

# ***Project report***

***NAME : B.G.KEDARNATH REDDY***

***TITLE : SMARTKITCHEN USING IBM CLOUD***

***CATEGORY:INTERNET OF THINGS***

***Internship at  
smartinternz.com***

# SMART KITCHEN USIN

## SMART KITCHEN USING IBM CLOUD

### ***Aim and scope:-***

***Smart kitchen is what the future kitchens look like. We can integrate the technology with the things in kitchen to make out daytoday life hassle free***

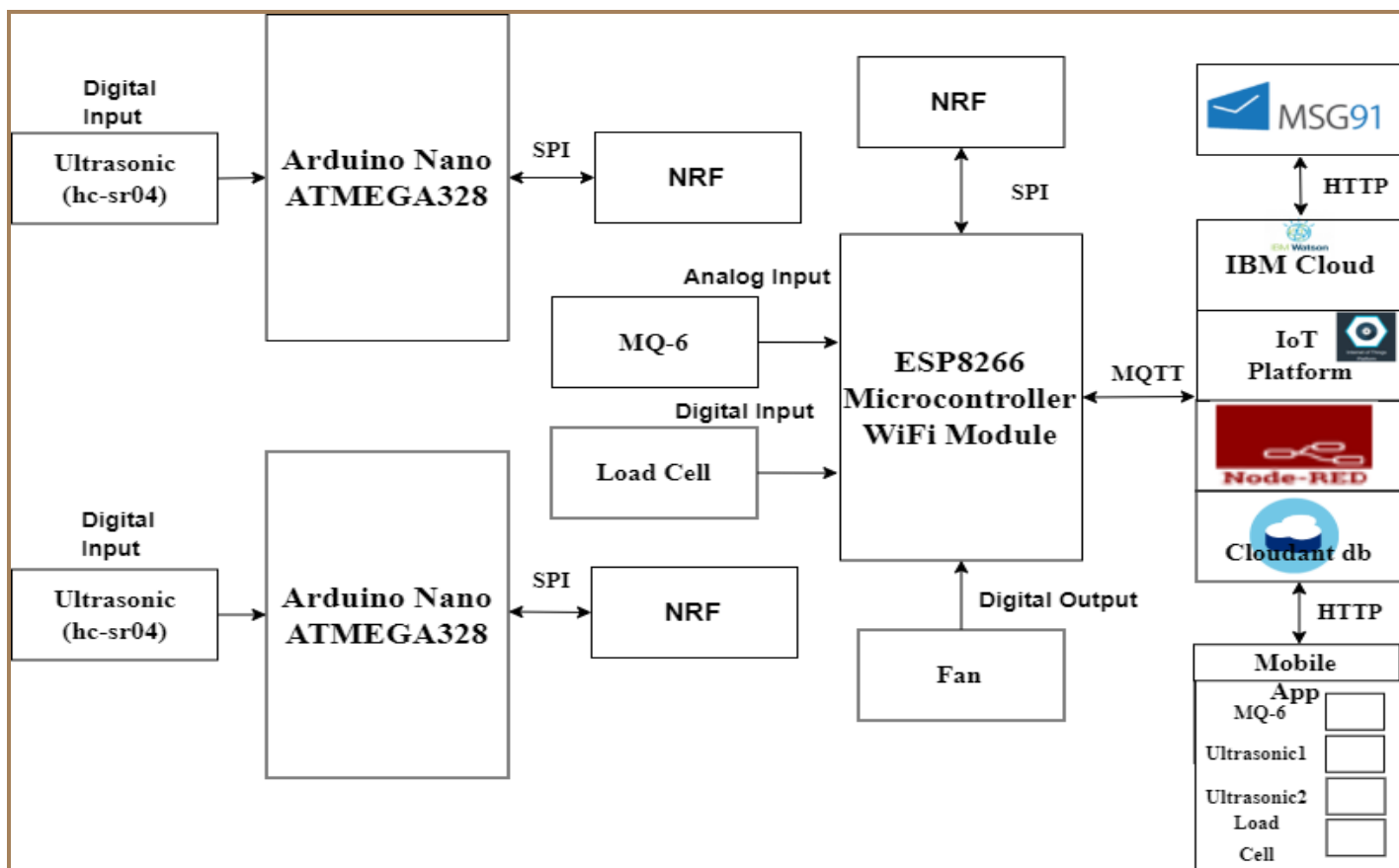
### Features:

- We can replace all the regular storage jars with the smart jars, which sends an alert when the jar gets empty or the measured sensor value is below the threshold.
- These jars communicate with the controller through Nrf communication.
- The cylinder is attached with a leakage sensor that detects the leakage from the cylinder and sends a notification if any leakage is detected.
- If any leakage is detected the exhaust fans are automatically switched ON.
- Cylinder weight is also measured and sends an alert when it is empty, based on the empty

cylinder weight.

- All these parameters can be monitored by both Mobile App and Web App.

***The project flow can be seen in the following diagram***



***Project deliverables:***

- 1. Create an ibm account, create nodered application , create a ibm watson iot platform***
- 2. Create a fast2sms account ( for sending alert messages )***
- 3. Code snippet for sending sensor data to the watsoniot platform and for sending alert messages to the user***
- 4. Create the nodered flow to get data from the device and http request to communicate with the mobile app***
- 5. Create a mobile app using MIT APP INVENTOR and configure it to get data from the cloud***

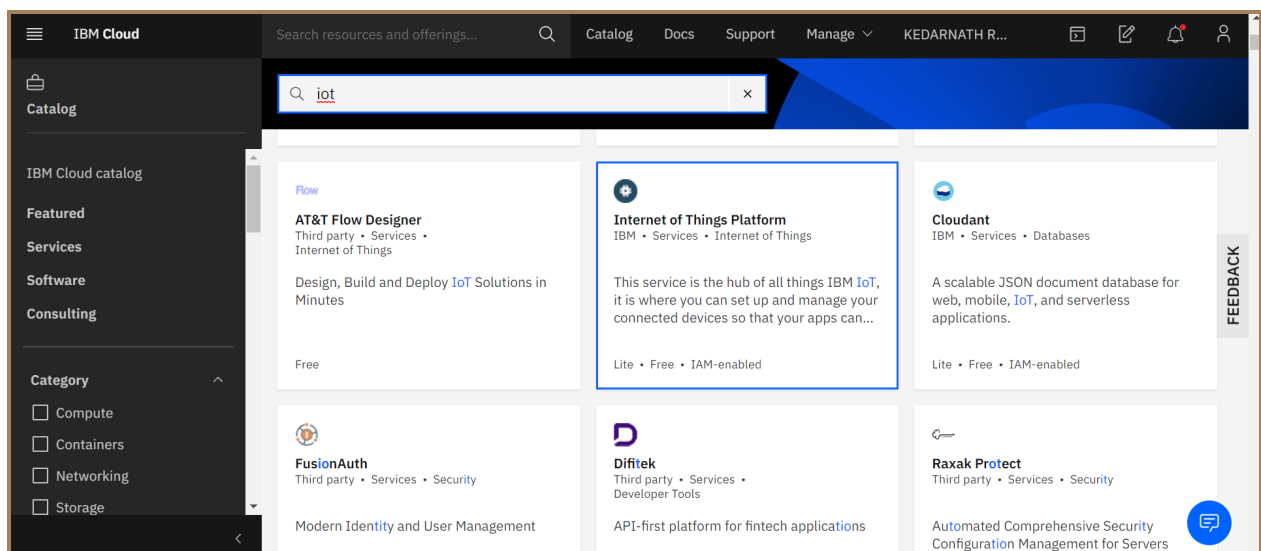
# **PROJECT TEAM:B.G.KEDARNATH REDDY**

- 1. Create an ibm account, create nodered application , create a ibm watson iot platform:-**

**We need a cloud to send sensor data so I am using IBM cloud for this task**

- 1. Open the website cloud.ibm.com in your browser**
- 2. Sign up with your details to create an account in the cloud**

**Go to catalog, search internet of things in the search bar and select INTERNET OF THINGS PLATFORM there**



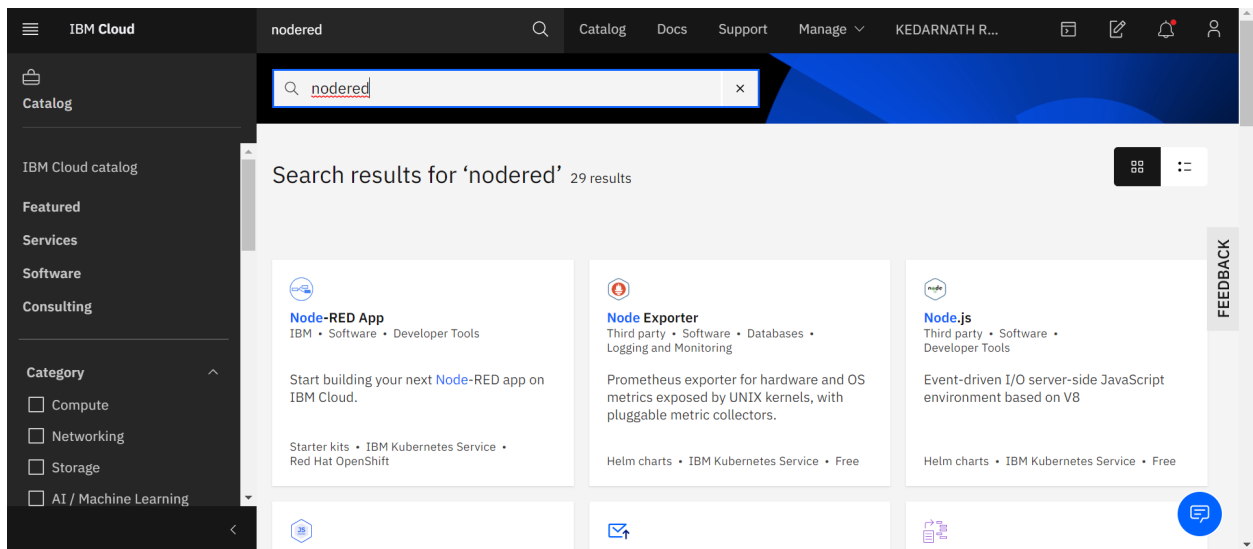
**Now create that service**

**After creation that service press on the launch button on the dashboard**

**After launching the watson iot platform add a device in it**

**After this come to the cloud main page**

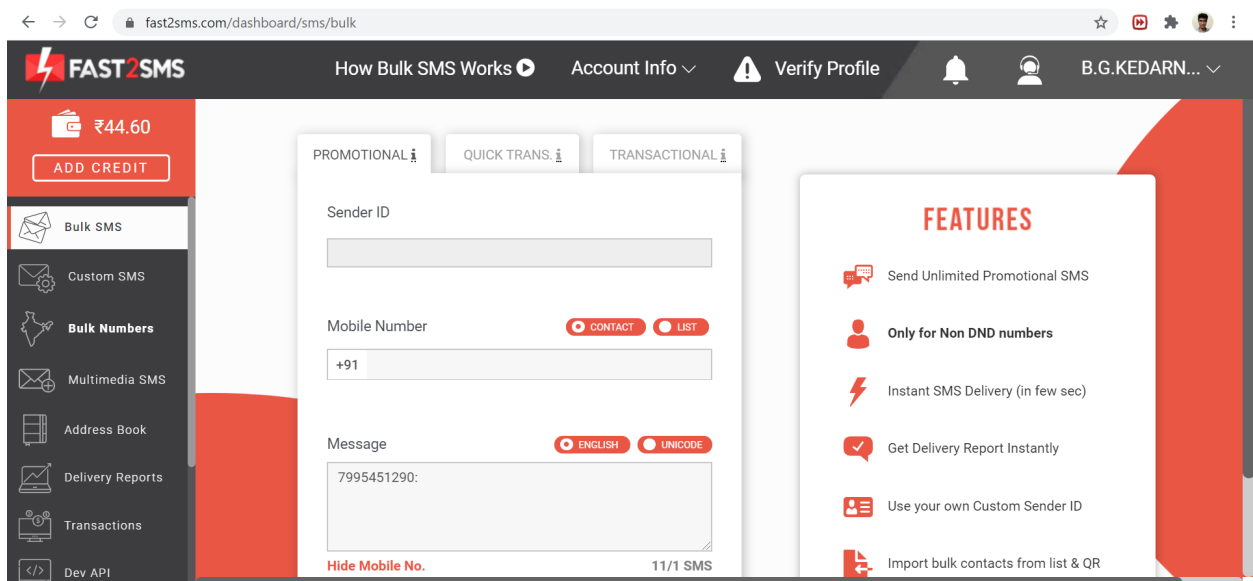
**Search nodered in the catalog, click on the nodered app and create nodered application**



## 2. Create a fast2sms account ( for sending alert messages ):-

Now create fast2sms account to send alert messages to the user

Search fast2sms in browser open that website and create an account there



## 3. Code snippet for sending sensor data to the watsoniot platform and for sending alert messages to the user

**Note:- we dont have any sensors to send the data to the cloud so we send sensor data with python code**

The following code is the code used for this task

*In the below code enter the credentials of the device that you created in the watson iot platform*

**PYTHON CODE**

**NOTE:- DONT PLACE MESSAGE CODE INSIDE THE LOOP IF YOU DO THAT THE FLOW OF MESSAGES WONT STOP**

```
jupyter device connection and alert messages Last Checkpoint: Last Thursday at 11:44 AM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [ ]: #try to use jupyter notebook while executing the program wait for atleast 40 seconds for the entire program to run

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
import requests
#Provide your IBM Watson Device Credentials
organization = "oqu8ly"
deviceType = "finalpro"
deviceId = "133"
authMethod = "token"
authToken = "87654321"

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)#Commands

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()
```

```
jupyter device connection and alert messages Last Checkpoint: Last Thursday at 11:44 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect() #try with different values
u11=5 # gives the level of item(salt) in container, 7 being threshold minimum
u12=6 # gives the level of item(sugar) in container, 7 being threshold minimum
cyl=5 # gives the weight of the cylinder 5kg being empty weight of cylinder minimum
leak="leakage" #detect the Leakage of CNG in kitchen
#enter your mobile number
if u11<7:
    r=requests.get('https://www.fast2sms.com/dev/bulk?authorization=2lhGxE6vBDIHkQ1jAyrtdqapNRUYoMOSSdn0fsmw7F3CLeT4z80t6TMq8uFH')
if u12<7:
    r=requests.get('https://www.fast2sms.com/dev/bulk?authorization=2lhGxE6vBDIHkQ1jAyrtdqapNRUYoMOSSdn0fsmw7F3CLeT4z80t6TMq8uFH')
if cyl<5:
    r=requests.get('https://www.fast2sms.com/dev/bulk?authorization=2lhGxE6vBDIHkQ1jAyrtdqapNRUYoMOSSdn0fsmw7F3CLeT4z80t6TMq8uFH')
if leak=="leakage":
    r=requests.get('https://www.fast2sms.com/dev/bulk?authorization=2lhGxE6vBDIHkQ1jAyrtdqapNRUYoMOSSdn0fsmw7F3CLeT4z80t6TMq8uFH')
while True:
    data = { 'ultrasonic1' : u11, 'ultrasonic2': u12 , 'cylwt':cyl , 'mq6':leak}
    #print (data)
    def myOnPublishCallback():
        print ("Published ultrasonic1 = %s " % u11, "ultrasonic2 = %s " % u12,"cylwt = %s " % cyl,"mq6 = %s" % leak,"to IBM I
    success = deviceCli.publishEvent("kitchen", "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 application from the cloud
deviceCli.disconnect()
```

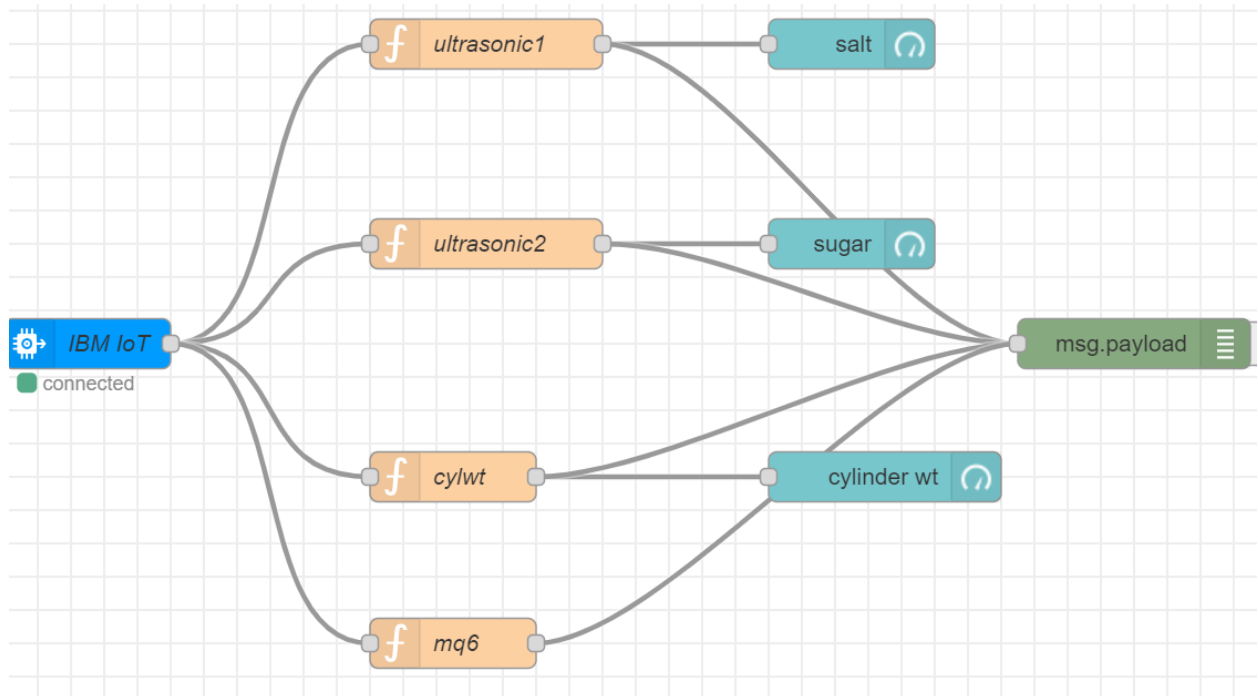
**4. Create the nodered flow to get data from the device and http request to communicate**

## ***with the mobile app***

***We need to create two flows to do this task***

### ***Flow1:-***

***To get data from the device***



***To connect ibmiot device node to the device double click on the node and enter the device credentials of the device that you have in your watson iot platform***

***like this***

Edit ibmiot in node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🔗

☔ Authentication

API Key

▼

🔍 API Key

713e72d6.e8053c

▼

✎

⚙️ Input Type

Device Event

▼

📡 Device Type

☐ All or

finalpro

👤 Device Id

☐ All or

133

☰ Event

☒ All or

+

📄 Format

☐ All or

json

○ Enabled

***Now the data that comes from the device is combined you need to parse the data and display data individually***

***Code the function node like this***



Edit function node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖼

📁 Name

ultrasonic1

📄 ▼

Setup

Function

Close

↗

1 global.set('Ultrasonic1',msg.payload.ultrasonic1)

i 2 msg.payload=msg.payload.ultrasonic1

3 return msg;

🔗 Outputs

1

▲▼

☐ Enabled

## ***Codes of all 4 function nodes***

***1.***

***global.set('Ultrasonic1',msg.payload.ultrasonic1)***

***msg.payload=msg.payload.ultrasonic1***

***return msg;***

***2.***

***global.set('Ultrasonic2',msg.payload.ultrasonic2)***

***msg.payload=msg.payload.ultrasonic2***

***return msg;***

3.

```
global.set('Cylwt',msg.payload.cylwt)
```

```
msg.payload=msg.payload.cylwt
```

```
return msg;
```

4.

```
global.set('Mq6',msg.payload.mq6)
```

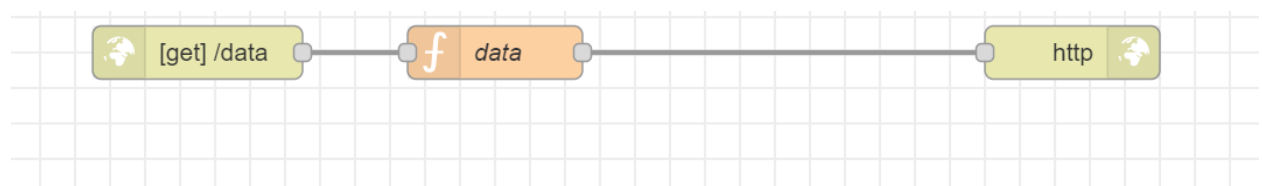
```
msg.payload=msg.payload.mq6
```

```
return msg;
```

***connect those function nodes to gauges to display information on the dashboard***

## ***Flow2:-***

***To create http request to communicate with mobile app***



Edit http in node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🔗

☰ Method

GET

▼

🌐 URL

/data

🏷️ Name

Name

☐ Enabled

*configure httpin node like this*

Edit function node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖼

🔑 Name

data

📄 ▼

Setup

Function

Close

1 msg.payload={ 'ultrasonic1':global.get("Ultrasonic1"), 'ultrasonic2':globa

2 return msg;

↻ Outputs

1

▲▼

☐ Enabled

*function node like this*

**code for the function node**

```
msg.payload={ 'ultrasonic1':global.get("Ultrasonic1"), 'ultrasonic2':g  
lobal.get("Ultrasonic2"), 'cylwt':global.get("Cylwt"), 'mq6':global.get("Mq6") }
```

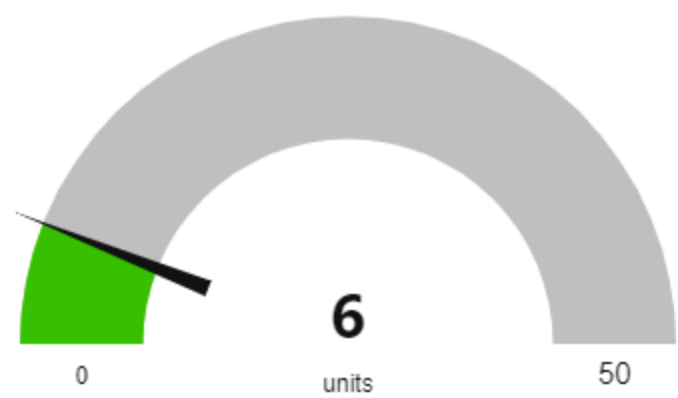
```
return msg;
```

***By this flow we are sending data to the server***

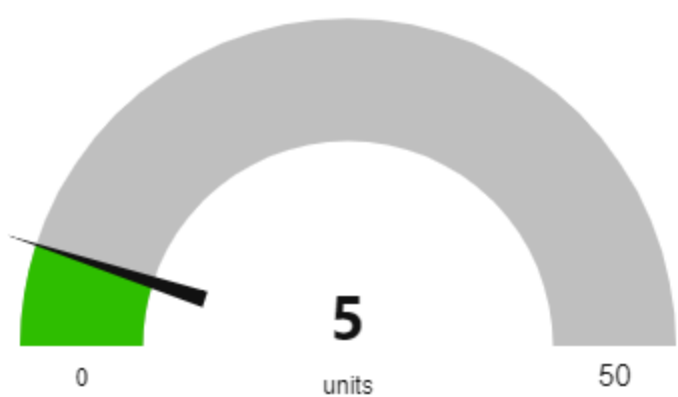
***The data that has been sent to the server will be like this***

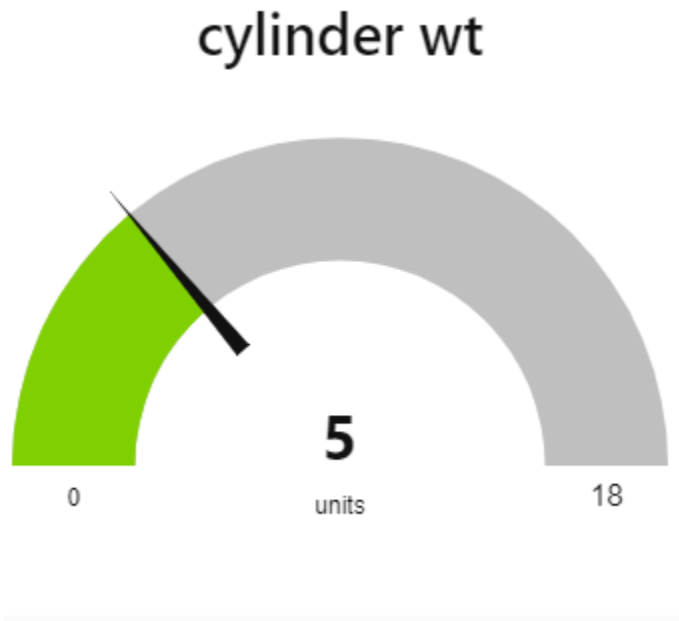
```
{"ultrasonic1":5,"ultrasonic2":6,"cylwt":5,"mq6":"leakage"}
```

*the web app ui will be like this*



salt





***note:- cylwt is the weight of cylinder with 5 being empty weight***

***ultrasonic1 gives level of sugar and ultrasonic2 gives level of salt  
in their respective jars***

***ultrasonic1 and ultrasonic2 are names sensors kept in jars***

## ***5. Create a mobile app using MIT APP INVENTOR and configure it to get data from the cloud***

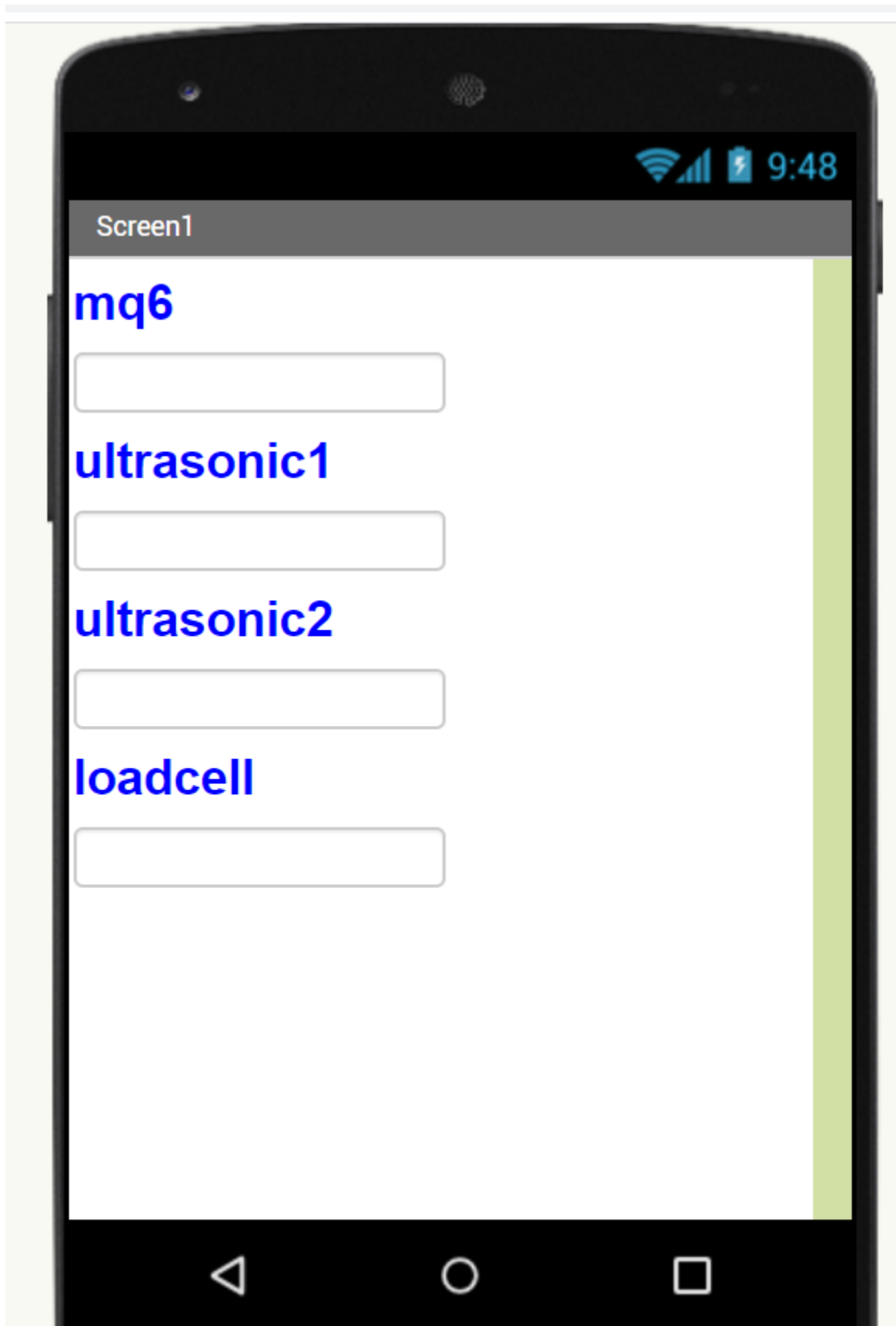
***Open your browser and search 'mit app inventor' and open the website***

***Click on 'create apps' on the dashboard and login with your google account***

***Give the name of your project you should not give spaces in your project name***

***configure the ui of your app like this it should have 4 labels and***

*their respective text boxes*



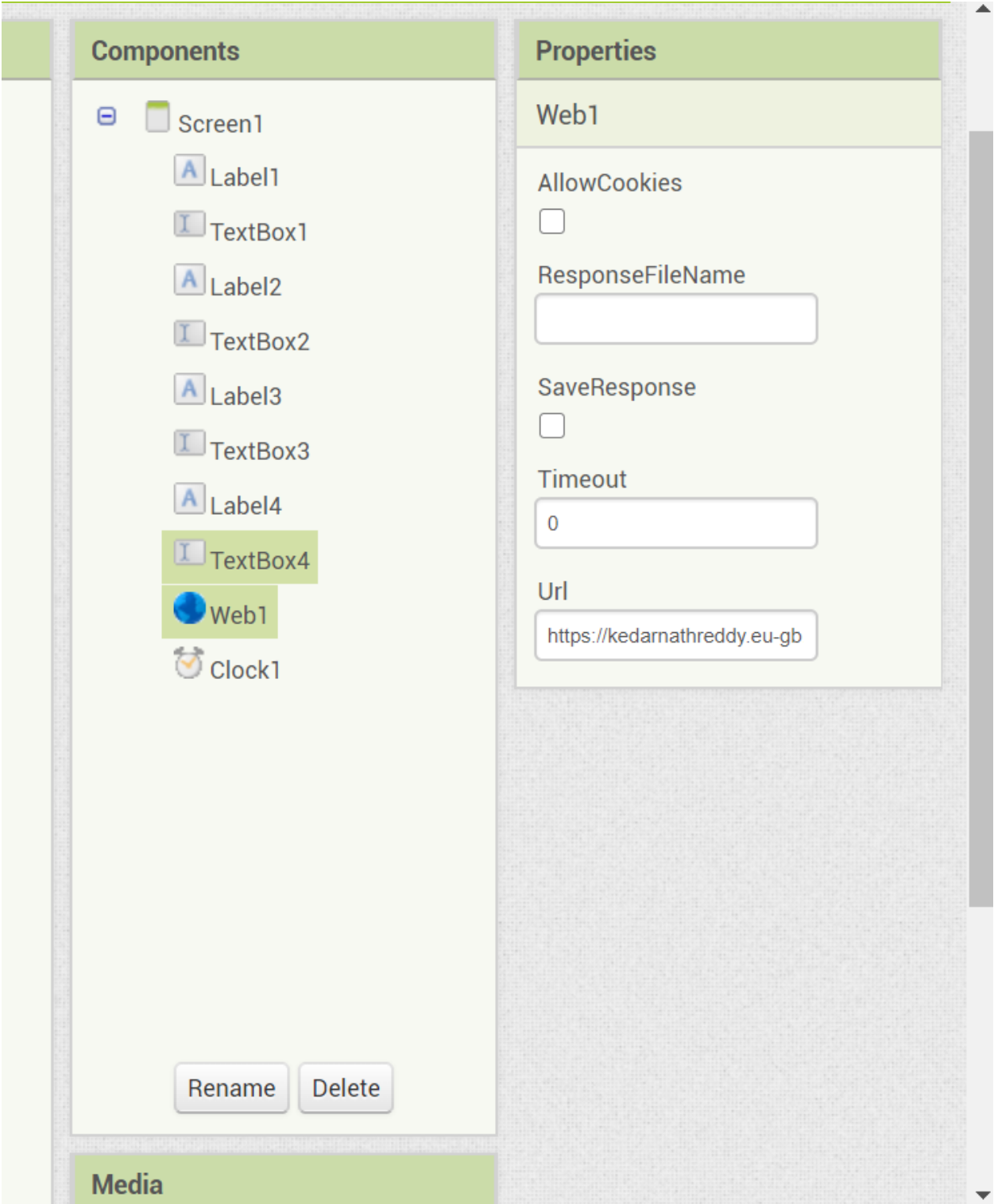
*mq6 tells if there is any cng leakage in the kitchen*

*ultrasonic1 and ultrasonic2 gives the level of salt and sugar in the*

*respective jars*

*loadcell gives the weight of the cylinder*

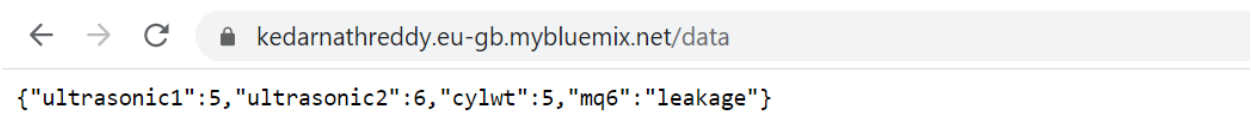
*Now drop the web from connectivity on to the board and enter the url in the web*



*Note:-the app receives data from the url that you enter in web so you should enter the url that receives data from ibm device*

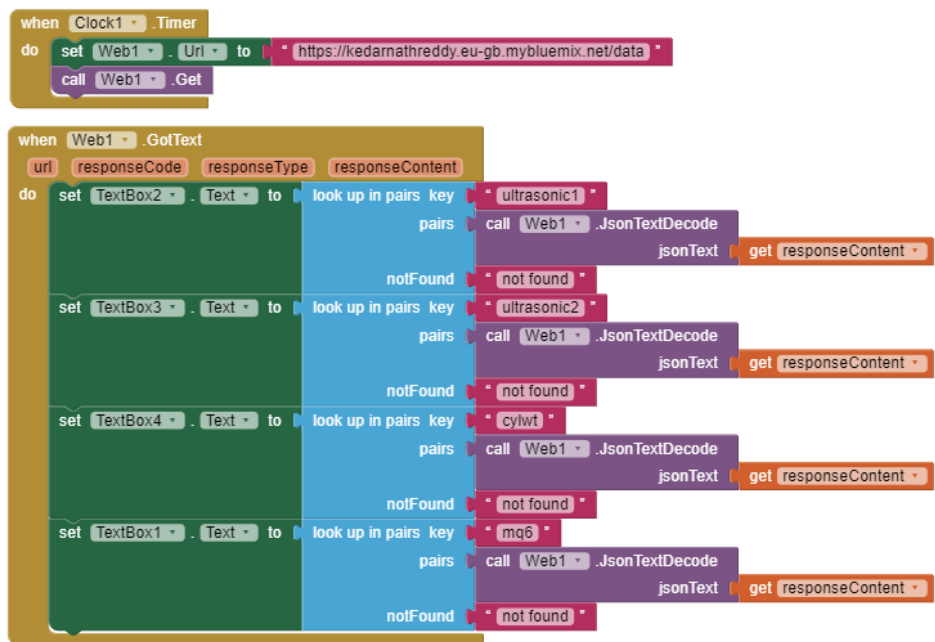


**Enter this url**



**Now click on the blocks on top right corner of screen and start arrenging the blocks to create backend of the app**

**Set the blocks in this manner for the text boxes**



**These blocks are there to decode data that is in the form of json and display then in their respective text boxes**

**Now everything is done click on build option on the top of dashboard and download the apk file, install it in your mobile**

**the app opened on mobile will be like this**

## Screen1

### mq6

leakage

### ultrasonic1

5

### ultrasonic2

6

### loadcell

5

***MQ6 showing there is a leakage***

***ultrasonic1 showing level of item in that jar***

***ultrasonic2 showing level of another jar***

***load cell showing weight of cylinder***

***This is end of the report***

***THE END***