Smart Energy Meter

**1. INTRODUCTION:**

1.1 Overview:

Tele-measurement (or Smart Metering) is one of the most popular domains in the Internet of Things. Remote reading of energy consumption (water, gas, electricity) of course, but in general, the remote measurement of any parameter available on a machine, in a room, an indoor or outdoor environment. The Internet of Things saves time and money by automating remote data collection. In addition, the analysis of these data makes it possible to identify problems or anomalies, the preventive treatment of which significantly improves the operational processes.

1.2 Purpose:

The aim of this project is to create a smart energy meter whose values are sent and stores in cloud and also be visible in mobile and web appliactions. This smart meter also allows the user to switch the meter off when the power value crosses a fixed threshold value.

**2. LITERATURE SURVEY:**

2.1 Existing Problem:

The existing problem with energy meters is that the user has to constantly check on the values to find out the power consumption and whether or not the value has crossed the threshold value.

2.2 Proposed Solution:

The need of the hour to solve this energy crisis and to make sure of minimum energy consumption, the energy meters are to be automised and be monitored from anywhere. This brings in the need for a smart energy meter which is accomplished by IoT and cloud database.

Features:

* Measuring the voltage and current of the meter box
* If the current and voltage crosses the threshold value, switch of the meter by using mobile app
* store the sensor values in the database
* create a web and mobile application to display the sensor value
* send the notification if the sensor value crosses the threshold.

3. Theoretical Analysis:

Hardware --> data sent to cloud -->cloud data to be accessed by mobile and web application.

Any command given inmobile or web applications--> sent to cloud --> to the code running the hardware --> harware part responds accordingly.

The flow is created using NodeRed. HTTP and MQTT protocals are used to establish the connections.

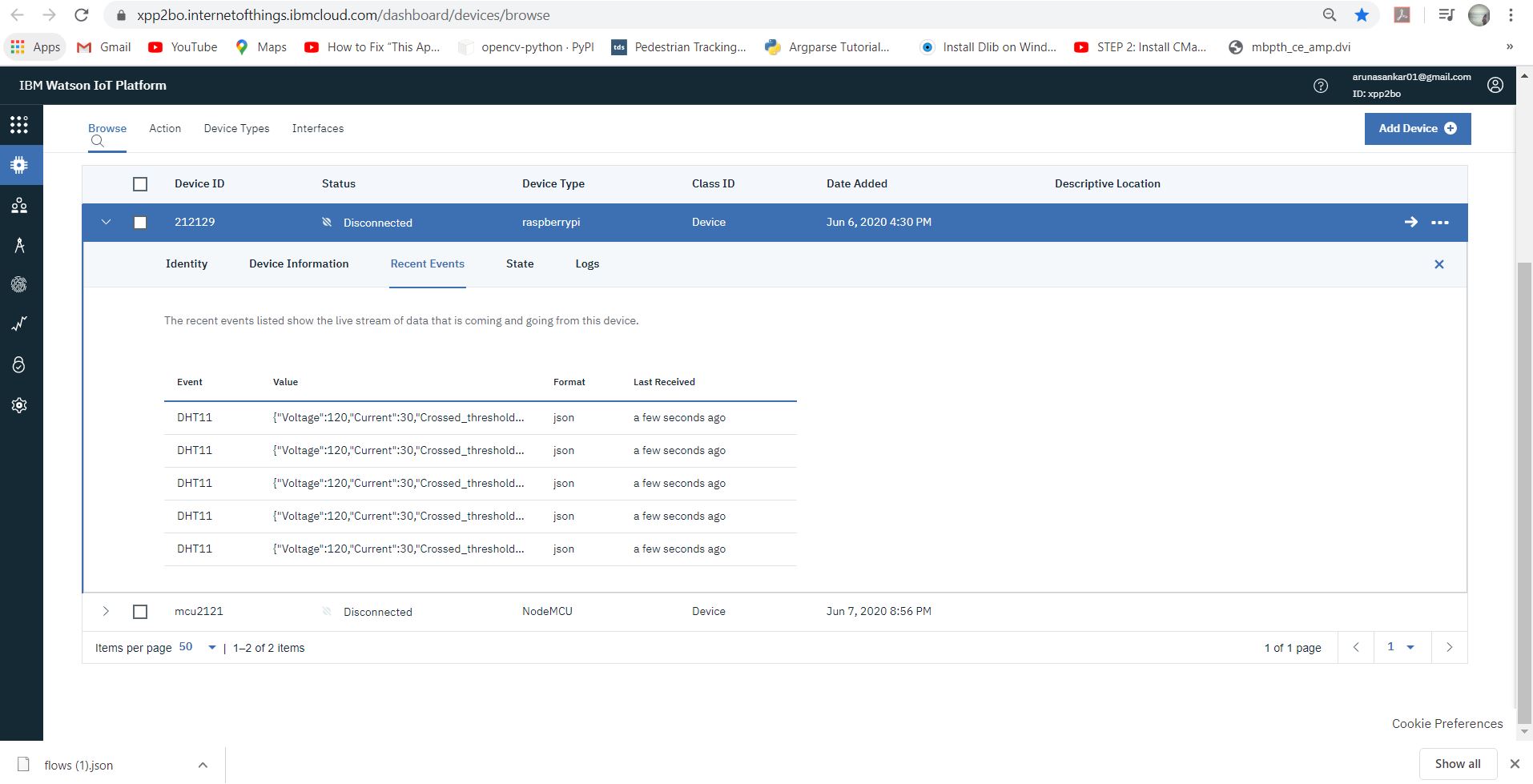
4. EXPERIMENTAL INVESTIGATION

Step 1:

Measuring the voltage and current values from the hardware segment is achieved by the python code.

From these two readings, power consumption is calculated. When power consumption exceeds the given threshold value, the mobile application will notify the user.

After reading the values from the hardware. these values are uploaded on IBM Cloud in the IBM Watson IoT Platform.



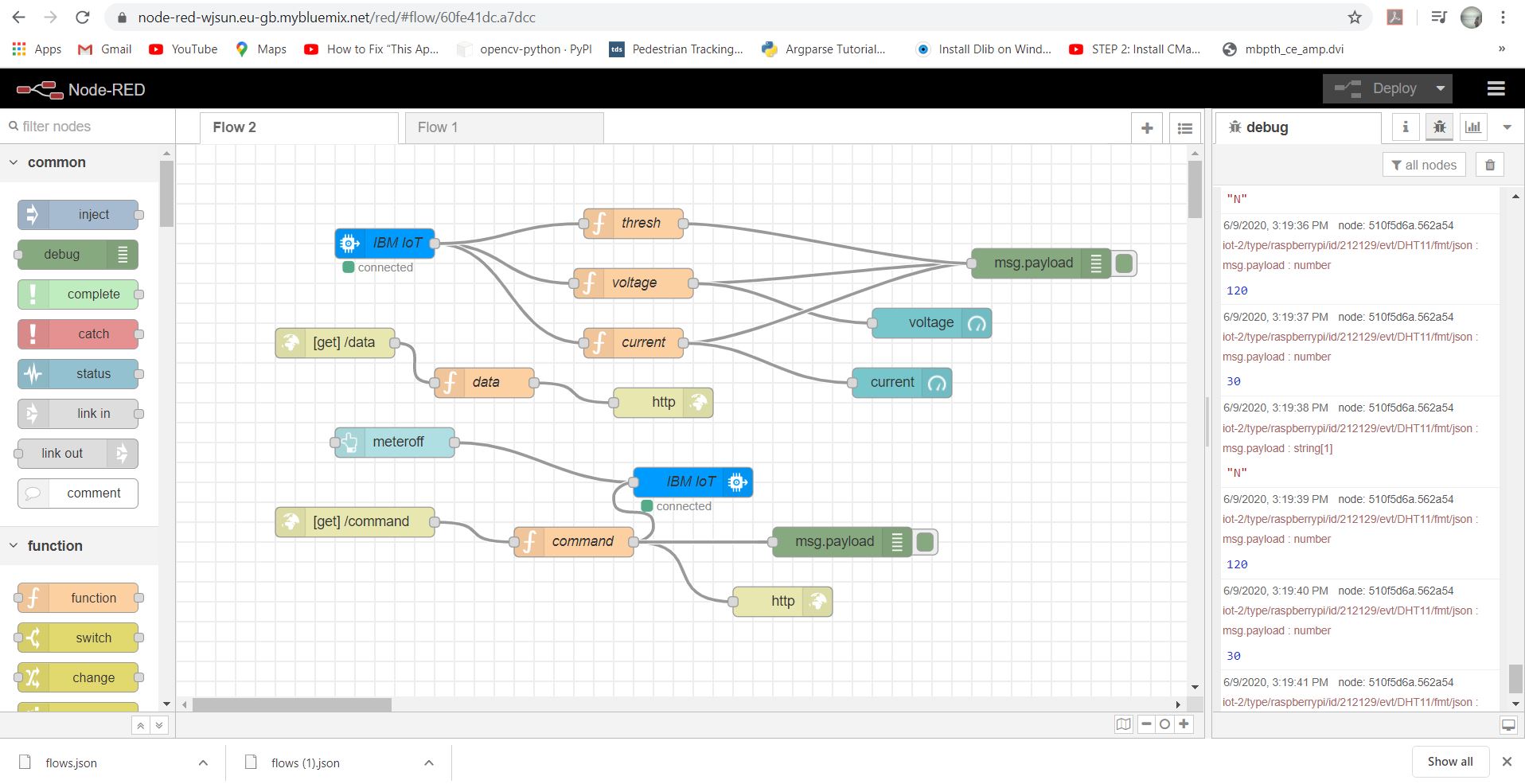
After this, a node red flow is created, where the input is obtained from IBM iot platform and the values of current and voltage, in addition to whether or not the power consumption is within threshold value are displayed.

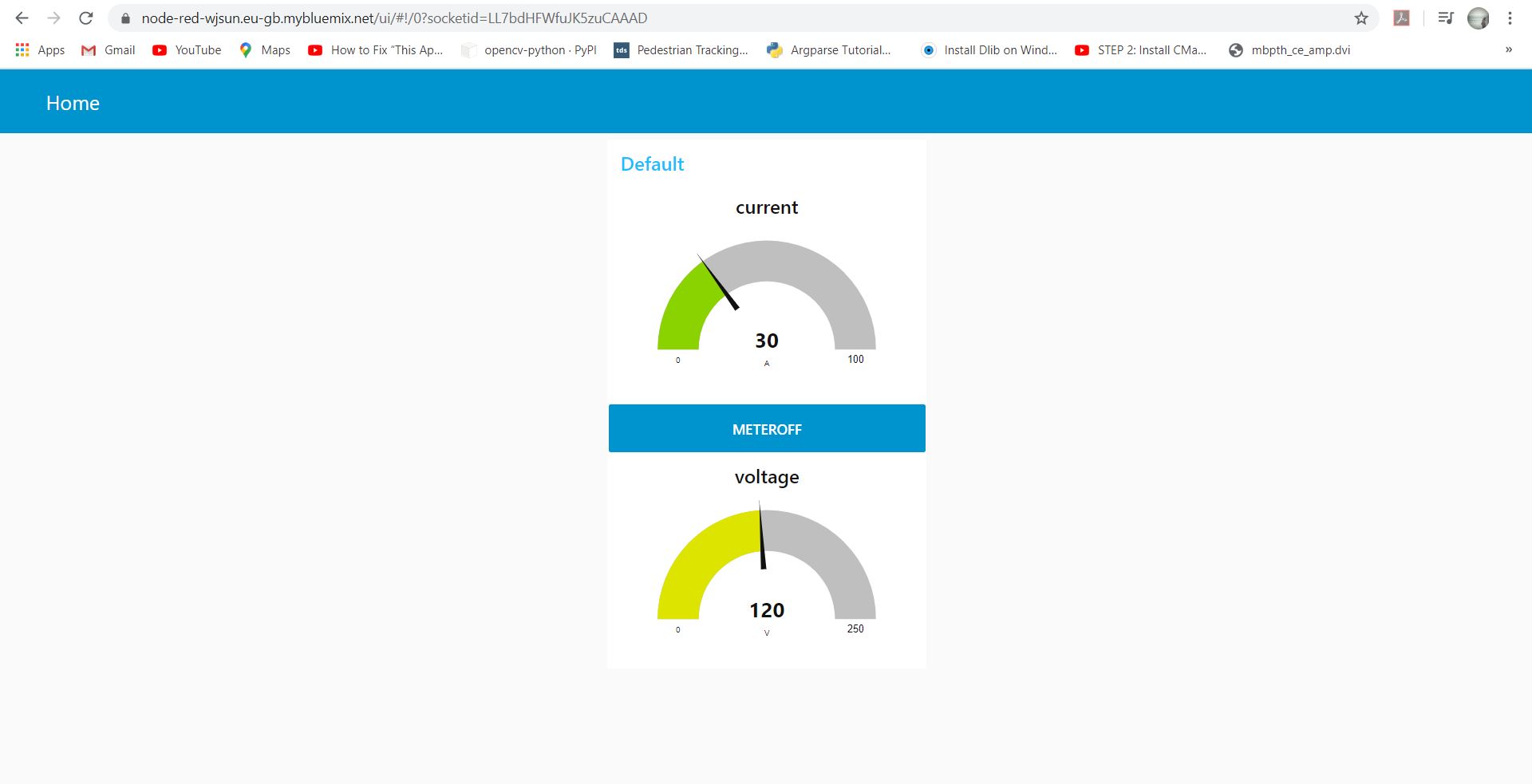
HTTP node is used to get the data from cloud and display it under a url which is later used in the MIT app inventor.

A UI is created to monitor the values using gauge tool. A button is provided to switch off the meter when the value exceeds the threshold.

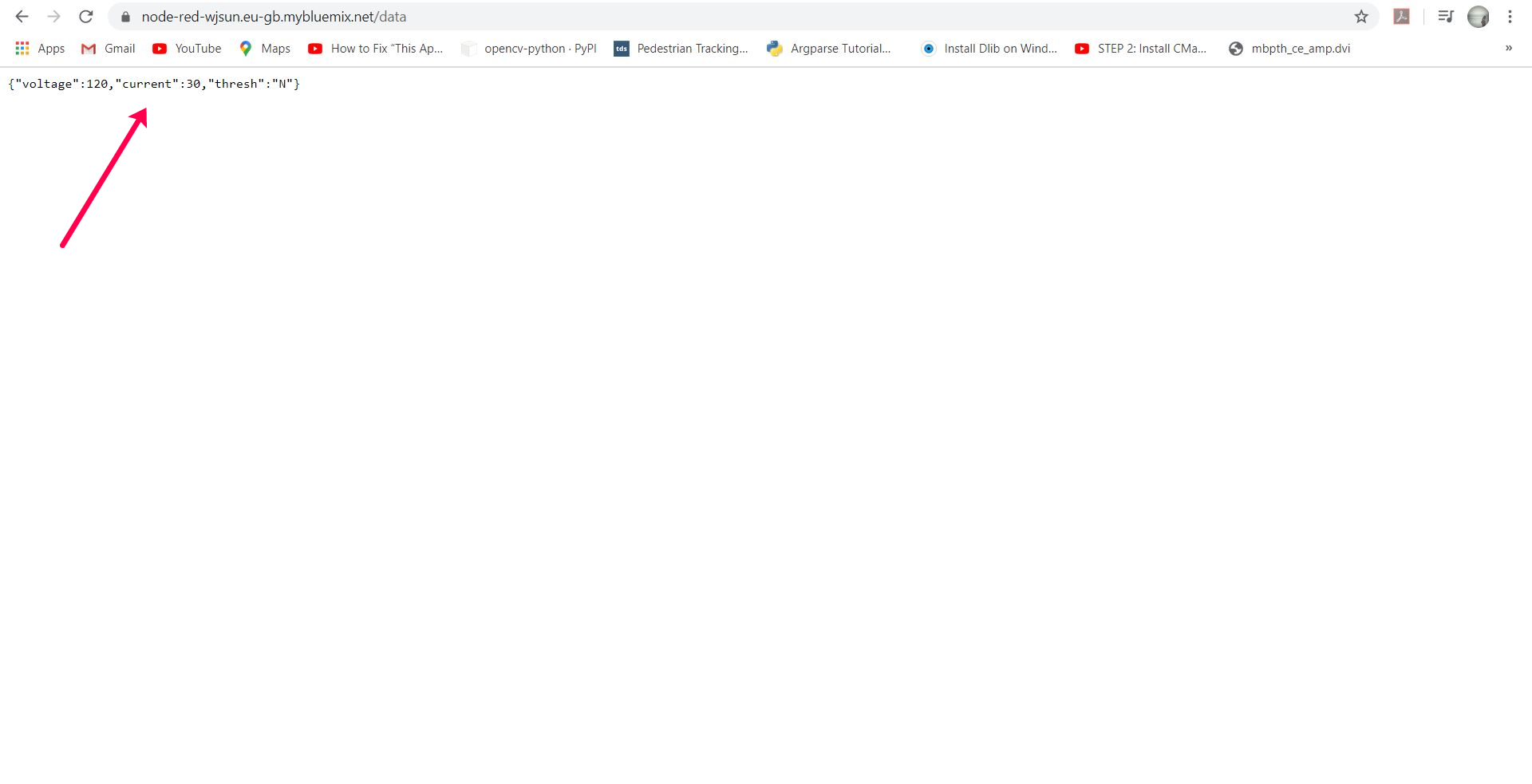
When the user presses the meteroff button, the code is made to stop reading values in this project. In real time application, a relay can be used to switch off the energy meter.

Ibm iot platform is notified when the user decides to switch off the meter by pressing the meteroff button.



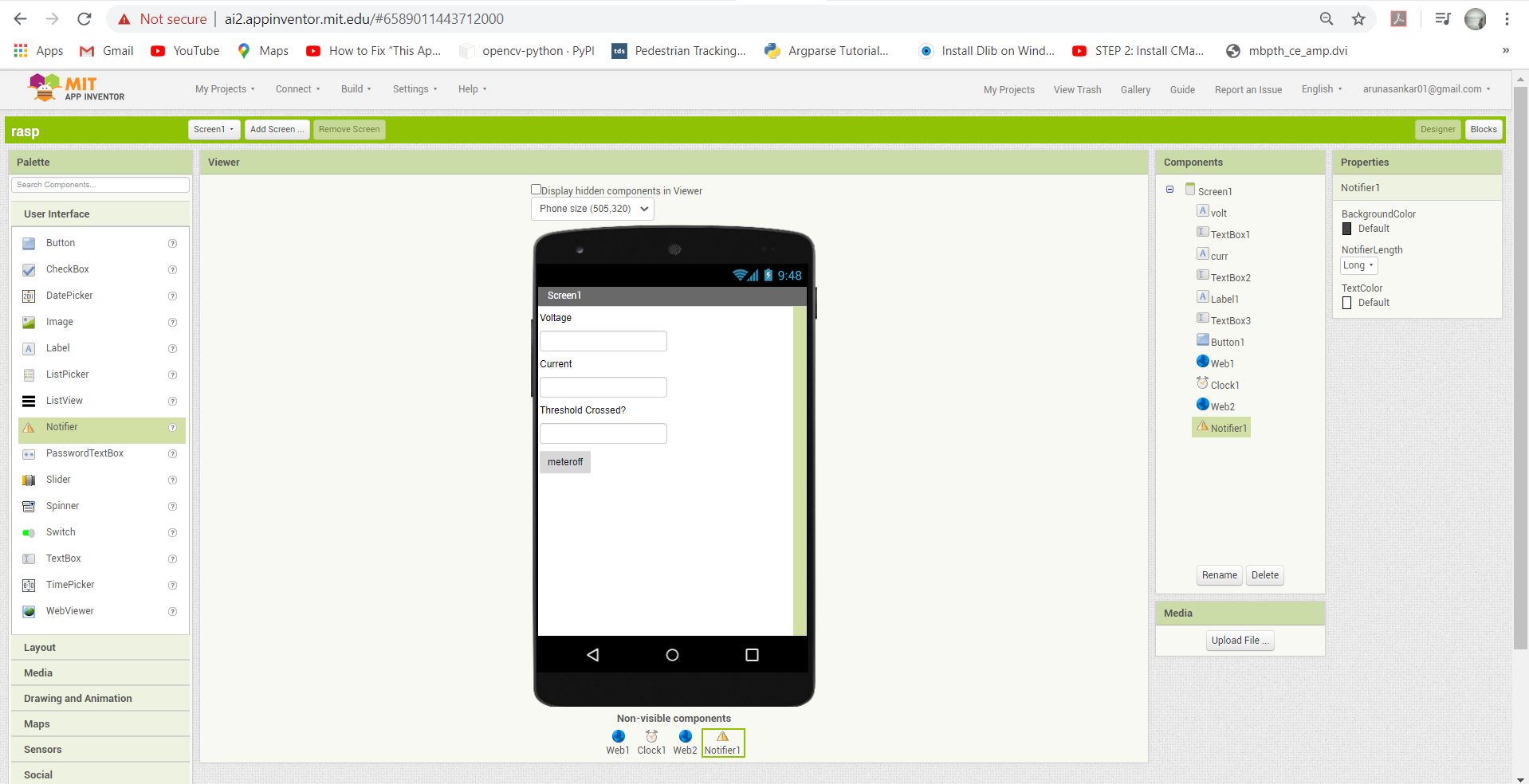


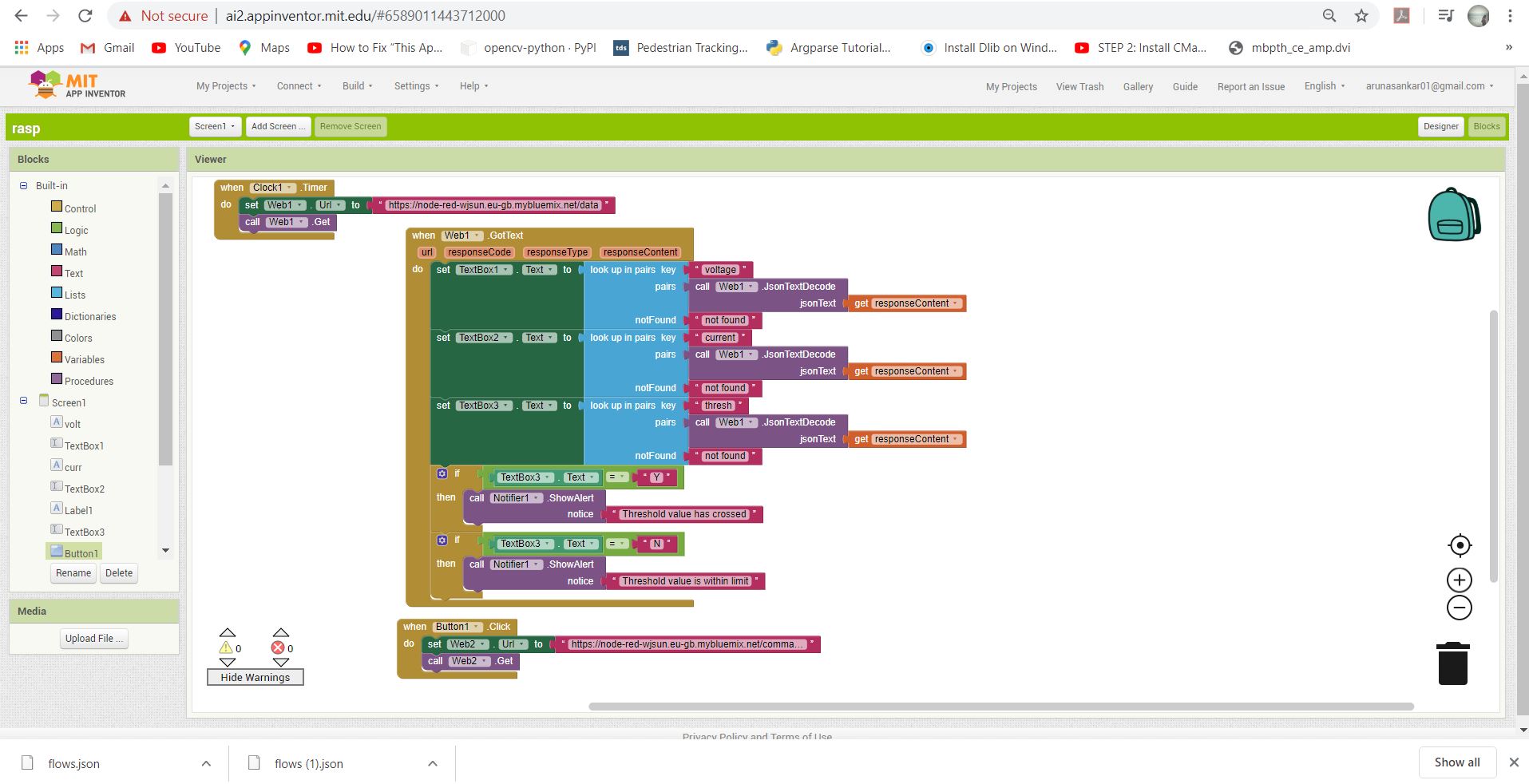
The data is displayed in url:<https://node-red-wjsun.eu-gb.mybluemix.net/>data in JSON format:



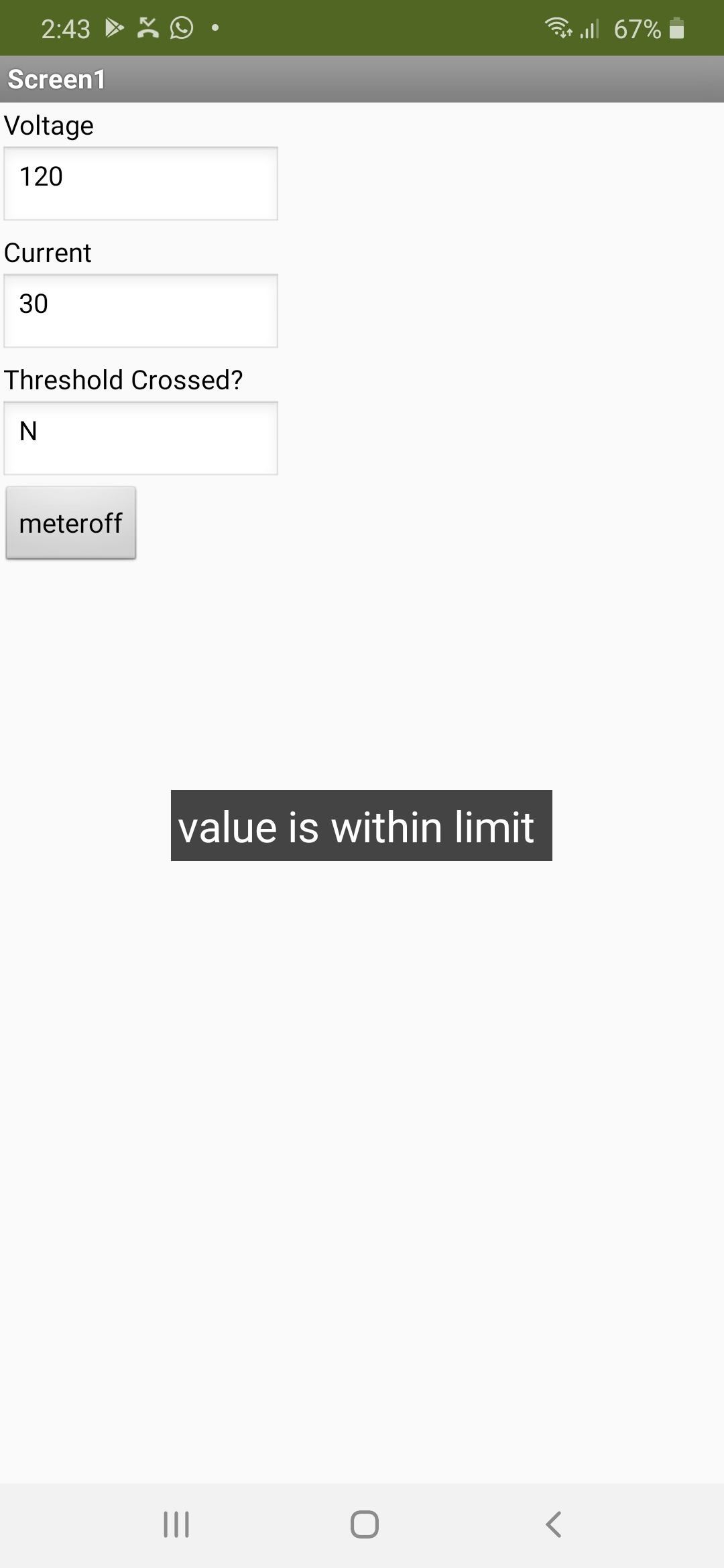
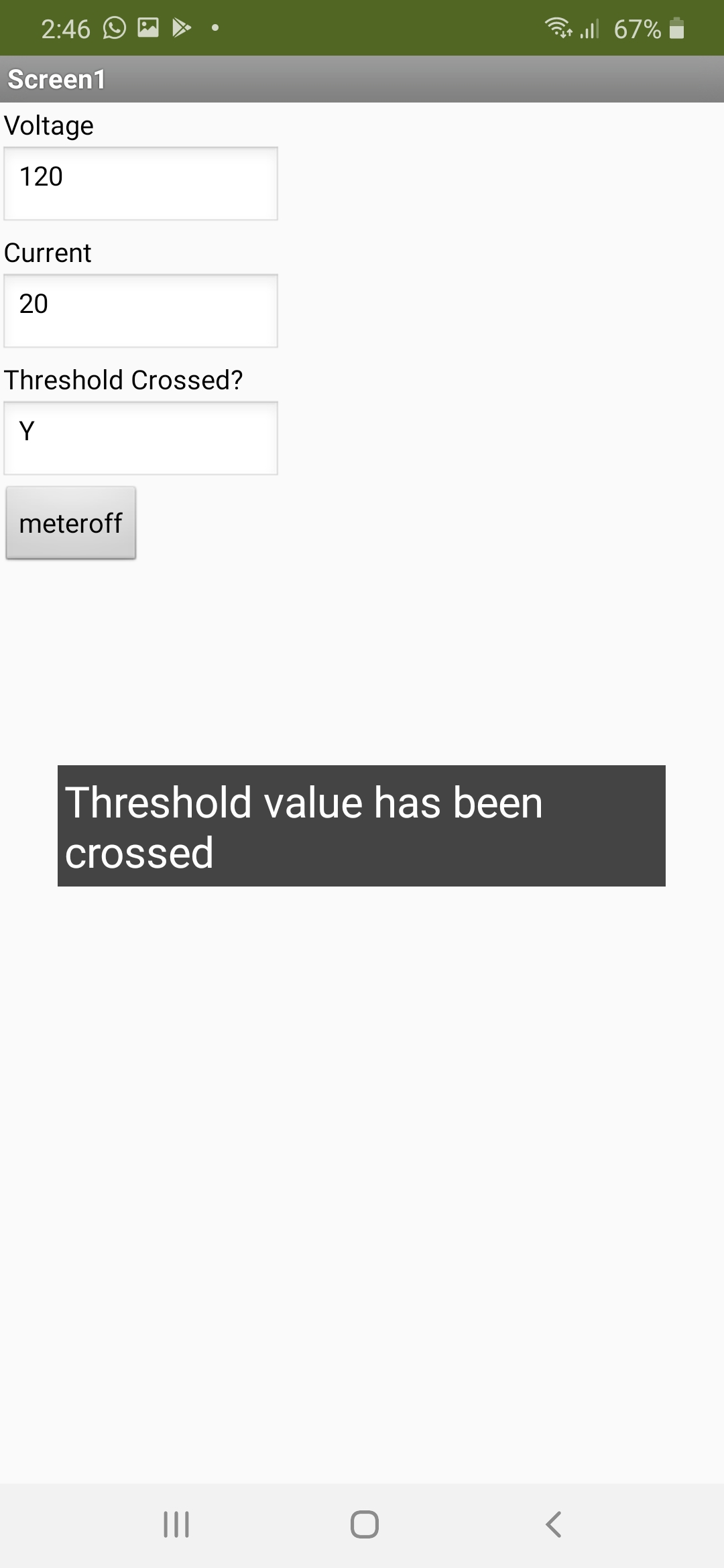
Now the connection has been established using nodered flow.

Creating mobile app and notifying when the value goes beyond the threshold.





Mobile Application:

5. RESULT:

Smart IoT based energy meter is created where the values are stored in cloud, displayed in mobile and web application. It can also be controlled from the mobile applications to ensure that the meter is switched off when the value exceeds the threshold.

6. ADVANTAGES AND DISADVANTAGES

A major advantage is getting to monitor the values from anywhere and the user does not have to go check the readings.

In this project the meter cannot be switched back on form anywhere outside the home.

7. APPLICATIONS:

The energy meter is a device found in all households. Smart energy meter finds application everywhere. **Energy meters** are widely used in domestic areas for the measurement of **electric** power consumed by the customers and these **energy meters** are commonly used in industrial sector for controlling the **electric** power of various machinery according to its reading and for measurement of **electric** power.

8. CONCLUSION:

The smart energy meter is achieved by empoying IoT, Cloud services and Mobile and web applications created to monitor the readings.

9. FUTURE SCOPE:

This can further be integrated with household devices and a complete control to all appliances in the household can be brought through cloud service, iot and the mobile application.

10. BIBILOGRAPHY

<https://www.microsemi.com/applications/industrial-m2m-wireless/smart-metering#:~:text=Smart%20meters%20enable%20two%2Dway,power%20sensor%20design%20is%20required.>

<https://www.telit.com/industries-solutions/smart-energy-utilities/smart-metering/#:~:text=Smart%20metering%20is%20one%20of,energy%20consumption%20and%20water%20usage.&text=With%20IoT%20solutions%20for%20smart,insights%20developed%20from%20meter%20data.>

<https://smartbridge.teachable.com/courses/880154/lectures/16611754>

<https://nodered.org/docs/tutorials/first-flow>

APPENDIX

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide

organization = "xpp2bo"

deviceType = "raspberrypi"

deviceId = "212129"

authMethod = "token"

authToken = "12345678"

#Initialise GPIO

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

#............................................

except Exception as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

deviceCli.connect()

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

print(type(cmd.data))

if cmd.data=="meteroff":

print("meter is off")

volt=0

current=0

data = { 'Voltage' : volt, 'Current' : current }

#print(data)

def myOnPublishCallback():

print ("Published Voltage = %s V" %volt, "Current = %s A" % current, "to IBM Watson")

success = deviceCli.publishEvent("DHT11", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

exit()

deviceCli.disconnect()

while True:

#volt=random.randint(120, 230)

volt=120

#print(hum)

#current=random.randint(30, 80)

current=20

power=int(volt/current)

power\_t=4

cross\_thresh='N'

if (power>power\_t):

cross\_thresh='Y'

#Send voltage and current to IBM Watson

data = { 'Voltage' : volt, 'Current' : current, 'Crossed\_threshold' : cross\_thresh }

#print(data)

def myOnPublishCallback():

print ("Published Voltage = %s V" %volt, "Current = %s A" % current, "to IBM Watson")

success = deviceCli.publishEvent("DHT11", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(2)

deviceCli.commandCallback = myCommandCallback

#Disconnect the device and applicatiom from the cloud

deviceCli.disconnect()