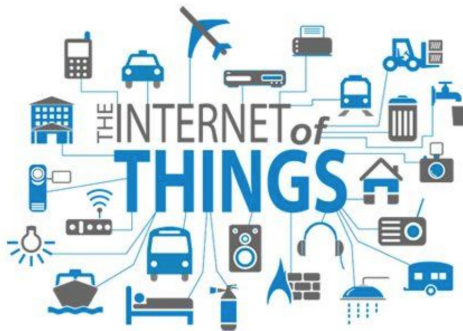




SMART HOUSE

IoT Lab



Department of Computer Science
Kristianstad University
Course: DT582C
Master's in computer science

Teacher: Qinghua

Dated: 2019-12-31

Table of contents

[Table of contents](#)

[Introduction](#)

[Installation](#)

[Implementation](#)

[Node JS](#)

[Babel](#)

[Express and related packages](#)

[Thingy 52](#)

[HS100](#)

[Icecast](#)

[Lame](#)

[Socket.io](#)

[Mongo Db](#)

[Required bluetooth packages](#)

[OpenCv4NodeJS](#)

[Firebase configuration](#)

[Noble device](#)

[Task 01](#)

[Task 02](#)

[Task 03](#)

[Task 04](#)

[Task 05](#)

[Task 06](#)

[Task 07](#)

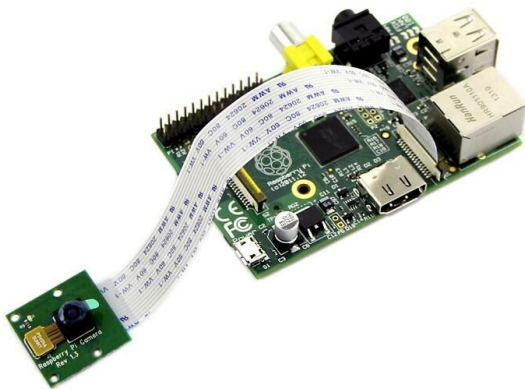
[Development environment and scenario](#)

[Problems and solutions](#)

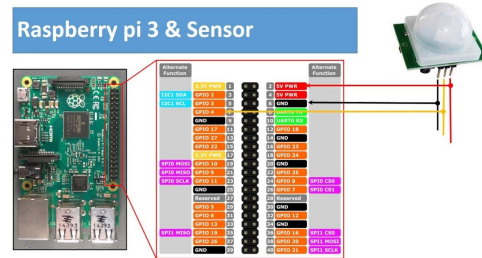
[Tips & Tricks](#)

[References](#)

a) Raspberry PI 3



b) PIR sensor



c) Netio 4



d) HS100



e) Nordic Thingy52



f) Wifi USB Dongle



Fig 1 : IoT devices used in this lab

Introduction

The task is to build a smart house demonstration of the Internet of Things using a set of sensors (Nordic Thingy 52, HS100, Netio 4). We fetch data from the sensor devices to show in our application. We also build an interactive interface to command the sensor devices.

We used Raspberry PI 3 with Raspbian OS installed. In this lab we worked specifically with 5 tasks (but i optionally completed the rest of the 2 also) as mentioned below. Each task has its own README[1] file in the source code repo.

The source code of this project is available on GitHub (link in reference section), and a demo link is given on github pages.

Installation

1. `sudo su`
2. Clone the repo
3. `npm install --unsafe-perm` # may take a long time (in hours)
4. `npm run task1` # task2 , task2, ... task7

Implementation

This is a crucial step to install the Raspbian OS on the Raspberry PI. There are several libraries and packages for Node to be installed. We have some issues in installation which sorted out during the installation and we have provided the solution to these issues in the section Problems and Solutions later in this article. We also provide some tips and tricks in working with raspberry pi.

We installed the following libraries and packages for this application. Please note that all these may not be required for the first few tasks but we prepared the system at one place for ease of reading and re-implementation of the tasks. We used Ubuntu 18.04 for flashing and some basic commands with SD card etc and therefore all commands are given for Ubuntu. But these could easily be followed (replaced) for other Operating systems like Windows and Mac.

Please note that to make the installations easy we recommend to switch to root. If some images are not available in this document (or want latest version) please refer to [github\[2\]](#) repo given in references section.

1. Node JS
2. Babel transpilers for ES 6 support
3. Express and related packages
4. Thingy 52
5. HS 100
6. Icecast
7. Lame
8. Socket.io
9. Mongo db
10. Install required packages i.e. bluetooth bluez libbluetooth-dev libbluetooth-dev libudev-dev
11. Opencv4nodejs
12. Firebase configuration
13. noble-device

1. Node JS

To install node js, the easiest way is to install through NVM (node version manage).

- Install NVM

```
curl -o-  
https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh |  
bash
```

There will be an export statement at the end, just copy that whole statement and paste it in the terminal

- To make the commands available to current terminal run:

```
source ~/.bashrc
```

- Install node v8

```
nvm install 8
```

- Set it as default

```
nvm use 8
```

2. Babel

```
npm install @babel/core @babel/node @babel/preset-env
```

Create a file named .babelrc with the following contents.

```
{  
  "presets": [  
    "@babel/preset-env"  
  ]  
}
```

3. Express and related packages

npm i -S express body-parser cors

4. Thingy 52

npm i -S thingy52

5. HS100

npm i -S hs100-api

6. Icecast

npm i -S icecast

7. Lame

npm i -S lame

8. Socket.io

npm i -S socket.io

9. Mongo Db

npm i -S mongodb

10. Required bluetooth packages

npm i -S bluetooth bluez libbluetooth-dev libbluetooth-dev libudev-dev

11. OpenCv4NodeJS

npm i -S opencv4nodejs --unsafe-perm

This may require a lot of compilation time and you may skip this package.

12. Firebase configuration

- Create a google account (google.com)
- Goto firebase console (<https://console.firebase.google.com>)
- Add a project
- Goto Project Overview and click Add app, select web (<>) and then follow the instructions.
- From the above step you will get web app credentials.
- To integrate with node js you need to install firebase-admin package
- Goto Project settings (gear icon) > Service accounts and then click Generate new key, download it.
- Now goto Real-time database and click ellipses (...) and then 'create new database'

13. Noble device

npm i -S noble-device --unsafe-perm

Task 01

Turn on and off a smart plug (HS100) wirelessly. The button on the Nordic Thingy 52 will be used to signal the HS100 turn on/off.

Implementation

To implement this task we built a small script in Node Js to read the state data when the button is pressed on the Nordic Thingy. Then we used the http API of HS100 to send on/off command to HS100.

We have used the following two functions and a global variable 'enabled' to hold the state value of button.

```
Var enabled = false;  
  
function switchOnOfHs100(status) { .... }  
  
function onButtonChange(state) { .... }
```

When the button on thingy is pressed the function onButtonChange is called which turn the HS 100 ON or OFF as per the value of 'enabled'. The HS100 device provides a very easy HTTP API for Node JS [2], you need to find the IP address (ref: Tips & tricks below) of the device and use it in the code.

To run the task script use the following commands:

```
cd iot-lab  
  
npm run task1
```

Output:

```
TASK 1 - started  
Discovered thingy  
Plug switched ON
```

Task 02

In this task we use IFTTT to send a sensor event to a remote device (Netio4). IFTTT (If This Then That) is a free web service which triggers different services based on events. For Example if a user tweets then send an email.

Implementation

This task can be divided into two parts. In the first part we create a web url with a key which is used to trigger the event that we will create in the second part. Note that part 1 is required only once and it will work for any event that is created in the second part.

Part 1:

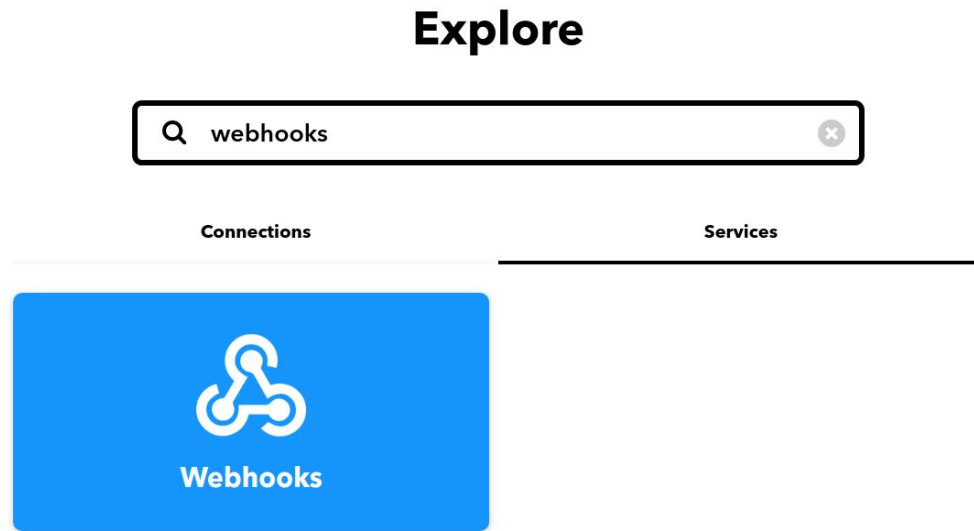
1. You need to register on <https://ifttt.com/>
2. You need to connect to Webhooks service, so type 'webhooks' in the search box at the top left corner.

IFTTT

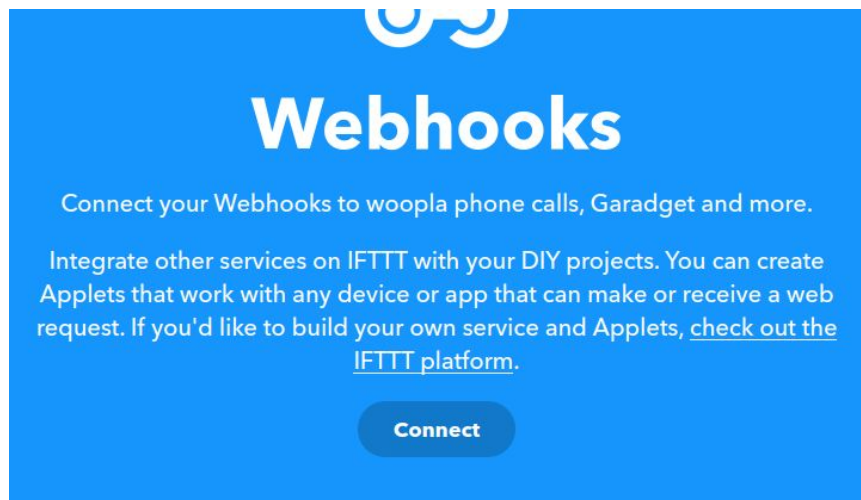
Home

Q Search

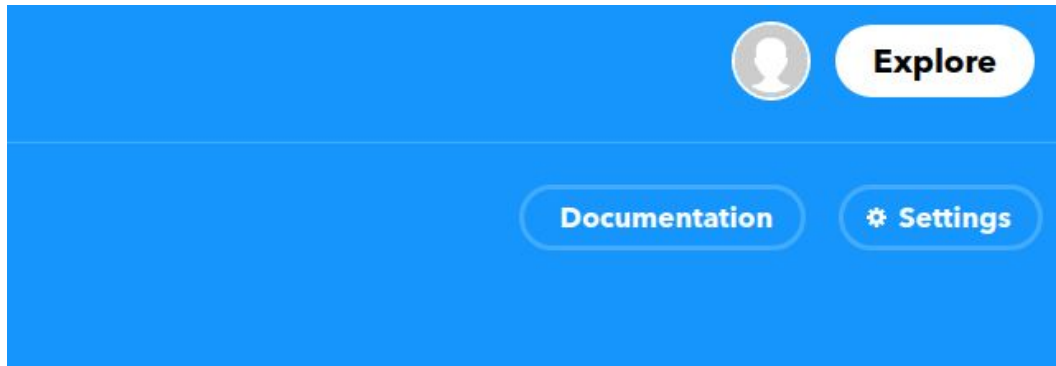
-
- Click the services tab and then click webhooks.




- Now connect this service to your account by clicking connect.



- Once connected in the above step, you would be able to see a button 'Documentation' at the top right corner.



6. After clicking the Documentation button above you come to the following page, which contains both the link along with the key. This link will be used to fire an event, so copy the url or bookmark the url in your browser.



Your key is:
esXhMacZzkk96SpwHaXE5LRLor2UcqXyV4o

[◀ Back to service](#)

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/{event}/with/key/esXhMacZzkk96SpwHaXE5LRLor2UcqXyV4oQBZC
```

With an optional JSON body of:

```
{ "value1" : " ", "value2" : " ", "value3" : " " }
```

The data is completely optional, and you can also pass value1, value2, and value3 as query param passed on to the Action in your Recipe.

You can also try it with `curl` from a command line.

```
curl -X POST https://maker.ifttt.com/trigger/{event}/with/key/esXhMacZzkk96SpwHaXE5LRLor2U
```

[Test It](#)

7. At this point you may click 'Test it' button in the above figure to see an alert about event.

Event has been triggered.

Your key is:

esXhMacZzkk96SpwHaXE5LRLor2UcqXyV4oQBZQ_

◀ Back to service

To trigger an Event

Make a POST or GET web request to:

Part 2:

In this part we will create IF THIS then THAT mean if this EVENT then that Action.

1. Navigate to User Avatar and click Create. (or simply browse to <https://ifttt.com/create>)
2. Click + in the page as shown below

Create your own

If
+ This
Then That

8. In the box 'Search services' type 'webhooks' and click it in the list.
9. Click 'Receive a web request' and type a name (netio4-on) for your event in the box.
10. Now click again + symbol in front of 'That' as shown below, to specify an action.



11. Type webhooks (once again) in the search box and choose webhooks.
12. Click 'Make a web request'
13. Here you need to type exactly (a globally accessible) url for your node server endpoint.
Please note that IFTTT can make request to public ips (domains) only. For this task I used Amazon web services, where I hosted my node app. Since AWS has public IP, therefore it can receive web request from IFTTT. The IP address and port forwarding was not working in the IoT Lab, so I used this work around successfully. You can use any public IP for example your home router IP address. Just connect your computer directly with network cable coming from the wall socket. In most cases you will get a public IP, but if not then you can use port forwarding technique on your home router. I tried several alternatives like University hotspot and home wifi but for some reasons port forwarding is blocked on the ISPs routers. So the only viable solution is VPS, Cloud service.
14. Choose GET in method, application/json for Content Type and click 'Create Action'.

Now whenever you fire your event to IFTTT

(<https://maker.ifttt.com/trigger/netio/with/key/dAYI-truncated>); the IFTTT service will make a get request to your node endpoint, which in turn switches ON/OFF netio4. In the url above replace 'netio' and hash after 'key' as per your settings.

```
> node index.js  
  
Your app is listening on port 5555  
Received IFTT request
```

To run the code for this task.

```
cd iot-lab
```

```
npm run task2
```

Task 03

In this task we are supposed to develop a Node JS server on Raspberry PI (3), fetch the sensor data from Nordic Thingy 52 and then display it on a web page.

Implementation

We developed a web server using node and express. We wrote a class MyThingy which reads all sensor data from thingy. We then make ajax calls from our web page to show that data periodically on the web page.

This lab has two parts, one for node server which reads data from thingy and saves to mongodb. In the second part we create a simple html page with some JS to show the data from mongodb (via node express) on the web page. It is recommended to install mongodb on your computer than PI as it has very limited resources.

1. To run the node server simply use :

`cd iot-lab`

`npm run task3`

The screenshot shows a code editor on the left and a web browser on the right. The code editor displays a JavaScript file named `show_sensor_details.js` with the following content:

```
3 function readData(cb) {
4   let sensor_data;
5   fetch('http://localhost:5555/api/v1/data')
6     .then(onfulfilled: resp => resp.json())
7     .then(onfulfilled: d => cb(d.slice(0,8)));
8 }
9
10 readData(cb => {
11   const table = document.getElementById('data-table')
12   d.forEach(row => {
13     const tr = document.createElement('tr');
14     row.heading = row.heading.toFixed(3);
15     let td = null;
16     Object.keys(row).map((callback: key => {
17       if(['temperature', 'pressure', 'humidity', 'eco2',
18         'td = document.createElement('td');
19         if(key === 'color' && row[key]) {
20           const color = rgbToHex(row[key].red % 255,
21             row[key].green % 255, row[key].blue % 255);
22           td.innerHTML = `<div style="background-color: ${color}"`
23         } else {
24           td.textContent = row[key];
25         }
26       });
27     });
28     tr.appendChild(td);
29   });
30   table.appendChild(tr);
31   readData();
32 })
```

The web browser on the right shows the title "Sensor data - Task 3" and the text "Reading data sensor data from Nordic Thingy52, saving in database using express server". Below this is a table with the following data:

temperature	pressure	humidity	eco2	tvoc
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0
23.24	1001.93	43	0	0

2. Then browse to <http://localhost:8080>
3. You may need to change the IP in server.js to point to your mongodb server.



Sensor data - Task 3

Reading data sensor data from Nordic Thingy52, saving in db and showing using express server

temperature	pressure	humidity	eco2	tvoc	color	heading
23.24	1001.93	43	0	0	#ddcdd1	359.928
23.24	1001.93	43	0	0	#ddcdd1	359.985
23.24	1001.93	43	0	0	#ddcdd1	0.043
23.24	1001.93	43	0	0	#ddcdd1	0.099
23.24	1001.93	43	0	0	#ddcdd1	0.154
23.24	1001.93	43	0	0	#ddcdd1	0.209
23.24	1001.93	43	0	0	#ddcdd1	0.262
23.24	1001.93	43	0	0	#ddcdd1	0.313

We can use either local node server IP address in public/index.html page or cloud functions for fetching data from real time database.

1. npm i -g firebase-tools
2. firebase login
3. firebase init functions
4. firebase deploy --only functions --debug

For hosting follow the following steps.

1. firebase login
2. firebase init hosting

3. Copy all files from public directory to firebase/public directory.
4. firebase deploy --only hosting --debug
5. Browser to <https://hkr-iot-lab1.firebaseio.com/>

Task 04

In this task we extend the task 3. We need to add a few buttons to command and control sensor devices (HS100, Thingy52).

Implementation

To do this we simply add another endpoint into our express server. This endpoint has service (device) name and status (ON/OFF, 1/0) which switches ON/OFF the devices accordingly.

When a user clicks a button on the page an ajax request is made to the node server which contains both service name and status (ON/OFF). The node server receives this request and then decides which service to switch ON/OFF.

ID	color	co2	heading	humidity	pressure	temperature	tvoc
1		517	191	28	555	21	18
2	#e89225	347	303	23	983	23	10
3	#3af133	442	19	32	1178	17	17
4	#bf87a0	925	75	30	1497	27	19
5	#457d06	758	41	36	912	19	18

To run the code for this task.

```
cd iot-lab
```

```
npm run task4
```

Then open a web browser and open url <http://localhost:8080>. You will see a web page with related buttons. Click and see the effect.

For HS 100 to work the network (Wifi) of PI should be IoT_Lab, and thingy should be discoverable.

Task 05

In this task we save the sensor data to Firebase and then display it on the web page in real time using Firebase library.

Implementation

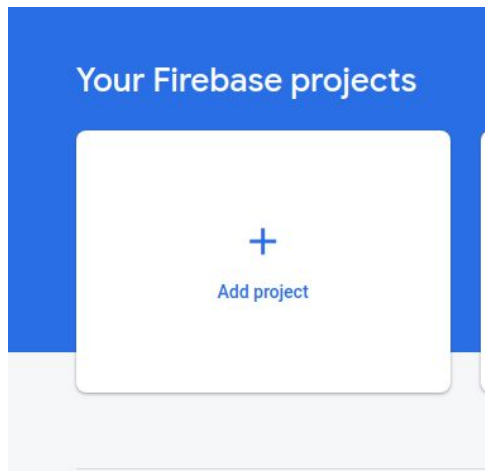
The implementation of this task includes creating a firebase account (instructions in installation section), creating a real time database, and creating a private key. You also need to get web integration details from firebase.

You can follow the following steps to create a firebase project, get the credentials, create real time database. We can divide these steps into 4 parts.

In the first part we create a project on firebase. In the second part create a web app to integrate firebase with our web page. In the third part we generate a private key for integration our node server. In the fourth part we create a real time database.

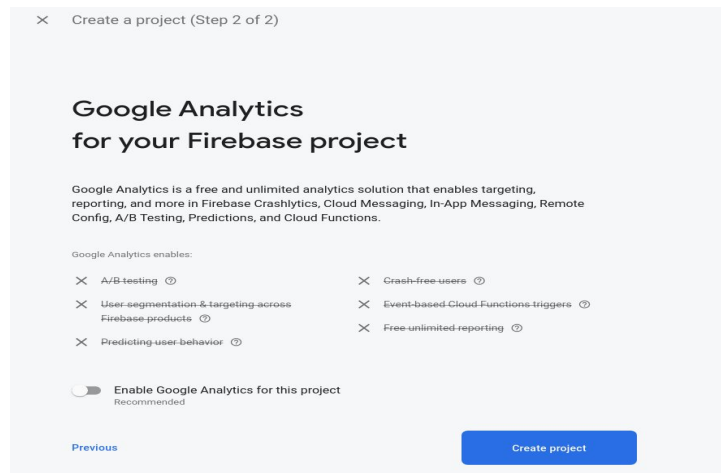
PART 1:

1. Browse to firebase console <https://console.firebase.google.com> and create a project.



2. Enter a name for your project and click continue. Your project is like a container for all your Apps.

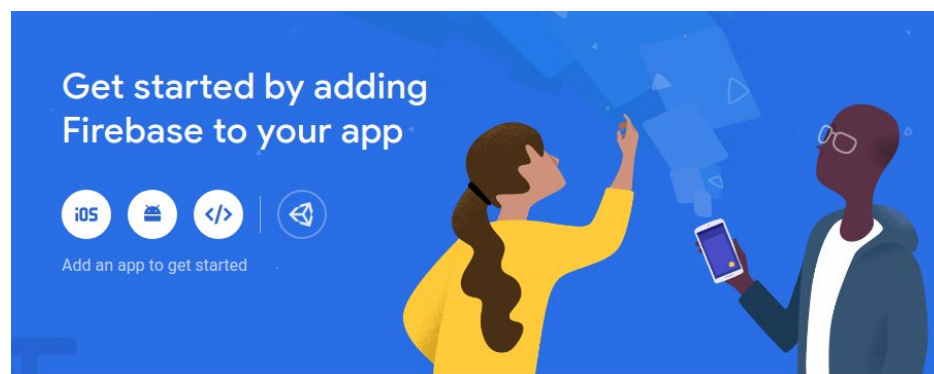
3. For this task you can disable google analytics, click create project. This will take a short while to initialize your project on firebase. Click continue and you will be forwarded to the dashboard where you can see all the services of firebase.



PART 2:

In this part we will integrate firebase with our web application. To do this follow the following steps.

1. On the main dashboard of firebase, click Project overview, and then click the web `</>` icon in the right pane. As you can see there are different types of integrations ie iOS, Android and Web. In this lab we only work with web integration.



2. Enter the name of your App and click Register app.

×

Add Firebase to your web app

1

Register app

App nickname ?

task5

☐

Also set up **Firebase Hosting** for this app. [Learn more](#) ↗

Hosting can also be set up later. It's free to get started anytime.

Register app

2

Add Firebase SDK

3. Here you can see the firebase sdk integration details, you need to copy it as it would be needed in web page. Click Continue to console.

×

Add Firebase to your web app

✓

Register app

2

Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.6.1/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyCXChIKK3oZ-DqZ590-xXEueT9YiVny0vA",
    authDomain: "iot-lab-b9c8b.firebaseio.com",
    databaseURL: "https://iot-lab-b9c8b.firebaseio.com",
    projectId: "iot-lab-b9c8b",
    storageBucket: "iot-lab-b9c8b.appspot.com",
    messagingSenderId: "584145155307",
    appId: "1:584145155307:web:4692c6c1e95b85e840baae"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

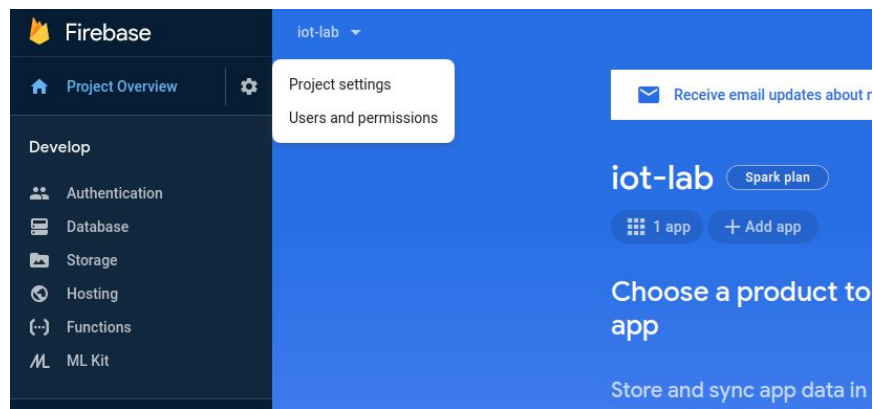
Learn more about Firebase for web: [Get Started](#) ↗, [Web SDK API Reference](#) ↗, [Samples](#) ↗

Continue to console

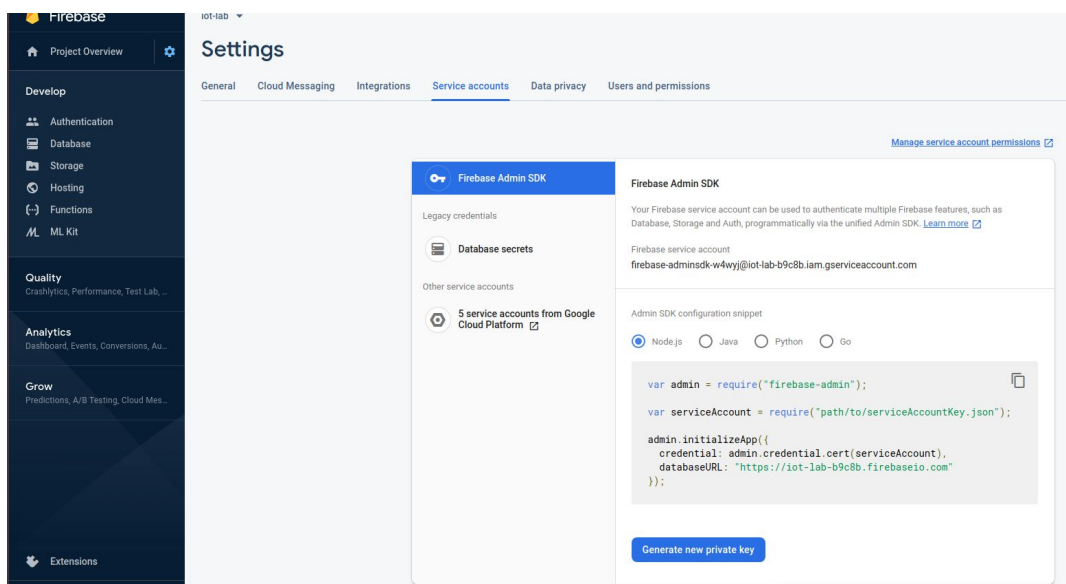
PART 3:

In this part we generate a private key for node.js. It is needed to authenticate your node application to the firebase. Node server will read data from thingy and save to firebase through the firebase-admin sdk (npm package).

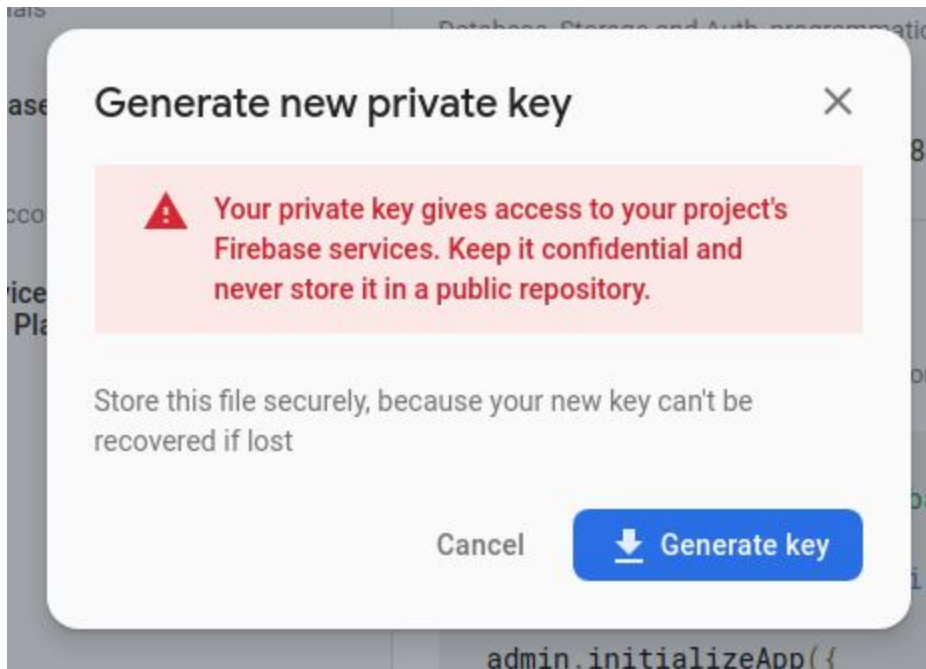
1. Click the small gear icon near Project Overview and select Project settings from the pop menu.



2. Here you will see different tabs about the project settings. Click the Service accounts tab and then click button Generate private key.



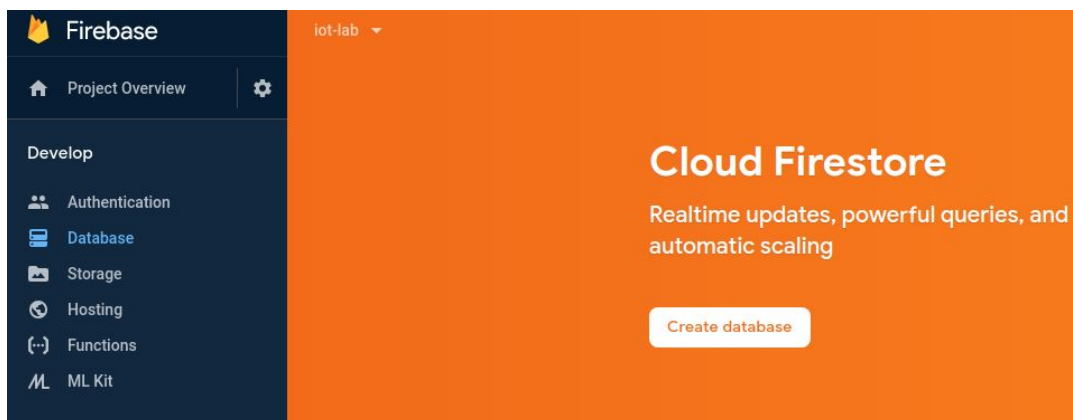
3. You will see a popup warning you to keep your private key safe. Click the Generate key button and you will be prompted to save the json file to your disk.



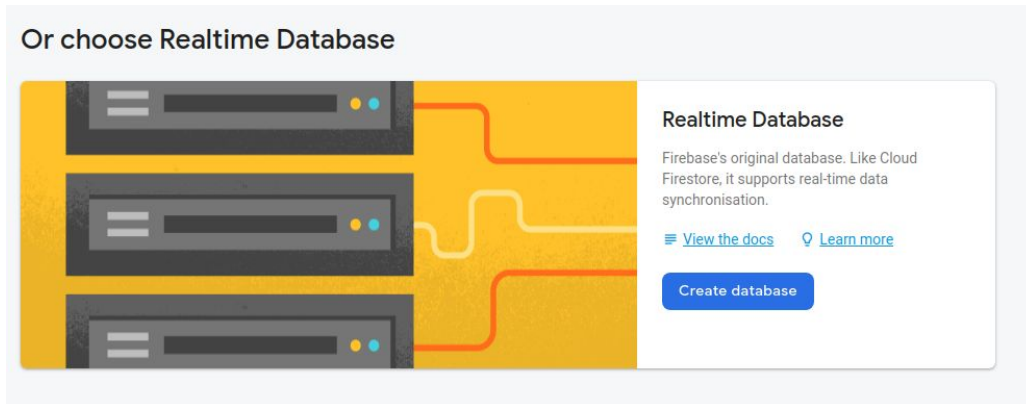
PART 4:

In this part we create a real time database to be used with our node and web app.

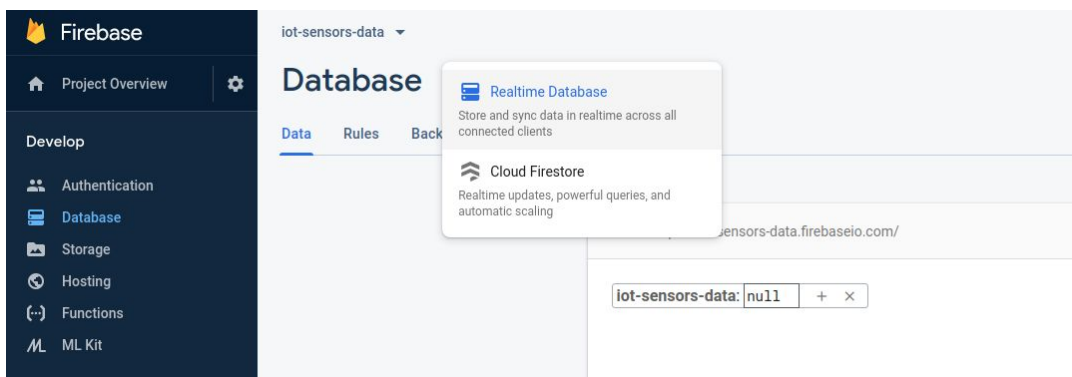
1. Click the database in the Develop section on the left pane of the console.



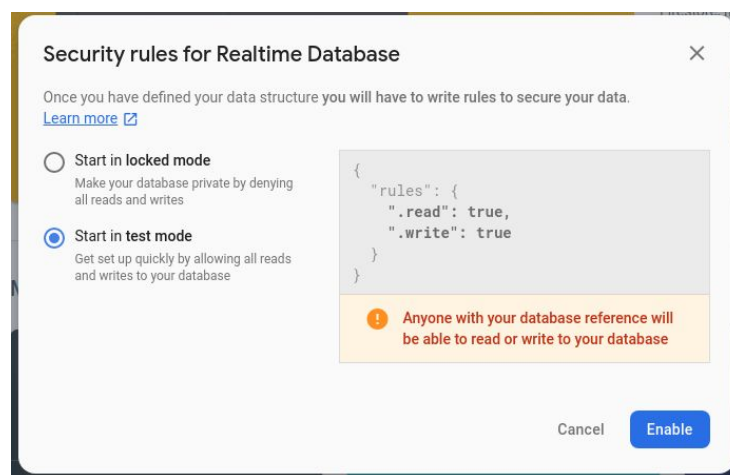
2. Scroll down to real time database section and click Create database.



3. If you already have created a database, you can switch to real time database section as shown below.



4. Click Create database (in step 2), choose test mode, and click Enable.



-
5. You have successfully created a real time database on firebase, now we need a little coding to connect our node server to it and save data. You can find this code in the lab task5 directory.

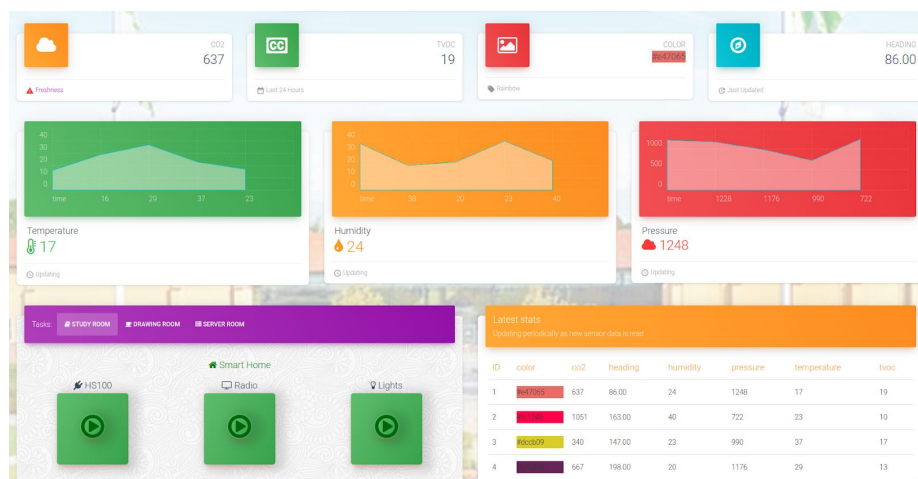
On node server we need to install a package called firebase-admin which will save the data to firebase real time database. You would need the private key from PART 3 above.

The web integration configuration (read only permissions) that you get from firebase will be used on the web page to read data from firebase (PART 2).

To run the code in the repo:

```
cd iot-lab
```

```
npm run task5
```



In this lab we are not using any cloud functions since we need to read data from thingy which should be accessed from local computer/pi through bluetooth, but we have used cloud functions and hosting for the sake of completion and learning of this feature in task3.

Task 06

In this task we implement a trigger, for example if temperature (sensor data from thingy) goes below 22, then a video clip should be taken and shown on the web page, and also switch on a smart plug.

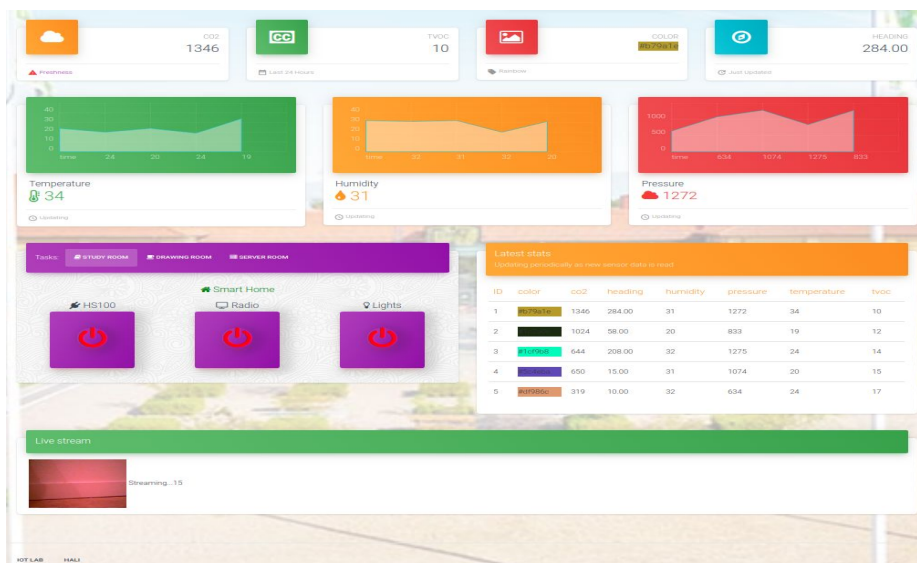
Implementation

The implementation involves using camera API, sensor data from thingy and using HS100 API. All of these have been used in the previous tasks except camera. We used both python based script and node js OpenCV based approach for real time video.

The code for camera is in camera.js, and it sends its data using socket.io.

To run the task use the following command.

```
cd iot-lab
npm run task6
```



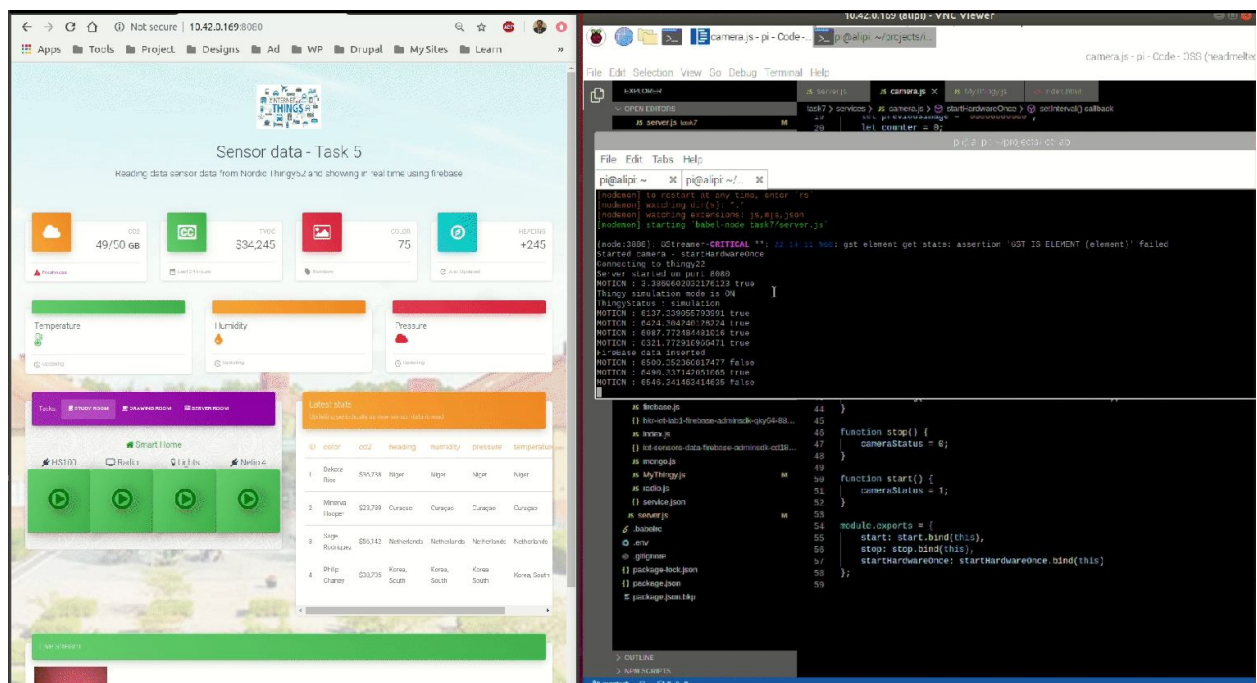
Task 07

In this task we implement PIR sensor to detect motion and then :

1. Report the event to firebase
2. Take a video clip
3. Switch on Philips Hue Lights using HS 100

Implementation

We used a custom node js script to detect motion as well as the python library for motion detection. This task is very similar to Task 6 except the trigger is now motion detection than temperature detection.



To run the code:

```
cd iot-lab
npm run task7
```

Development environment and scenario

I used Ubuntu 18.04 for developing, testing the code and for connecting PI. The development tool set has the following.

- Visual Studio Code
- Postman
- Node JS
- Mongo DB
- Firebase
- AWS
- VNC
- SSH
- Shell

I designed a specific scenario for this lab since we were supposed to work on different IP networks. I used my personal Raspberry PI 3, but it could not be connected to University (WPA-Enterprise) Wifi. The IoT_Lab wifi has no internet access. So the only viable scenario was to make my laptop both as connected to university wifi (using usb adapter) and hotspot (using built in wifi adapter) to let the PI connect with my laptop.

To do this I bought a Wifi USB adapter as shown in Fig 1. I used the built in Wifi as hotspot and the new USB adapter connected to any wifi (university or home). I made linux based bridge between the two Wifi cards so that the device connected to hotspot could use internet as shown below.

Wifi Router <-----wifi1> Laptop <wifi2-----> PI

This setup enabled me to connect to both PI and internet. It was easy to ssh or VNC to PI anytime. I did not use any monitor or keyboard for PI but I used VNC instead, from my laptop.(read more in Tips & Tricks below).

Problems and solutions

The following issues occurred but we solved their solution at one place.

1. The web app may not read the data initially from the realtime database in firebase.

To fix it click Develop > Database > Real time database > Rules > then change read, write to 'true', then click publish.

2. To set the display resolution

``nano /boot/config.txt`, and set `hdmi_mode 16`.`

3. Set home directory as

``export NODEJS_HOME=/usr/share/nodejs/bin``

4. To set the display resolution

``nano /boot/config.txt`, and set `hdmi_mode 16`.`

5. * sh: 1: node: Permission denied

`npm config set user 0`

`npm config set unsafe-Color sensor started!`

`perm true`

Or simply append every npm command with following flag

`--unsafe-perm`

Tips & Tricks

1. Enable SSH in Raspberry PI without a monitor.

Solution:

- a) Connect your SD to your computer using SD Card reader.
- b) In the boot partition of SD card use the following command to enable ssh.

```
touch ssh
```

2. Auto connect to Wifi without a monitor

Solution:

- a) Use command to edit the file

```
nano /etc/wpa_supplicant/wpa_supplicant.conf
```

- b) And add the following lines (replace credentials)

```
network={  
    ssid="your-wifi"  
    psk="your-wifi-password"  
}
```

3. Find IP address of Raspberry PI. You need to find the IP address of the PI, you may use the following command (change your network address) to scan your LAN and look for IP address of PI.

```
nmap -sS 192.168.0.0/24
```

-
4. Switch to root, super user mode

sudo su

5. To enable VNC run

`sudo raspi-config` and then choose Interfacing Options, enable VNC.

6. Download VNC Viewer [3] for your desktop computer from the following link:

<https://www.realvnc.com/download/file/viewer.files/VNC-Viewer-6.19.1115-Linux-x64>

References

- [1] "Markdown : <https://guides.github.com/features/mastering-markdown/>."
- [2] Ali, Hazrat, "Iot Lab : <https://github.com/iloveyii/iot-lab>," HKR.
- [3] "Real VNC - <https://www.realvnc.com>."