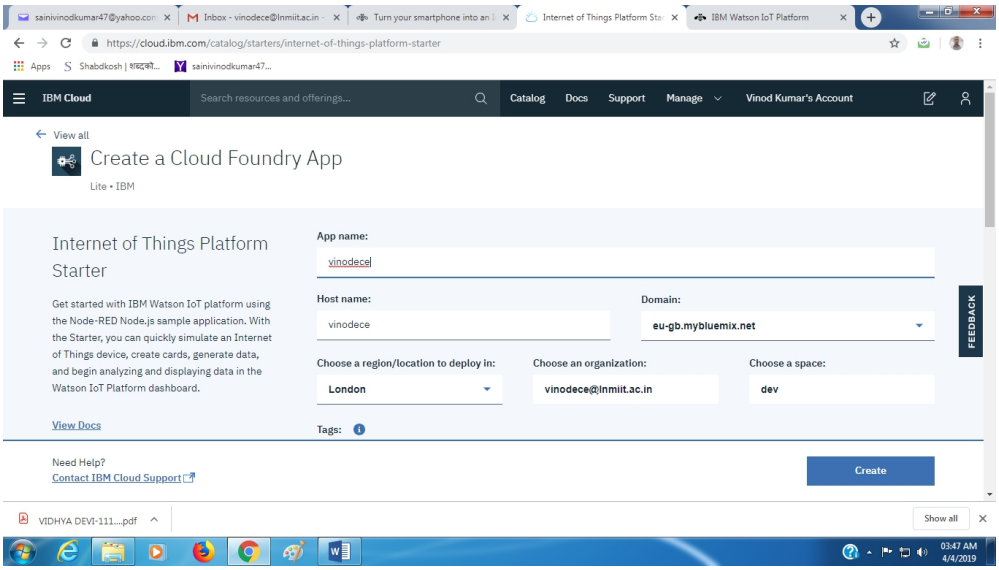
The Internet of Things Platform Starter boilerplate contains a Node-RED engine that you will use later to process IoT messages. For now, you will create an Internet of Things service to send and receive MQTT messages to and from the IBM Watson IoT Platform.

1. Log in to your IBM Cloud account.
2. Click **Create Resource**.
3. In the Catalog, under Platform, click **Internet of Things Platform Starter**.

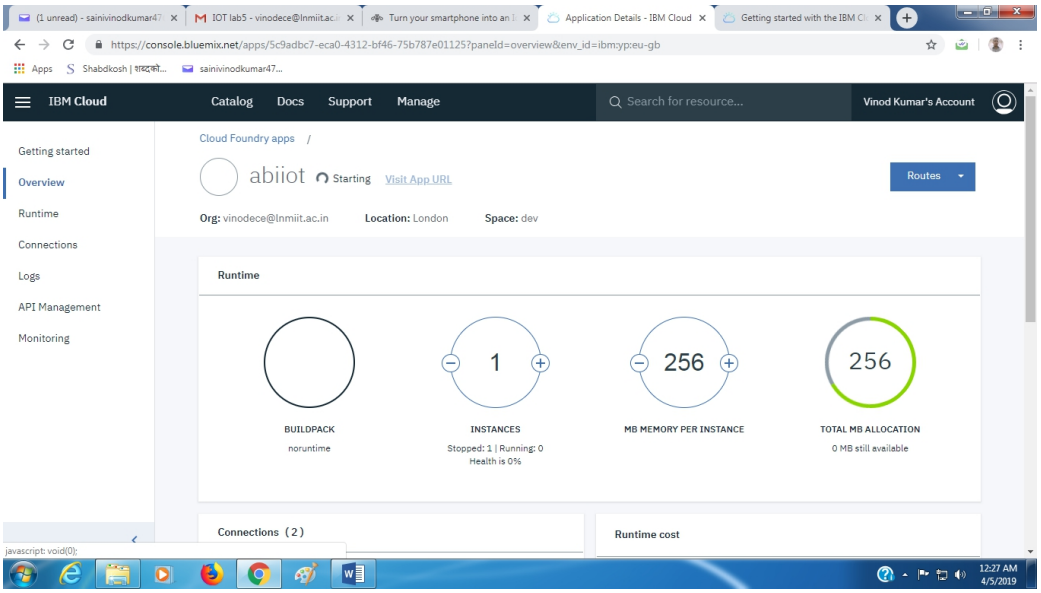


4. Enter a name for your application. Because this name is also used as the host name, it must be unique within the IBM Cloud, for example, “iot“. For example, I might use iottutorialdaniel.

Click **Create**.

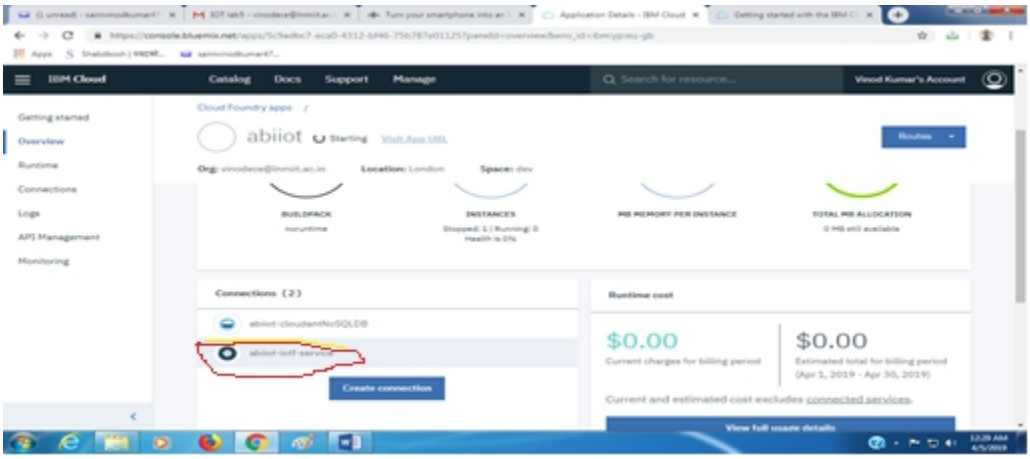


After your app is created, in the left pane, click **Overview**. Notice that your app contains two connections, one to a **Cloudant NoSQL database** and another to an **Internet of Things Platform service.**

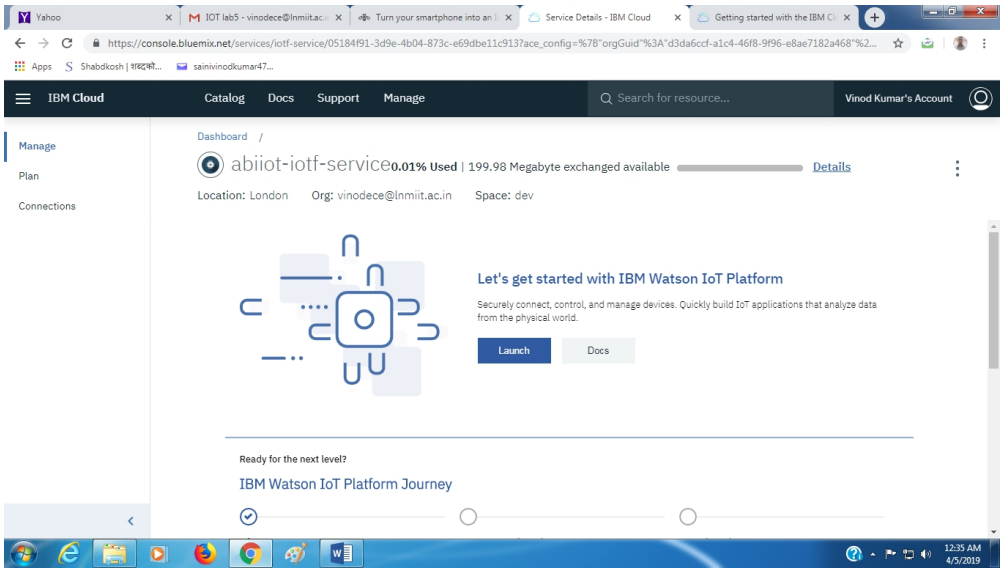


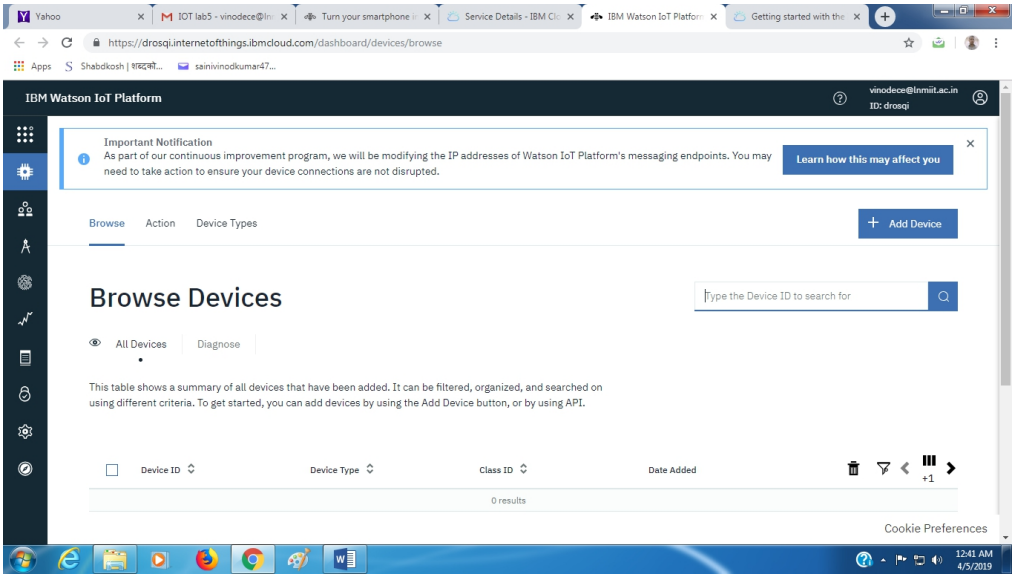
Add a device that will send MQTT messages to the Watson IoT Platform

In the Overview view of your app, under Connections, click the Internet of Things Platform service, named something like iot<_your name_="name_">-iotf-serviceiot<your name>-iotf-service.

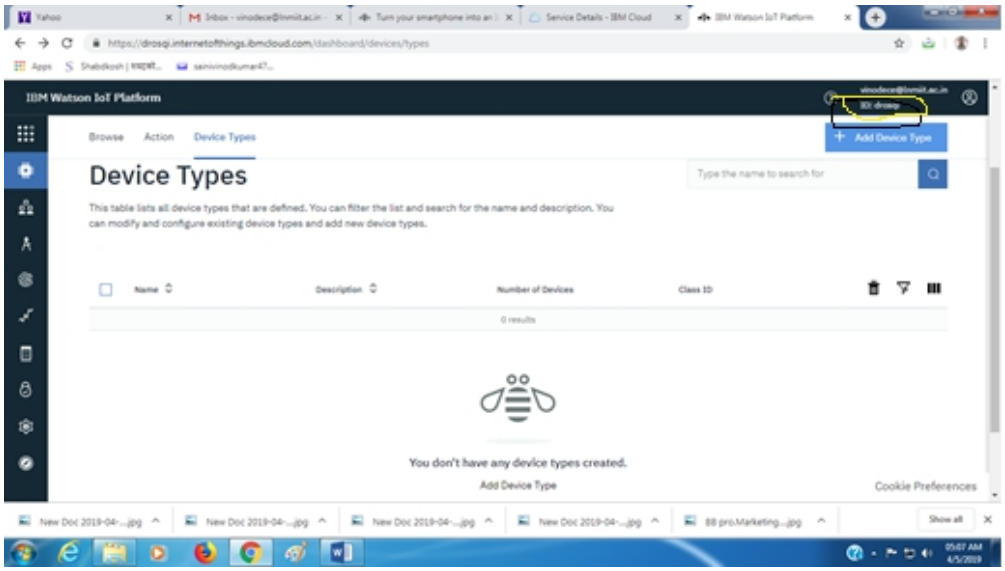


Click **Launch** to open the Watson IoT Platform dashboard.

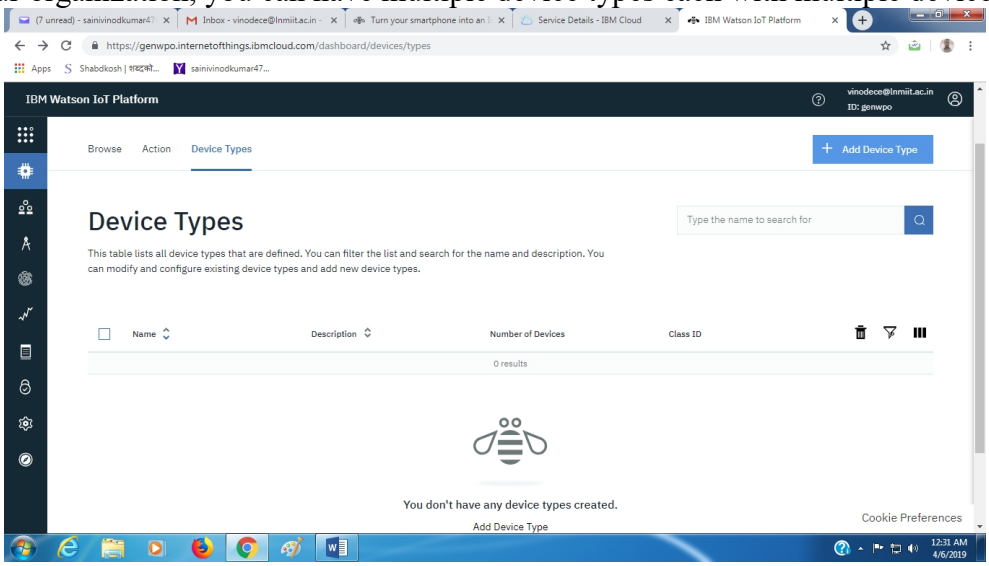




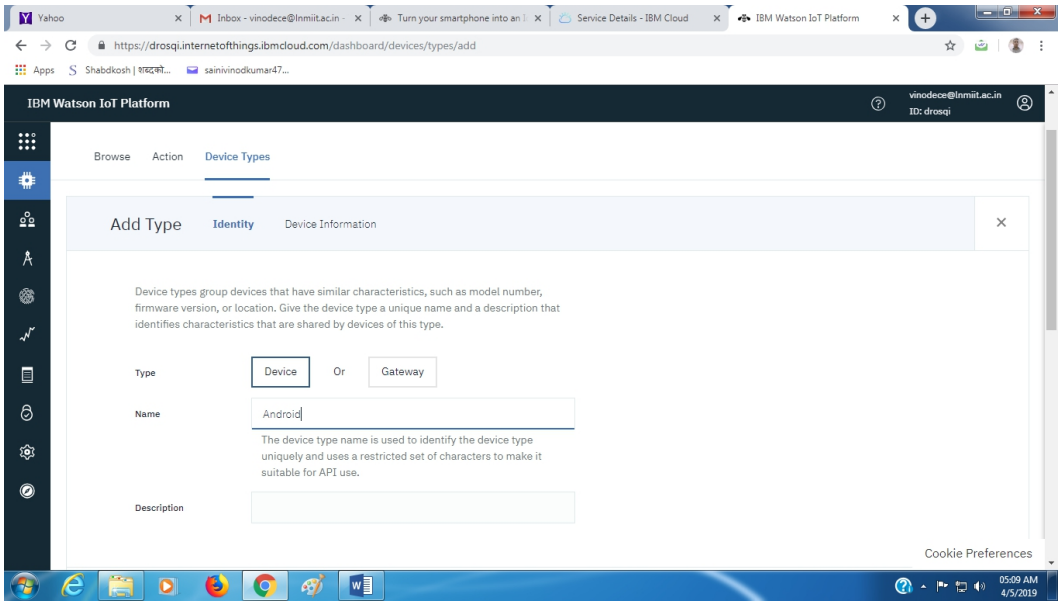
The IBM Watson IoT Platform dashboard is displayed, which is a service that is independent of the IBM Cloud. An organization ID is assigned to your app, and you will need this ID later when developing the mobile app. In the following image, the organization ID is **drosqi**,which is displayed under your login information in the upper right corner of the dashboard.



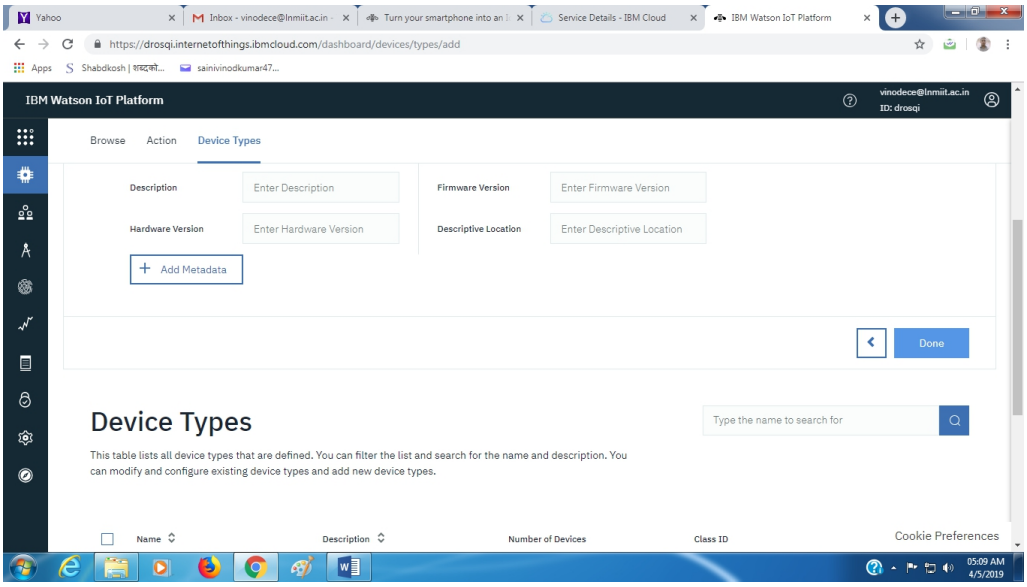
On the left menu, which pops out when you hover over it, click **Devices**. Then, click **Add a device type**. In your organization, you can have multiple device types each with multiple devices.



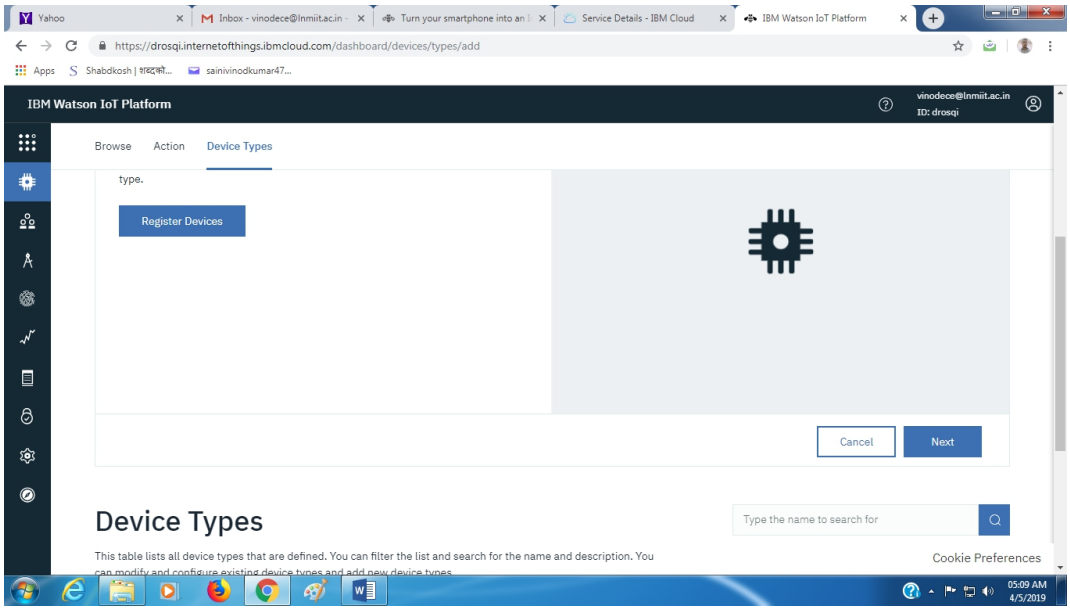
A *device type* is a group of devices that share characteristics; for example, they might provide the same sensor data. In our case, the device type name *must* be “Android” (this device type name is required by the app that you will use later).



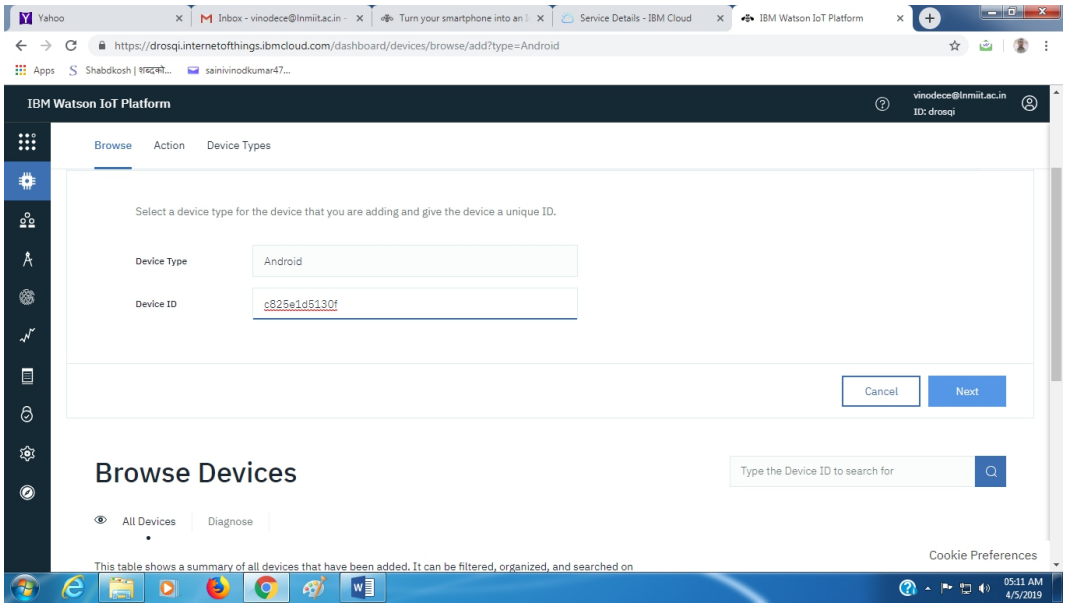
1. Click **Next**.



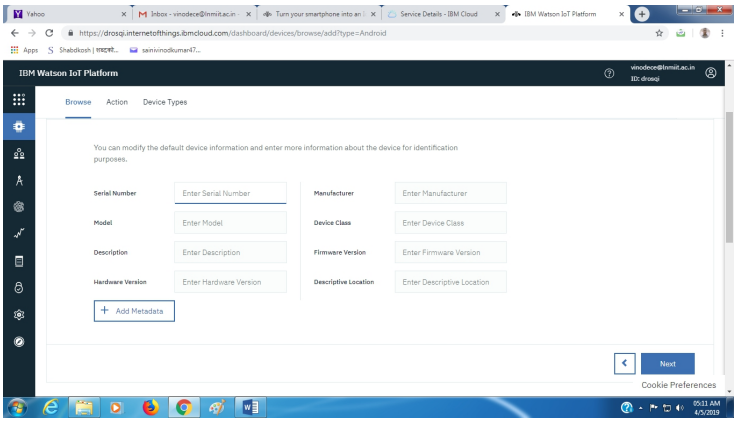
A page is displayed where you can enter metadata about the device type, such as a serial number or model. You don't need to specify this information for this demo. Just click **Done**.



1. Click **Register Devices**. Enter the device ID. The *device ID* can be, for example, the MAC address of your smartphone. However, it must be unique within your organization only. Therefore, you might enter, as I did here, something like "----------".

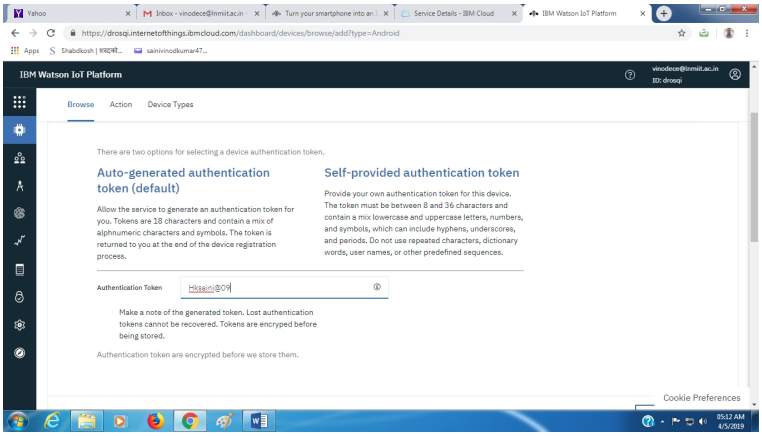


Click **Next**. A page is displayed where you could enter metadata about the device. Leave it blank, and click **Next**.

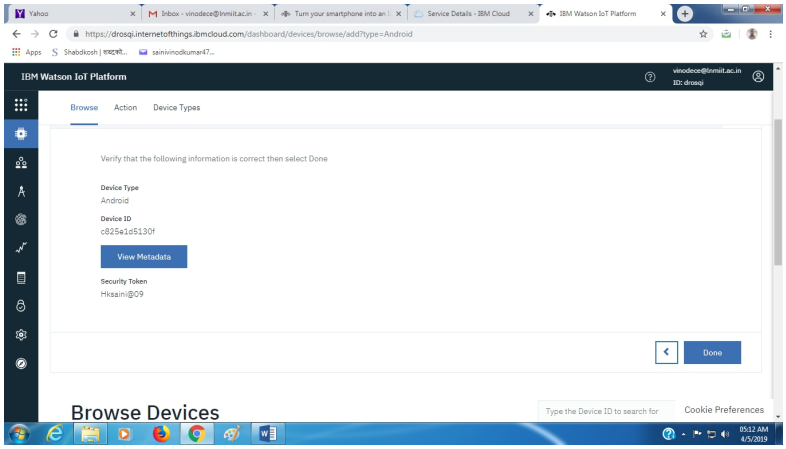


Click Next.

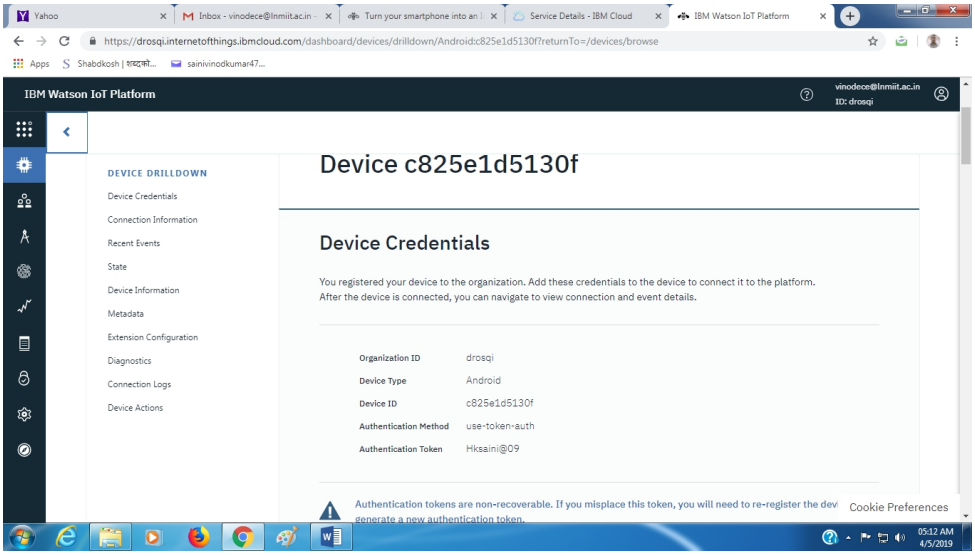
Provide a value for the authentication token. Remember this value for later. Then, click **Next**.



Click **Done**.



Click Next. A page is displayed where you could enter metadata about the device. Leave it blank, and click Next.
Provide a value for the authentication token. Remember this value for later. Then, click Next.



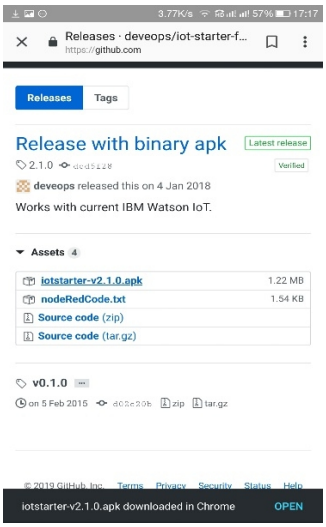
Click **Back**.

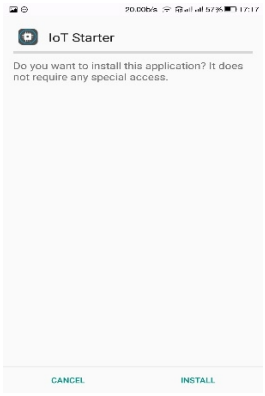
Now you are ready to send MQTT messages from a device to the IBM Watson IoT Platform.

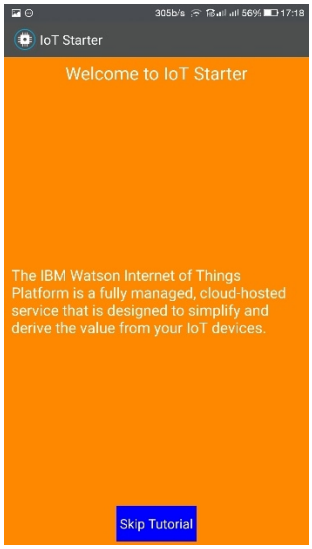
Configure App on Android Phone

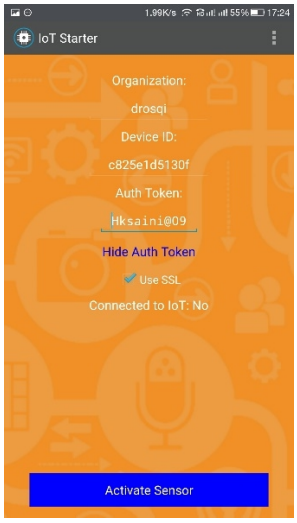
1. On your phone, go to **Settings** > **Security**. Under Device Administration, enable **Unknown sources**. Now you can install .apk files from outside of Google Play.
2. Open the browser on your phone, and enter this URL:

`https://github.com/deveops/iot-starter-for-android/releases`



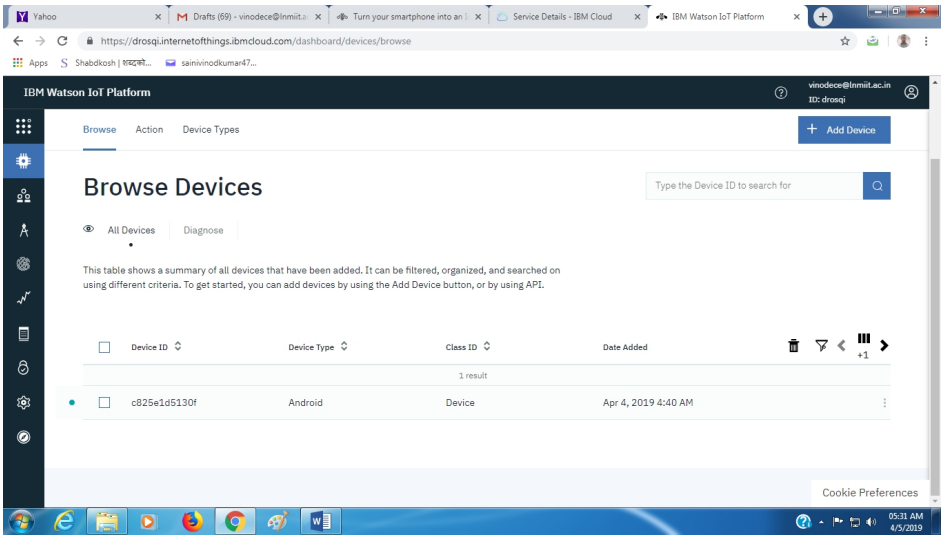

IoT Starter
Do you want to install this application? It does not require any special access.
CANCEL
INSTALL


IoT Starter
Welcome to IoT Starter
The IBM Watson Internet of Things Platform is a fully managed, cloud-hosted service that is designed to simplify and derive the value from your IoT devices.
Skip Tutorial

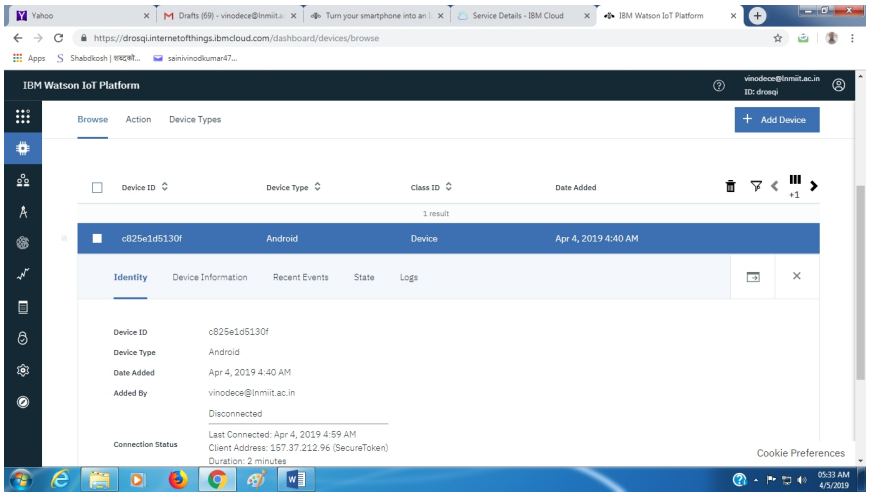

IoT Starter
Organization:
drosqi
Device ID:
c825e1d5130f
Auth Token:
Hksaini@09
Hide Auth Token
Use SSL
Connected to IoT: No
Activate Sensor

Verify that messages are being sent from your smartphone to the Watson IoT Platform

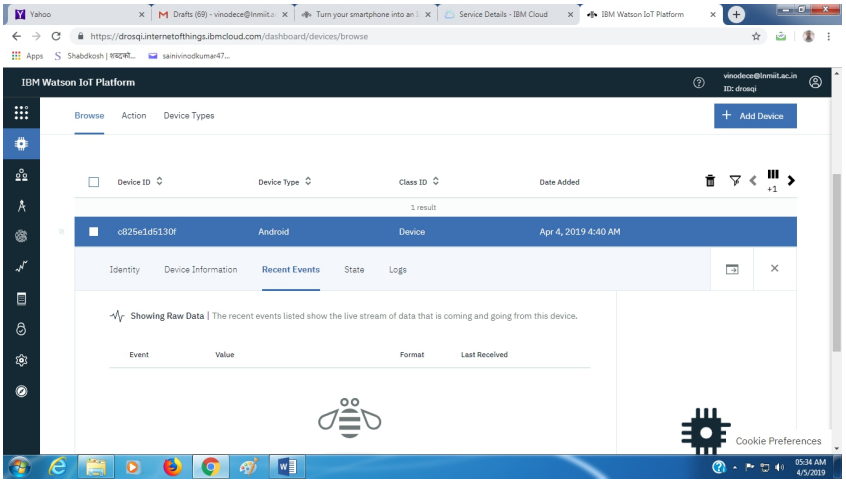
1. Back on your computer, open the IBM Watson IoT Platform page for your organization again (see the start of).
2. In the left menu, click **Devices**. Your Android device is displayed.



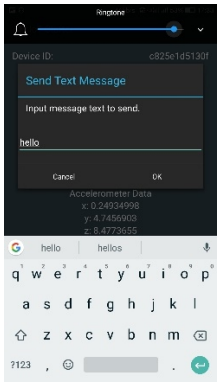
Click the Device ID, and then go to the Recent Events tab. You should see events coming from your smartphone.

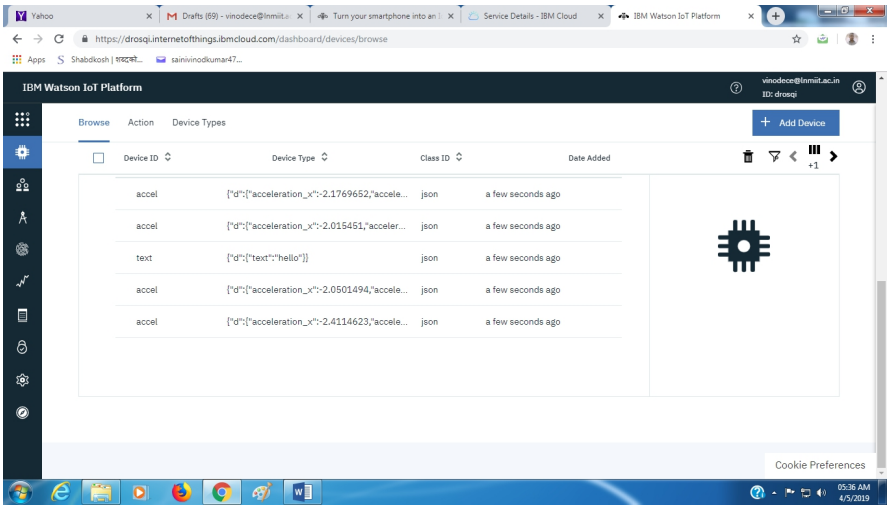


Click the Device ID, and then go to the Recent Events tab. You should see events coming from your smartphone.

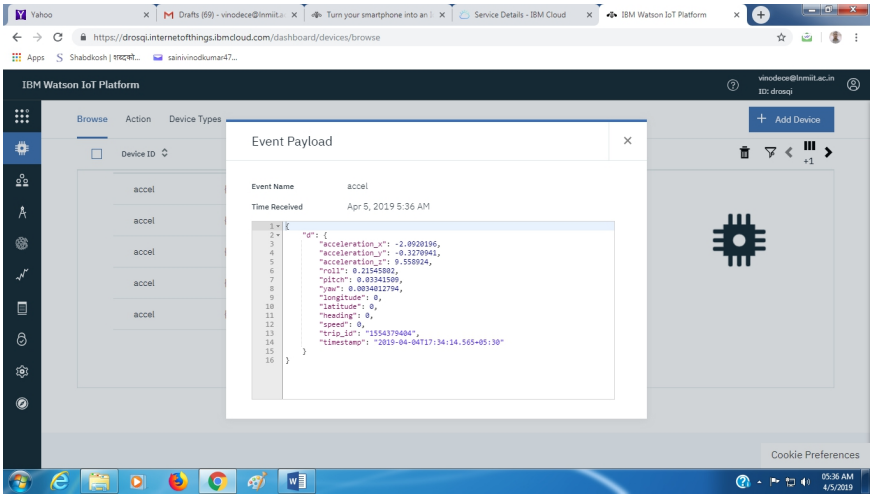


Click one of the events. The messages that are sent from your smartphone are in JSON format. They contain acceleration and position data.





Click one of the events. The messages that are sent from your smartphone are in JSON format. They contain acceleration and position data.



Now you are ready to work with the message data on IBM Cloud.

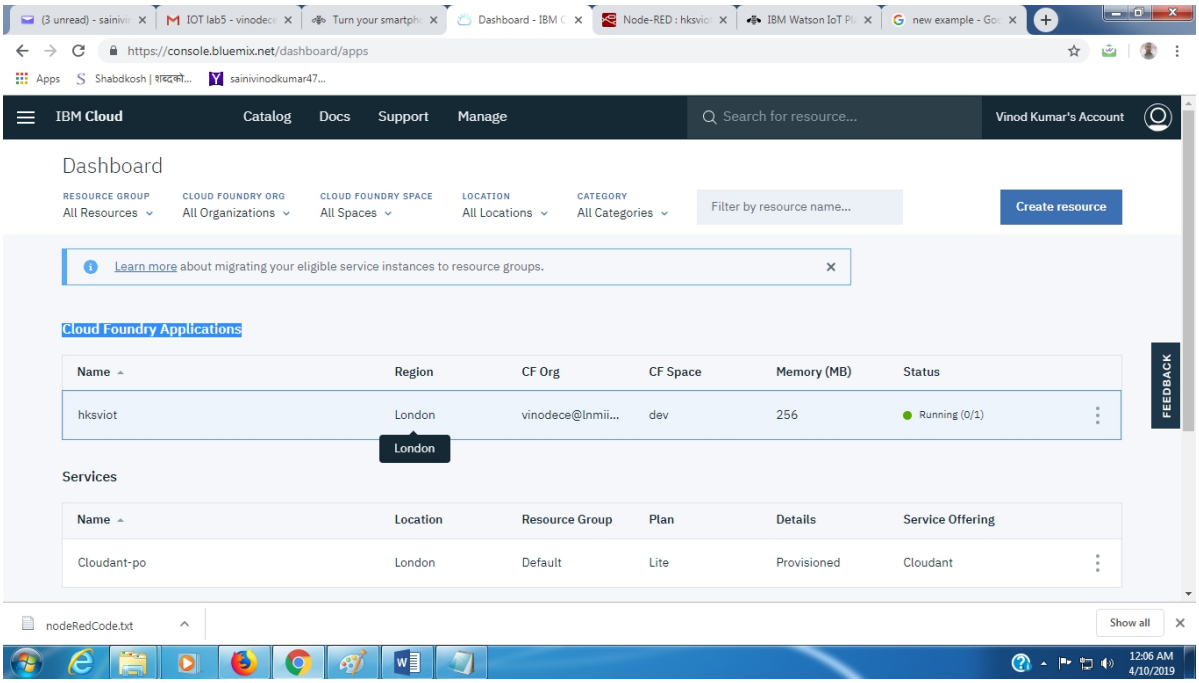
Process messages in a Node-RED flow

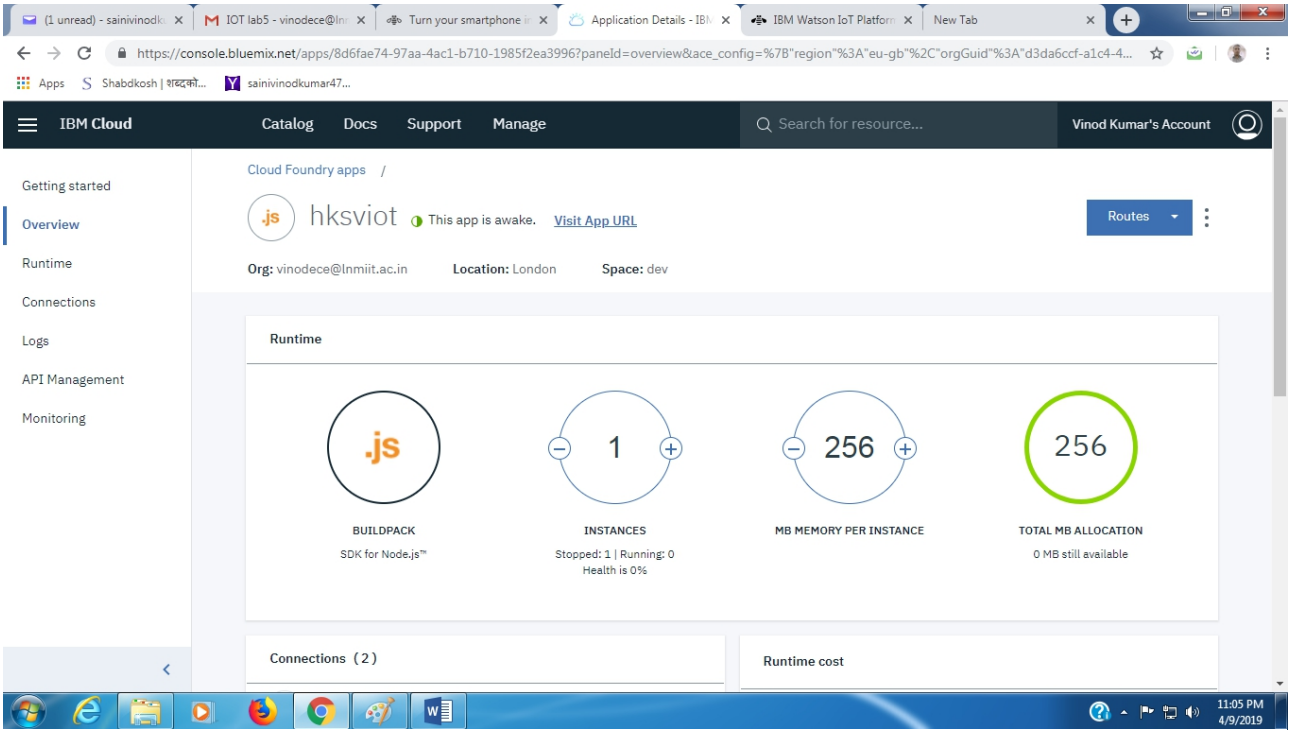
Open your go IBM Cloud dashboard,

Cloud Foundry Applications Click

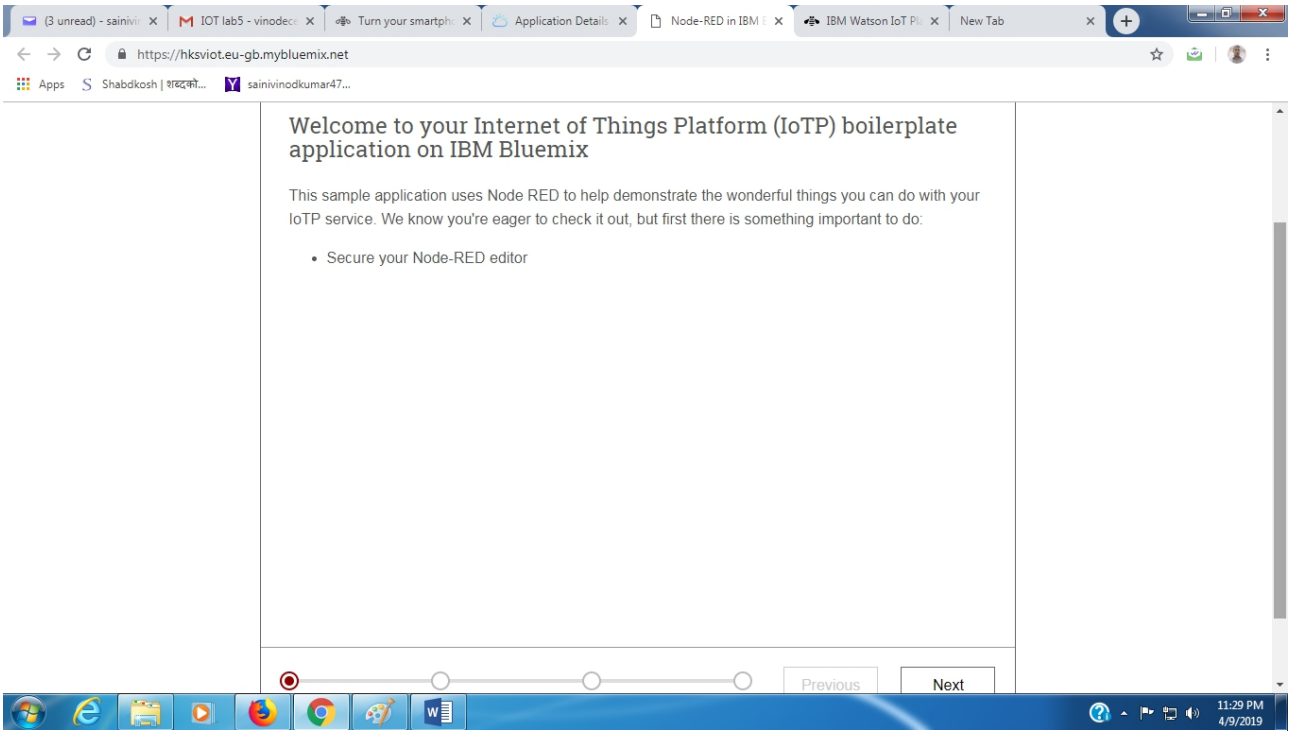
The LNM Institute Of Information Technology
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

LNMIIT
The LNM Institute of Information Technology



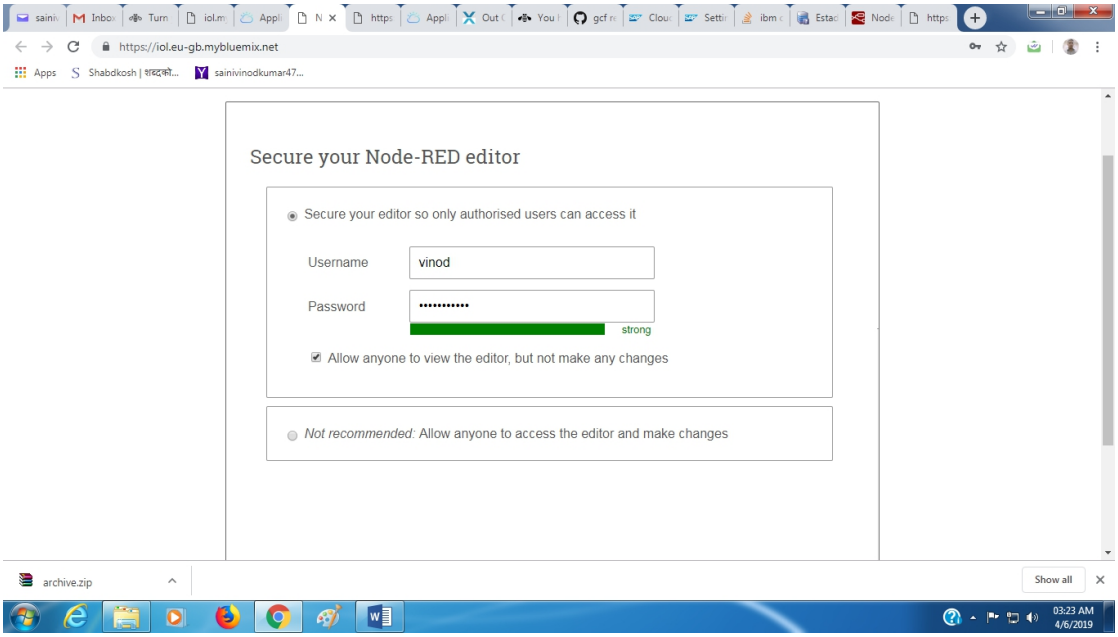


Visit App. URL

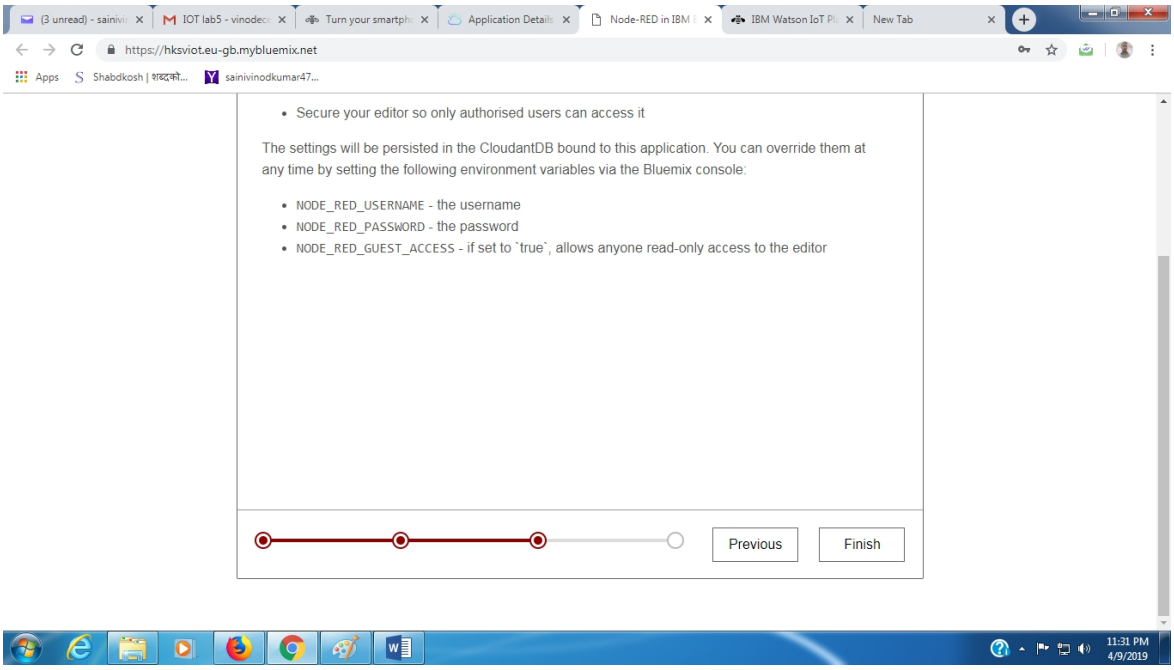


Next

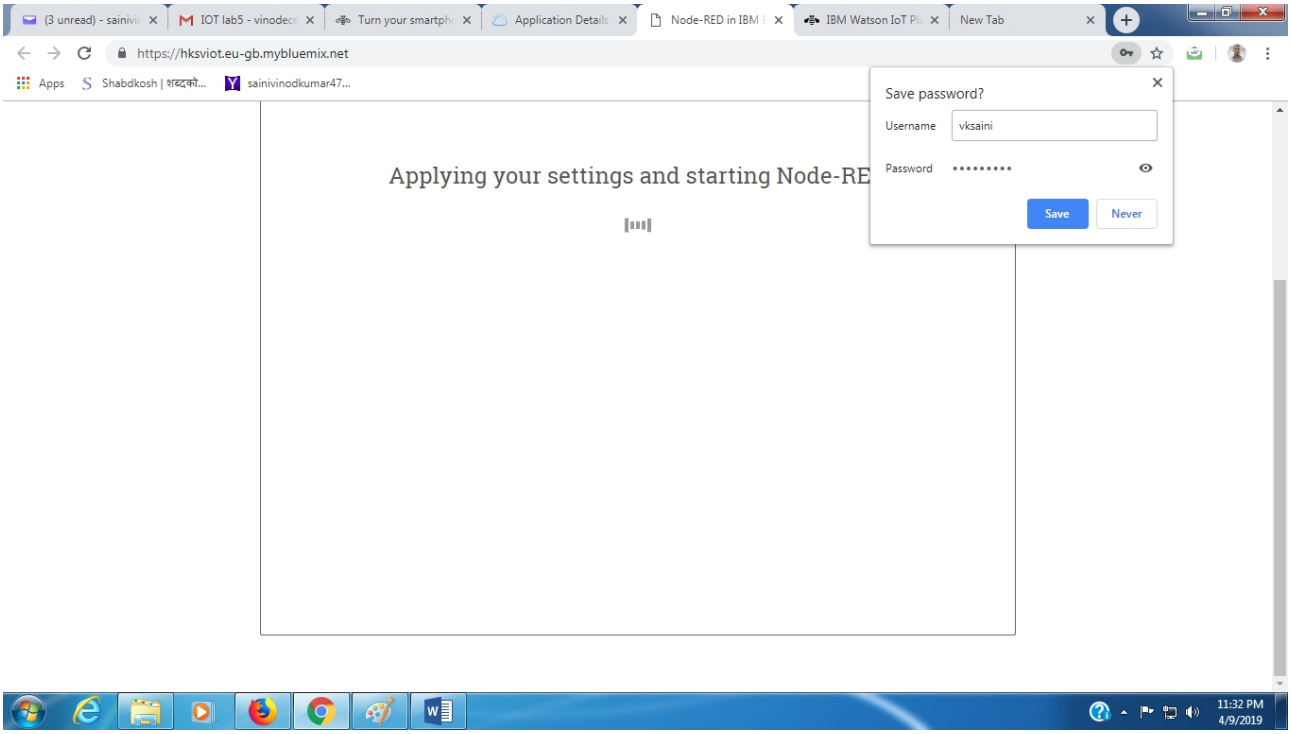
Follow the wizard steps to set a user name and password for the Node-RED editor.



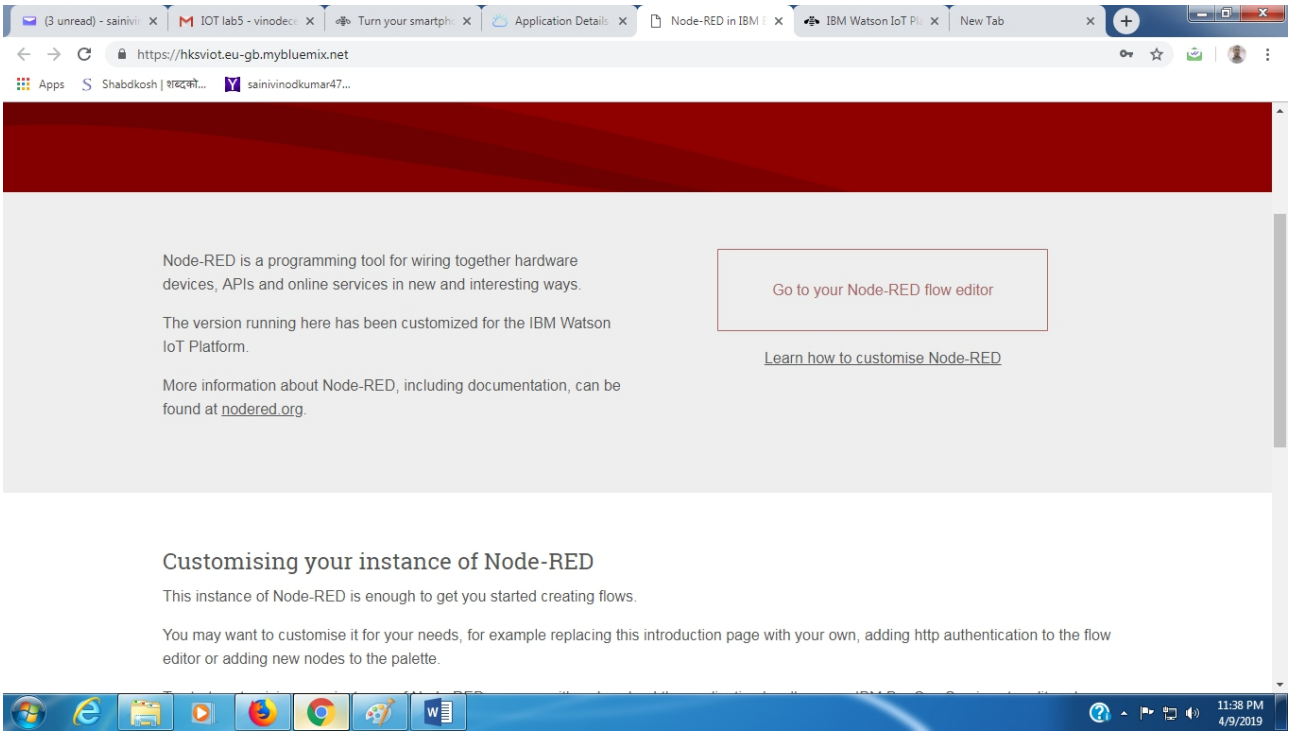
Next

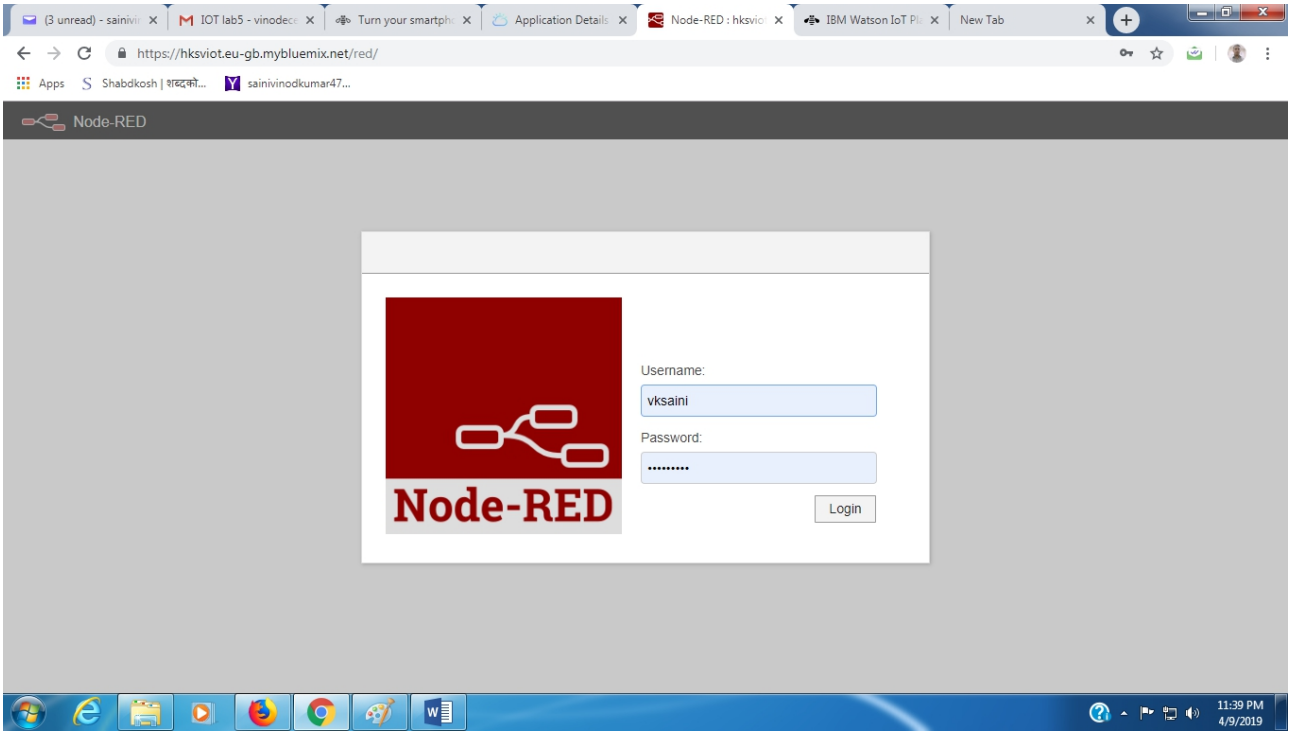


Finish
Save Password

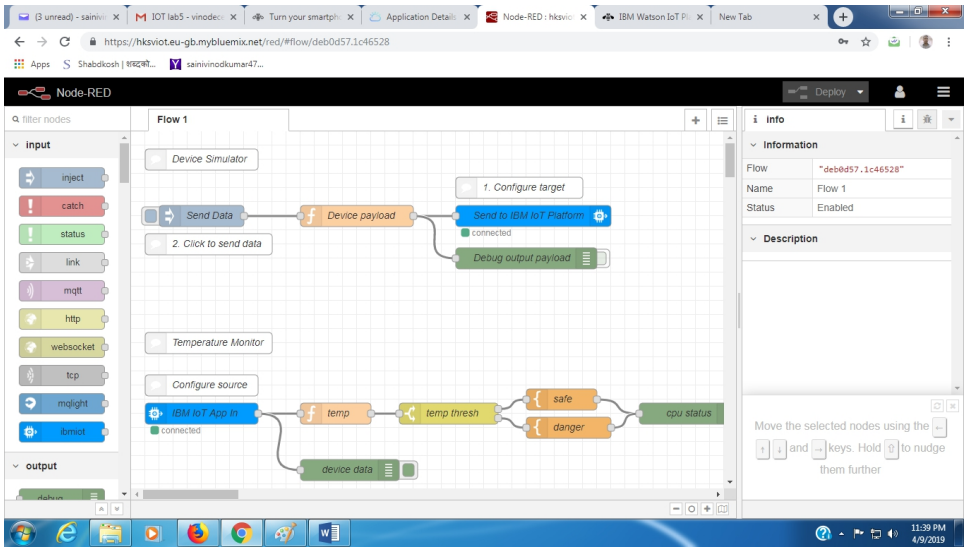


On the Node-RED page for your IoT app, click **Go to your Node-RED flow editor**. The editor opens, containing a sample flow.

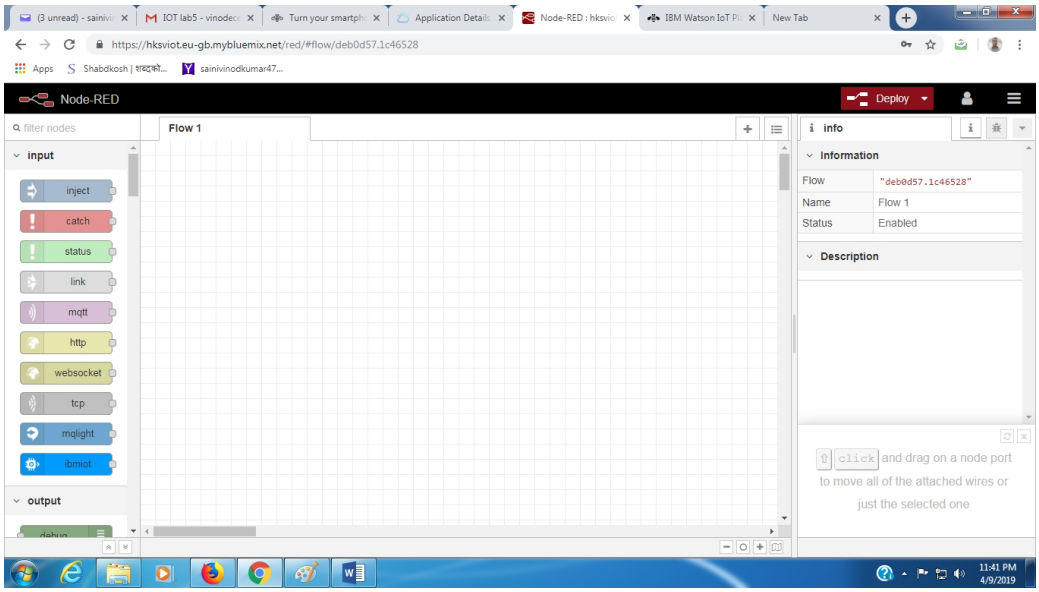




Click to Login

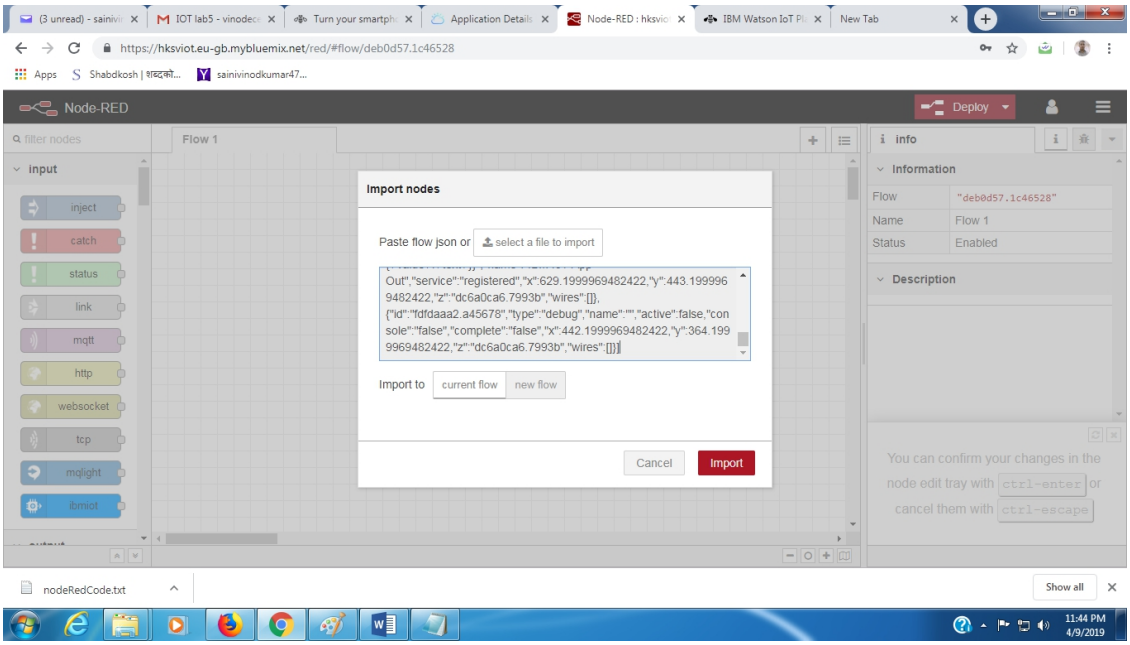


Using the drag-and-drop features of this editor, you can plug together a flow of messages. Although you can create your own flow here, we will import the code below. But first, select all existing nodes, and delete them by pressing the Delete key.

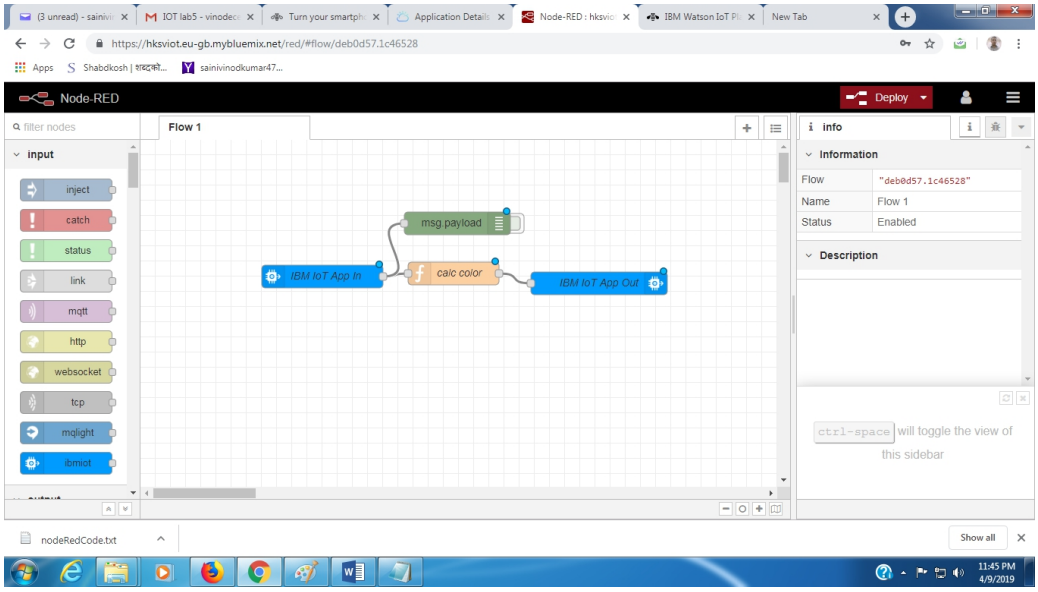


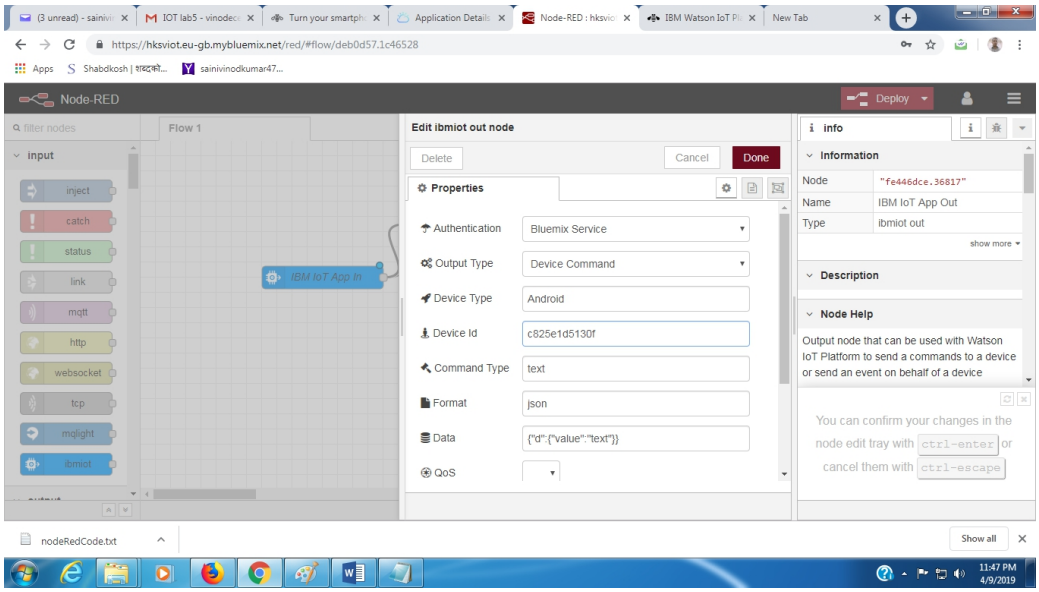
1. Download the following code (as a long single line of code) as a text file (`nodeRedCode.txt`) from **GitHub**.

In the Node-RED editor, press **Ctrl-I** to open the Import Nodes dialog. Paste the code, and click **Import**.

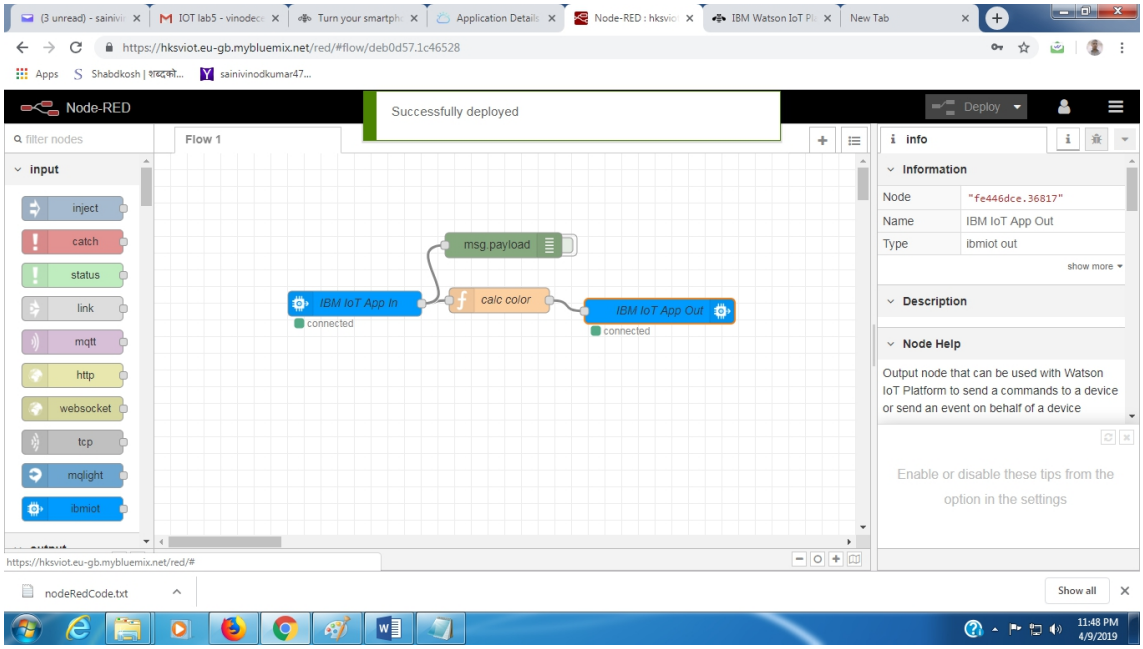


Now you need to adapt the flow to your specific parameters. The only relevant parameter is the Device ID. Double-click the node **IBM IoT App out**. In the pop-up window, enter the Device ID that you used earlier (for example, `112233445566`), and click **Import**.

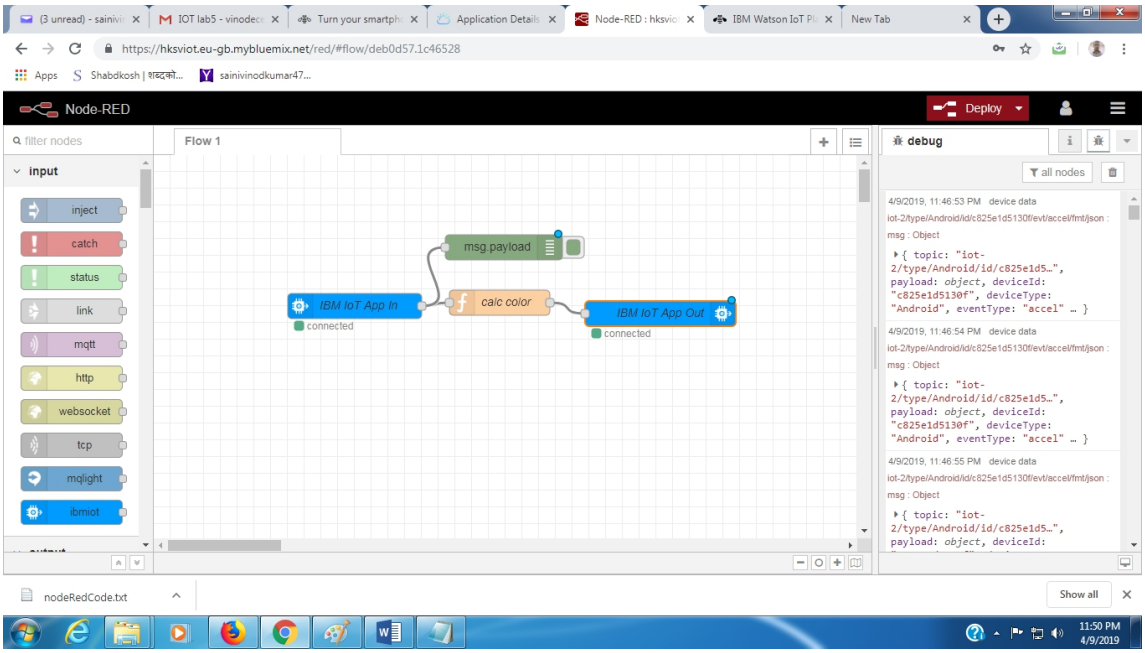




Click **Deploy** in the flow editor. The flow is deployed and should be active immediately.



Move your smartphone around; flip and tilt it. The background color of the app on your phone should now change colors, depending on the orientation of the z-axis.



Result:-

Conclusions:

Precautions: