Internet of Things (ECE:5550)
Spring 2024
Lab 02

Start Date: Monday, February 26
Due Date: Sunday, March 10 by 11:59pm (ICON submission)
Lab Teams: same as Lab 01

Important Note: This lab requires the use of JavaScript in the Node.js framework on the Raspberry Pi. If neither member of your lab team is familiar with JavaScript and if you have concerns/questions using JS, please contact Prof. Bell.

Overview and Setup

The objective of this lab is to gain experience with a cloud-based publisher/subscriber service called Firebase. By integrating the Pi's sensors with Firebase, you will hopefully gain insight into one of the ways we can start architecting connected IoT solutions. As you may recall from lecture, Firebase is the pub/sub service used by the Nest Learning Thermostat and other commercial IoT applications.

For this lab, you will first need to install several frameworks onto your SD card, including: (1) the **Node.js** JavaScript run-time environment; (2) the **Firebase** JavaScript library for Node.js; (3) the "**nodeimu**" library for Node.js; and (4) the "**sense-hat-led**" library for Node.js. Installing each of these is covered in detail below. You will then write a Node.js application that communicates and facilitates actions locally, with the Pi's sensors, and remotely, with a connection to a Firebase database. To get started, first install the dependencies.

Installing the Node.js JavaScript Run-Time Environment

For this lab, you'll be using Node.js to run your JavaScript application that facilitates interaction between the Pi's sensors and the remote Firebase database. To install the Node.js run-time environment on your Pi, first ssh into your Pi and then download the Node.js (v20) source.

```
curl -sL https://deb.nodesource.com/setup_20.x | sudo -E bash –
```

Following the completion of this command, install Node.js:

```
sudo apt-get install -y nodejs
```.

You should be able to get affirmation of the successful Node.js installation by typing

```
node -v
```

which should print v20.11.X (e.g., v20.11.1).

Note: if you have general issues with the above, one common reason is that you are not connected to the public Internet. If you need help troubleshooting this, please let us know.

For the remaining installations, you'll use the Node.js Package Manager (npm). It is the default package manager for Node.js and is distributed along with Node.js. Given this, it is available and ready to be used after performing the above Node.js installation.

Create a new folder for your Lab 2 project (e.g., `mkdir lab2`) and navigate to it (`cd lab2`).

Installing the Firebase Library for Node.js

To create and manage a remote data store for our application, you'll be using Firebase. To integrate with Firebase from within your Node.js application, you'll use the Firebase JavaScript library which can be obtained using npm. Once in your working folder for this lab, run:

```
npm install firebase
```

Installing the "nodeimu" Library for Node.js

Next, install the "nodeimu" npm package. This library for Node.js allows you to use JavaScript to interface with the Pi's sense-hat module to collect information from the sensors. To install this library, run:

```
npm install @trbll/nodeimu
```

Installing the "sense-hat-led" Library for Node.js

Finally, install the "sense-hat-led" npm package. This library for Node.js allows you to use JavaScript to interface with the LEDs on the Pi's sense-hat module. To install this library, run:

```
npm install @trbll/sense-hat-led
```

Creating you Node.js App

From within your lab2 directory, create a new file with the '.js' file type (e.g., `touch app.js`) and then open the file in your preferred editor (e.g., VS Code, vim) to begin creating your Node.js application. The first thing you'll need to do is access the packages/libraries you've just installed. This can be done by simply creating variable references to them. An example is below. It is recommended that you type these lines and do not copy and paste them.

```
var firebase = require( 'firebase/app' );
var nodeimu = require( '@trbll/nodeimu' );
var IMU = new nodeimu.IMU( );
var sense = require( '@trbll/sense-hat-led' );
const { getDatabase, ref, onValue, set, update, get } = require('firebase/database');
```

To run your Node.js app, save and close out of your editor (if on Vim) or open a terminal (if on VS Code) and type `node app.js`. This will execute your application file, in this case app.js, within the Node JavaScript run-time environment. With the above example, your program should run, init the RTIMULib and sensor(s), and terminate.

To use each of these libraries, please look into their documentation online, either on their npm page or on their respective GitHub repositories. The 'nodeimu' library also has several examples that can be explored. You can find these tests (test.js, testSync.js) on the library's GitHub page or within …`/yourpath/lab2/node_modules/@trbll/nodeimu/`. To run them, navigate to the nodeimu folder and type `node test.js`, for example. Readmes can be found here:

- sense-hat-led : https://www.npmjs.com/package/@trbll/sense-hat-led
- nodeimu : https://www.npmjs.com/package/@trbll/nodeimu

Setting up Firebase

Before beginning, you will need to set up a free Firebase account (one account per lab team). To do this, go to https://firebase.google.com/, click the "Get Started" button, and login with a Google account.

Once logged in, click "Add Project" to make a new project. After naming and creating the project, you will be taken to a screen similar to Fig. 1. Click the button as pointed to by the green arrow to open the "Add Firebase to your web app" dialog box. You will then have to create a new app for this project. Give your app a name and then you will be greeted with appropriate snippets to add to your project (Fig. 2).

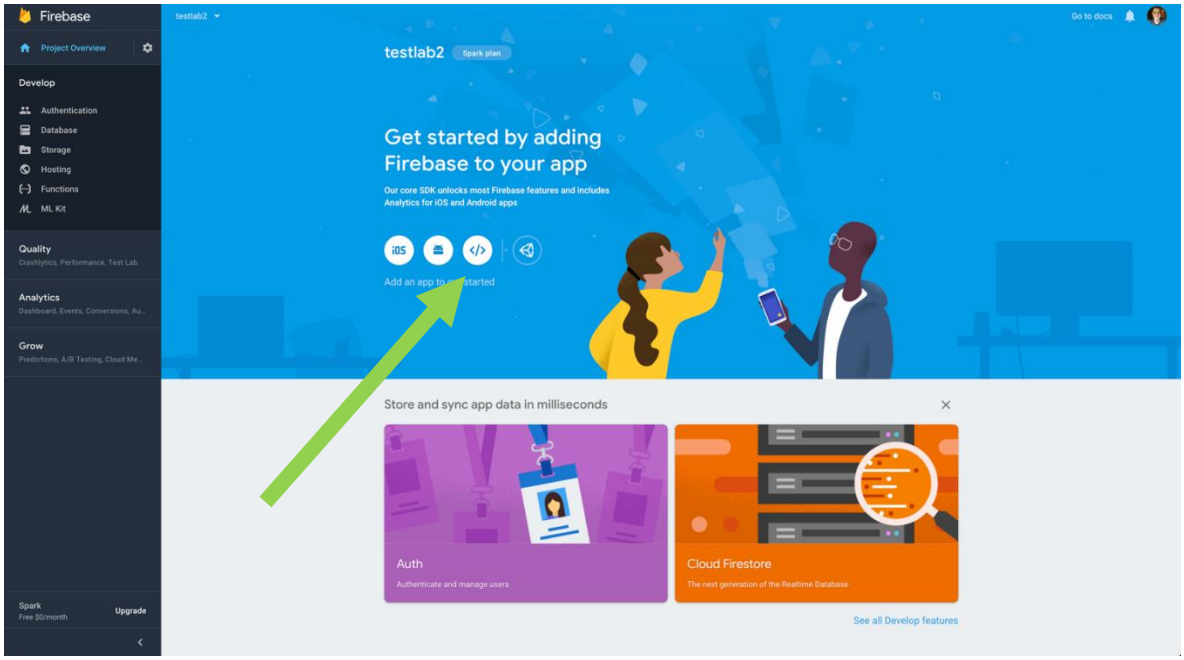

Figure 1: Project Dashboard with the important button pointed to.

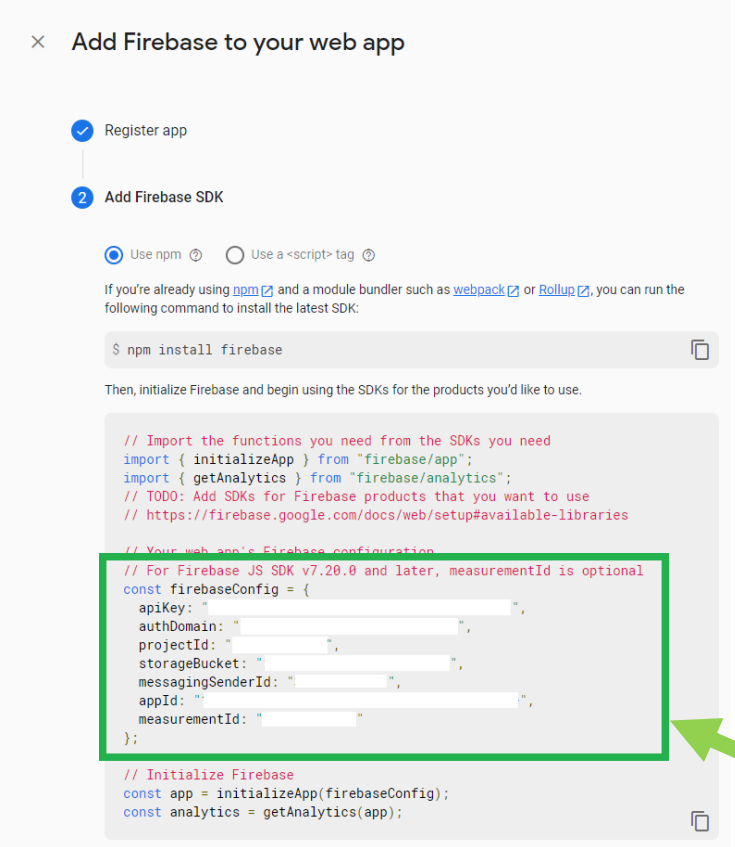

Figure 2: Important snippets, in green, you'll need to add Firebase to your project.

A tutorial is available here. Please note the following:

- You've already installed the Firebase library so you can skip the 'npm install firebase' command and the tutorial's "Prerequisites" section.
- This tutorial is written for implementing this on a website with JavaScript, but the Node.js library works exactly the same. You've already created a reference to the Firebase library at the top of your Node script, so the rest should work the same.
- Your project's *config* object can be accessed by clicking on "Add Firebase to your web app" pointed to in Figure 1/2.
- Firebase offers a variety of services, but for this lab you'll use the "Realtime Database."

Copy the *const firebaseConfig* object (green box in Fig. 2) into your code and then add the line *firebase.initializeApp(firebaseConfig);* to your Node.js application. You should not need anything else from the provided code (e.g., you don't need to run npm install firebase again, don't copy any of the script tags). Next, create a database in the Firebase web portal. To do this, navigate to Build > Realtime Database on the navigation pane on the left side of the Firebase dashboard. Once here, scroll down and "Create database" button on the "Realtime Database" card item. **To make things easier for the sake of this lab you may disable authentication on your database.** After clicking the "Create Database" button, a dialog window should appear titled "Security rules for Realtime Database." Select the option "Start in test mode" for this lab. If you use Firebase for your eventual team projects, you will need to utilize Firebase's security.

Lab Assignment

Add the code *const database = getDatabase();* to your app.js file. You now have a way to talk to your new Firebase Real Time Database (with the *database* variable).

For this lab, you will use Firebase services to push data from the Pi to the cloud and also control the Pi from the cloud. You will do this by creating a very simple Firebase database that stores the following values:

- Humidity
- Temperature
- Light to change on the sense-hat's light grid. This will be composed of the following:
    - Light's row (valid values 0-7)
    - Light's column (valid values 0-7)
    - New light color's R, G, and B values (valid values 0-255)
    - Boolean value indicating whether to update the light.

As mentioned in the overview, you have installed two Node.js libraries on your Pi to allow you to interact with the sense-hat module. For the "sense-hat-led" library, please see the following link: https://github.com/aonghusonia/sense-hat-led for documentation. For the "nodeimu" library, please refer to `…/yourpath/lab2/node_modules/@trbll/nodeimu/testSync.js` for an example. For communicating with Firebase, this tutorial will be useful for the rest of the lab: https://firebase.google.com/docs/database/web/read-and-write.

Specific requirements for your Node.js app on the Pi:

- Create the Firebase database with values "temperature", "humidity", "update_light", and "light_info" → light_info should then have the nested children "light_r", "light_g", "light_b", "light_row", and "light_col". Please note that Firebase needs initial values for these data entries (e.g., 0, false).
- Set up your Node.js app to receive a callback whenever the value of "update_light" changes. **If** the new value of "update_light" is *true* **then** (1) obtain the remaining *light_info* information from Firebase and (2) change the indicated light to the indicated color. To accomplish all of this, you can use the Firebase *onValue()*, *once(),* or *get()* methods as described in the "Read and Write Data" section of the tutorial (https://firebase.google.com/docs/database/web/read-and-write). Your Node.js app should also print out a message locally whenever it changes a light, indicating which light it changed, and what it changed it to.
    - Note: you should **ONLY** listen for updated values of *update_light*; NOT listening for updates on any of the other database entries or on the database as a whole
    - Hint: you'll want to make a database "ref" for the "/update_light" in your database.
- Set up your Node.js app to send the most recent temperature and humidity values to the Firebase database once per 5 seconds (5000 ms). You can use the JavaScript *setInterval()* method to control the timing. You can use the Firebase *update()* to send the updated temperature and humidity data to the database. Your app should also print out a message whenever it sends this information to the database.

Testing Your Program

Open your project dashboard and click the “Database” option on the navigation pane to the left. You should then see the following Database Overview screen:

Figure 3: The Database Overview screen.

After starting your Node.js app, you should be able to see your data values on the dashboard. Verify that Temperature and Humidity update once per 5 seconds. From the Database Overview, you can change the value of the light control objects. Change them and verify that the appropriate light changes on the sense-hat module.

Important Note: Due to its close proximity to the Raspberry Pi (which get fairly warm), the temperature sensor on the sense-hat reads a little high. Temperature readings of 29-32 degrees Celsius are normal. In general, for our purposes here, we will not worry about the exact values.

Submission & Check-off (one submission per lab team)

Submit to the ICON dropbox screenshots showing your Pi ssh terminal window and Firebase Database Overview (same as Figure 3). The screenshot should be taken immediately after changing the color of a light on the Firebase Realtime Database and should show the appropriate message displayed by the Pi ssh terminal window in response to receiving the *update_light* update and changing the light. Please submit one screenshot for each lab member.

In addition, please submit the source code for your Node.js app.

The submission deadline is Sunday, March 10 by 11:59pm (ICON submission).

As before, you will need to demonstrate your completed lab to the TAs and receive their check-off. A sign-up sheet will be available in the Teams Checkoff channel’s files to allow teams to reserve slots for this check-off. If you would like to meet with a TA outside of these times, please make these arrangements with him via email.

If you have any issues, particularly as they relate to installing Node.js or the npm packages, please contact our TAs or Prof. Bell so we can help get you on the right track. Feel free to post a question on the course Teams Questions channel, send a DM in Teams, or send an email. We are here to help!