trackr

User Guide

Ver 1.0

Dec 2015

Document Change History

Date	By	Description
12/3/2015	Murtaza Haveliwala	Initial Draft
12/9/2015	Murtaza Haveliwala	Update TBD links with actual paths and expanded trouble shoot section with web socket test
13/9/2015	Murtaza Haveliwala	Inserted probe and frontend snaps

Table of Contents

1 Overview
  1.1 trackr Architecture
    1.1.1 Probe device
      1.1.1.1 Probe constituents
        1.1.1.1.1 Main board
        1.1.1.1.2 Child module
          1.1.1.1.2.1 Sensor detection range
    1.1.2 Web Application
    1.1.3 Component Interactions
2 Usage
  2.1 Setting up - Probe
    2.1.1 Installing NodeJS
    2.1.2 Installing Tessel Command Line Interface (CLI)
    2.1.3 Fetching trackr probe scripts
    2.1.4 Configuring WI-FI credentials for location
    2.1.5 Uploading Scripts to Board
  2.2 Setting up - Web Application
    2.2.1 Installing NodeJS
    2.2.2 Fetching trackr remote application scripts
    2.2.3 Configurations
    2.2.4 Installing Dependencies
  2.3 Running trackr
    2.3.1 Probe
      2.3.1.1 Probe Status
    2.3.2 Web Application
  2.4 Troubleshooting Probe
    2.4.1 Wifi connection
    2.4.2 Running scripts on board
      2.4.2.1 via RAM
        2.4.2.1.1 Loading script(s) to board
        2.4.2.1.2 Unloading script(s) from board
        2.4.2.1.3 Stop current script execution
      2.4.2.2 Via Flash
        2.4.2.2.1 Loading script(s) to board
        2.4.2.2.2 Unloading script(s) from board
        2.4.2.2.3 Reading logs at execution
    2.4.3 Restarting board without Power Disconnection
    2.4.4 Resetting the board
    2.4.5 Troubleshooting scripts
      2.4.5.1 Board Tester
      2.4.5.2 Climate Modules Tester

2.4.5.3 Web Socket Connection Tester

1 Overview

trackr is a solution that facilitates remote measuring of temperature and humidity at a location using a probe and allows representing the collected measurements to users for analysis via a web application. The probe and web application communicate wirelessly using wifi and web sockets.

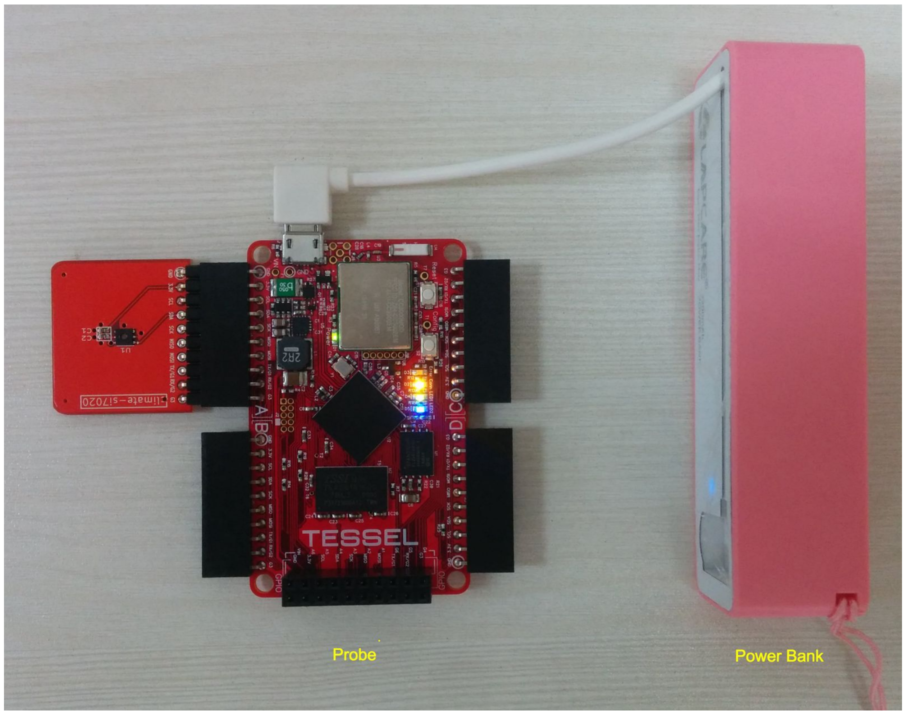
It can be used to provide live monitoring of premises which could then help facility admins in -

1. Reducing cooling costs at spots which might be cooler than required
2. Normalizing temperatures in premises thus reducing employee inconvenience
3. Keeping eye on crucial areas like server rooms, pantries/kitchen etc for overheating and avoid potential hazard/damage

1.1 trackr Architecture

trackr comprises of two entities:

1. 1.1.1 Probe device



The probe needs to be deployed to the location whose temperature is to be monitored. It need not be connected to a host computer but can operate in “detached” mode on a Micro USB based power bank.

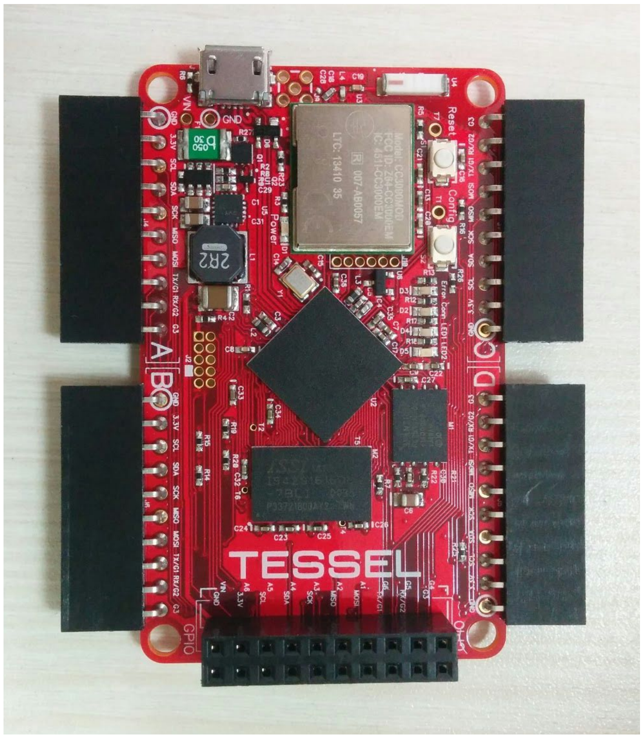
It comprises of sensors that reads temperature and humidity of the environment. It is capable of connecting to available wifi network at that location to transmit the readings to a central web application. The network details are pre-programmed into every device for a specific location and can be re-configured.

Every probe is configured with an ID that helps distinguish results when multiple probes are connected to same web application.

1.1.1.1 Probe constituents

The probe consists of hardware comprising of the main board and a child module.

1.1.1.1.1 Main board



The main board is a tessel.io board (v1.0). It comprises of a micro-controller, RAM, Flash memory, wifi module, micro USB port, GPIO pins and other ports for connecting child modules.
The board is completely programmable and can be fed with programs/scripts. The scripts are written in JavaScript (NodeJs). The board is responsible for executing the scripts, managing its internals and communicating with add-on modules.

The `trackr` probe scripts are fed in the board's Flash memory which retains the scripts even after device restarts. The scripts can then run in suspended mode right after connecting it with a power source.

1.1.1.1.2 Child module

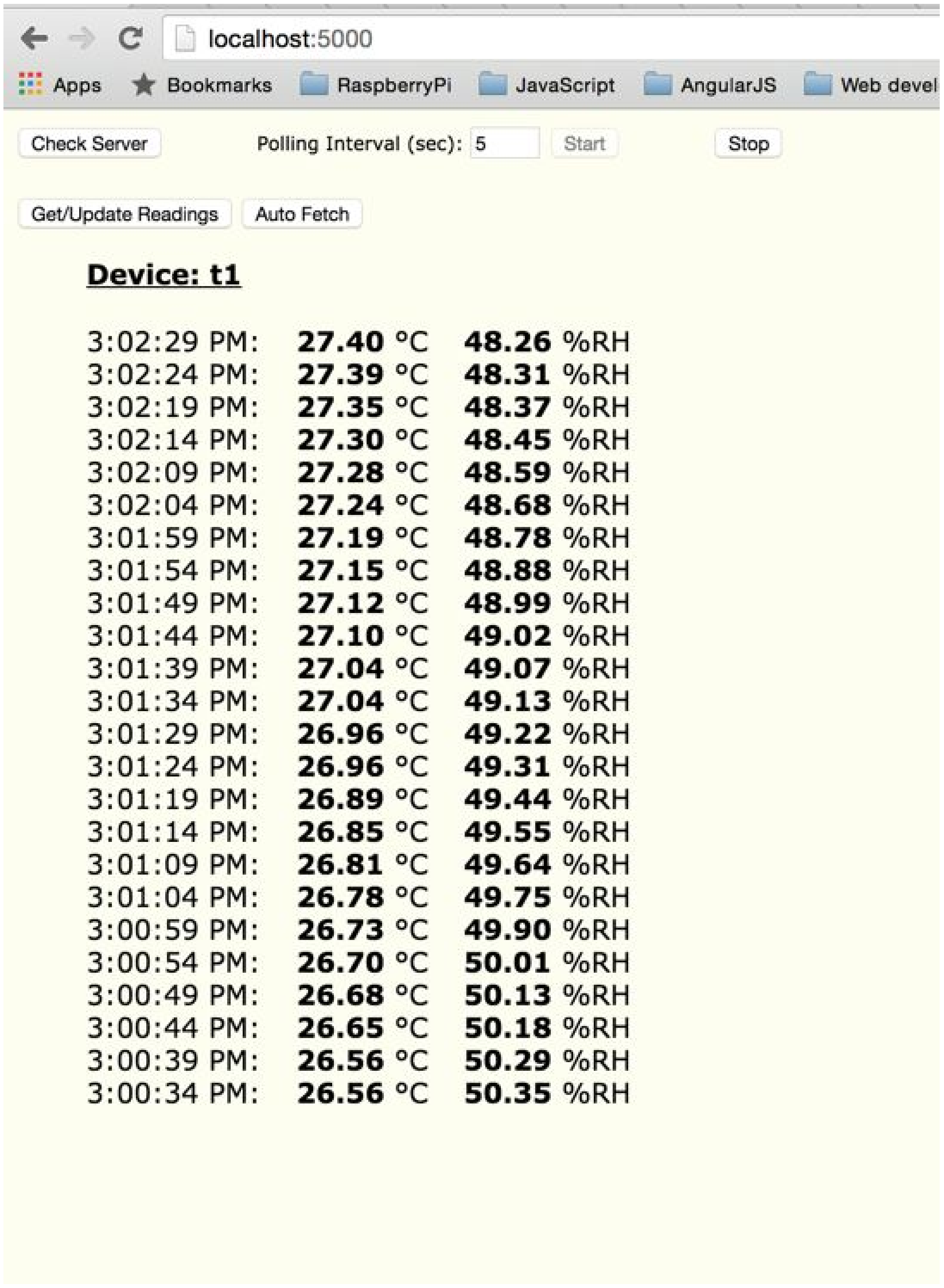
The child module that we use is called "climate module" (details). This module houses a sensor (data sheet) for reading temperature and humidity.

1.1.1.1.2.1 Sensor detection range

Temperature - 0 to 70 °C (32 to 160 °F) with ±1° accuracy
Humidity - 0% to 80% relative humidity

WARNING: Using the module in environments beyond the prescribed range may ***damage*** the sensor or the module.

2. 1.1.2 Web Application



Polling Interval (sec): 5

Device: t1

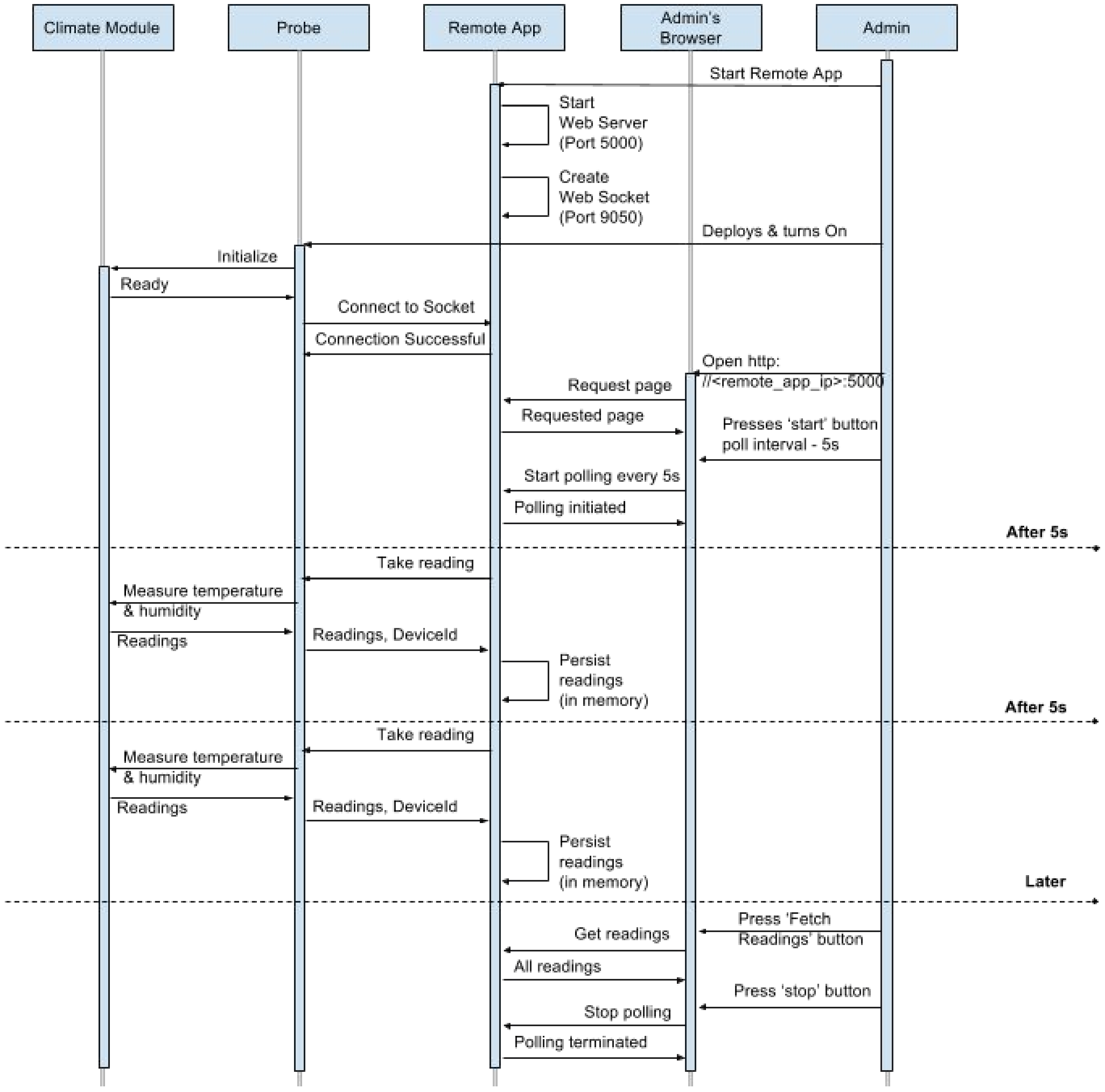
```
3:02:29 PM:   27.40 °C   48.26 %RH
3:02:24 PM:   27.39 °C   48.31 %RH
3:02:19 PM:   27.35 °C   48.37 %RH
3:02:14 PM:   27.30 °C   48.45 %RH
3:02:09 PM:   27.28 °C   48.59 %RH
3:02:04 PM:   27.24 °C   48.68 %RH
3:01:59 PM:   27.19 °C   48.78 %RH
3:01:54 PM:   27.15 °C   48.88 %RH
3:01:49 PM:   27.12 °C   48.99 %RH
3:01:44 PM:   27.10 °C   49.02 %RH
3:01:39 PM:   27.04 °C   49.07 %RH
3:01:34 PM:   27.04 °C   49.13 %RH
3:01:29 PM:   26.96 °C   49.22 %RH
3:01:24 PM:   26.96 °C   49.31 %RH
3:01:19 PM:   26.89 °C   49.44 %RH
3:01:14 PM:   26.85 °C   49.55 %RH
3:01:09 PM:   26.81 °C   49.64 %RH
3:01:04 PM:   26.78 °C   49.75 %RH
3:00:59 PM:   26.73 °C   49.90 %RH
3:00:54 PM:   26.70 °C   50.01 %RH
3:00:49 PM:   26.68 °C   50.13 %RH
3:00:44 PM:   26.65 °C   50.18 %RH
3:00:39 PM:   26.56 °C   50.29 %RH
3:00:34 PM:   26.56 °C   50.35 %RH
```

Web application has two primary roles -

1. Collect readings from individual probes
2. Provide a frontend to admins for controlling probes and presenting recorded temperature and humidity readings

For this purpose, the application exposes two ports -
1. HTTP port - For serving web pages relating to probe control, presenting readings and API endpoints for such pages to retrieve data
2. Websocket port - This is the port on which probes communicate to central server once they boot and for responding to server's requests for readings

1.1.3 Component Interactions



2 Usage

2.1 Setting up - Probe

1. Insert Climate module into port ‘A’ of main board
   Ensure that chip face of the module is on the same face as the main board’s chips while inserting.

   WARNING: Plugging it in reverse may ***damage*** the board, the port and module.

2. Connect main board to a system - via micro USB port of main board, USB port of system and micro to normal USB cable provided
3. Install Node and Tessel Command Line (CLI) on the system
4. Download `trackr` probe script package from here
5. Configure device with
   a. Remote web application’s IP address and websocket port
   b. Wifi credentials of available network to be used by probe at location
   c. Provide the probe with a unique ID
6. Upload these scripts to main board’s flash

Once the probe is uploaded with `trackr` probe scripts, it no longer required to connect it to the system. It can simply be connected to other power sources to work.

2.1.1 Installing NodeJS

Tessel board requires NodeJS version 0.12.7 to function properly. NodeJs Installer can be downloaded from here.

After downloading, install the package.

Installation can be verified by issuing this on prompt -
```
$> node --version
$> npm --version
```

If the prompt specifies the version, the installation was successful.
To install tessel CLI, it is required that node and npm programs are installed successfully.

2.1.2 Installing Tessel Command Line Interface (CLI)

On the prompt, use npm to install tessel CLI -

```
$> npm install -g tessel
```

This will install the Tessel's command line on the system.

To verify installation, on the prompt, issue -

```
$> tessel list
```

If the board was connected to the system, it will print out on console the board's manufacturing ID.

Official detailed instructions on installing Node and tessel CLI is provided here.

2.1.3 Fetching trackr probe scripts

`trackr` probe scripts are located here.

2.1.4 Configuring WI-FI credentials for location

Once the script package is downloaded, update 'configs.json' with actual values -

1. Wifi settings to be used by probe at remote location
   a. `ssid` - ssid of the wifi network
   b. `password` - password (in plain text) of the wifi network
   c. `security` - authentication protocol to be used for connecting to wifi. Available options are
      i. `wep` - for using WEP encryption protocol
      ii. `wpa2` - WPA2 Personal encryption protocol
2. Device settings
   a. `id` - Identifier to use that denotes this probe. Useful when remote app is monitoring multiple locations via multiple probes. Must be unique.
3. Remote Application details
   a. `socketUrl` - socket url (with protocol, host and port) of the remote application which it is listening on. It will be used by probe to connect to and send readings.

2.1.5 Uploading Scripts to Board

After downloading scripts and making necessary configurations, to push the scripts to board's flash memory, issue these on command prompt -

```
$> cd <top level of folder containing scripts>
$> npm install
$> tessel push .
```

Since the scripts are “pushed” to flash memory, they will survive even after probe is hard booted.

2.2 Setting up - Web Application

2.2.1 Installing NodeJS

Download and install NodeJS from here.

2.2.2 Fetching trackr remote application scripts

trackr’s remote application scripts are available here and can be downloaded from the same.

2.2.3 Configurations

Remote application uses to two software ports for its web server and web socket. These by default are set to 5000 and 9050 respectively.
If required, they can be reconfigured by updating these in ‘configs.json’.

2.2.4 Installing Dependencies

The remote application is developed using many other frameworks and packages like ExpressJs, WebSockets etc.

To install these, we use ‘npm’ (node package manager) that is installed along with NodeJS.

On prompt, use -

```
$> cd <trackr scripts package directory>
$> npm install
```

The dependencies information is provided in package.json. NPM install will use it to fetch and install all these dependencies.

2.3 Running trackr

2.3.1 Probe

To use the probe at a remote location,

1. It must be configured with a unique ID and active wifi credentials for that specific location (instructions). The wifi network should allow the device to reach the central web application.
2. The probe must be uploaded with the necessary scripts to its flash memory (instructions)
3. Web application should be running and listening for connections

Once these are done, all is required for the device to be turned on by connecting it to a power source (microUSB power bank or other socket based power source via corresponding adapter). It will connect to remote web application and wait for reading requests from it.

2.3.1.1 Probe Status

To aid the detection of the probe's states, few LEDs are programmed to denote its status:
1. When no lights are on - Probe is in idle state with no connection to wifi.
2. Red LED is on - There was an error when connecting to Wifi.
3. Blinking Yellow LED - Attempting wifi connection
4. Steady Yellow LED - Connected to Wifi network
5. Steady Green LED - Connected to web application's socket
6. Blue LED is on - Climate module is busy measuring readings of the surrounding

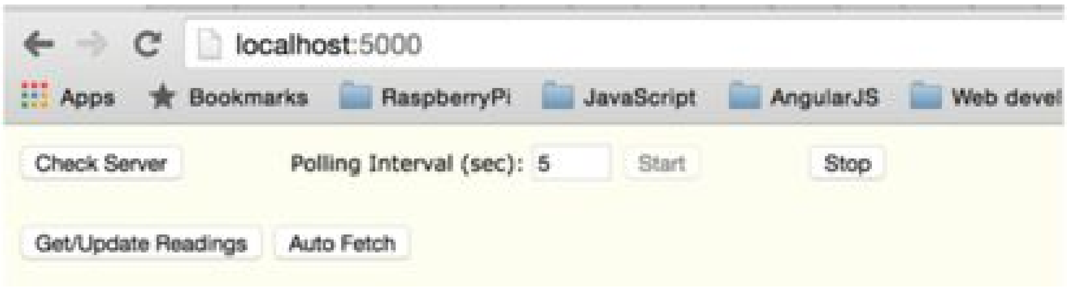
Initially the probe would be in idle state. Soon, it will attempt to connect to wifi. It may require a few attempts before successfully connecting to it. Once it has successfully connected to wifi, it will then establish a websocket connection to remote app. When connected successfully to remote app, it waits for reading requests.

2.3.2 Web Application

Deploy server scripts on a system by ensuring the system and probe would be reachable via the same network.

To initiate the web application -
1. Setup and install all app dependencies on the remote system (instructions)
2. Run the scripts on the system - Issue following commands on the prompt
   ```
   $> cd <directory where scripts are located>
   $> node index.js
   ```

   After doing this, the application would open default ports (if not re-configured) 5000 as web server and 9050 for websocket connections.



3. Open browser and perform the following
    a. Connect to the application via url - http://<remote_app_ip_address>:5000/
    b. Configure the poll duration(in seconds) in duration input
    c. Press ‘start’ button to trigger polling by server to probes for readings
    d. Press ‘Fetch Readings’ button to get the readings from server and display them
    e. Press ‘Auto-fetch’ to automatically refresh readings after every few seconds

4. To stop reading from probes -
    a. Open browser connect to the application via url - http://<remote_app_ip_address>:5000/
    b. Press ‘stop’ button

2.4 Troubleshooting Probe

Probe, being a hardware device and tasked with performing many tasks like connecting to wifi, taking readings & handling scripts, there could be many issues that could be encountered.

This section provide details about many scripts/commands to help in detecting the actual cause and troubleshoot it.

Since the probe doesn’t have a console of it’s own, it is required that the device be in “attached” mode (device connected to a system via corresponding ports and cables) to a system which has ‘tessel’ command line tool installed (installation steps outline here) to perform troubleshooting.

2.4.1 Wifi connection

Tessel board has a built-in chip that enables connecting to any wifi network. The chip, however, has a few limitations (reference) -

1. It supports 802.11b/g only
2. It does *not* support 802.11n.
3. Supported Wifi channels are from 1 through 11. Channels 12, 13 and 14 are *not* supported.

Tessel provides connecting the board to wifi via command line (CLI) as well as programmatically. The `trackr` probe scripts utilise the Tessel APIs to connect to wifi.

For troubleshooting, tessel command line (CLI) can be utilised.

Before troubleshooting, ensure board is connected to the system via respective usb ports and cables and tessel CLI is installed (installation steps detailed here).

These commands are provided to deal with wifi capabilities of the board and to be issued from the system's console:

1. Visible wifi networks for the board - lists all the wifi networks available for board to connect to
   Command: `$>tessel wifi -l`

2. Connecting to a specific wifi network from the list -
   Command:
   ```
    $> tessel wifi -n <network-name> -p <network-password> -s
   <security-type>
   ```

   `-p` switch - Optional. Not required for unsecured networks
   `-s` switch - Optional. Provides the authentication protocol to be used while connecting. Tessel supports 'wep' and 'wpa2'. Default is 'wpa2'.

3. Disconnecting wifi -
   Command: `$>tessel wifi -d`

4. Knowing MAC address of the chip -
   Command: `$>tessel wifi -m`

5. Current connection information -
   Command: `$>tessel wifi`

   If board is connected to a wifi network, it displays details like network name, ip address among others on console.

6. All wifi command options -
   Command: `$>tessel wifi --help`

Tessel board provides feedback about wifi connections status via it's onboard LEDs. IT uses the yellow and red LEDs for this purpose. The states and their corresponding meaning is as follows

Yellow LED	Red LED	Means

Off	Off	Not connected to Wifi
On & Blinking	Off	Attempting to connect to a wifi network
On & Steady	Off	Wifi successfully connected to a provided network
Off	On	There was an error while connecting to provided wifi network. To recover from it, initiate wifi connection again

Note: Once the board gets successfully connected to a wifi network, via CLI or programmatically, it automatically persists the details/credentials so that it would automatically re-connect to same network once device is rebooted.

2.4.2 Running scripts on board

Tessel provides two modes of persisting scripts for execution on the device. They are -

1. #### 2.4.2.1 via RAM

   This is useful for immediate execution of any script(s), useful in development stage. Here the scripts are pushed to device's RAM. However, the scripts are cleared once a new script is pushed to RAM or the board is rebooted.

   To run scripts in this mode, following are few helpful tessel CLI commands -

2.4.2.1.1 Loading script(s) to board

Command:
```
$> tessel run <script_filename | directory_containing_scripts>
```

Sample:

- For running index.js on the board -
  ```
  $> tessel run index.js
  ```
- For loading wifiTest directory to board which contains index.js as entry point
  ```
  $> tessel run wifiTest
  ```

2.4.2.1.2 Unloading script(s) from board

To unload, simply restart the device. The RAM would be cleared on rebooting the program and thus the script.

2.4.2.1.3 Stop current script execution

Command: `$>tessel stop`

2. 2.4.2.2 Via Flash

This is useful for execution of script(s) and retaining them for execution even after the board reboots. This loads the script(s) on board's Flash memory instead of RAM. The scripts are re-executed immediately on board boot-up. This allows running the scripts on the board in "suspended" or "detached" mode.

2.4.2.2.1 Loading script(s) to board

Command:
```
$> tessel push <script_filename | directory_containing_scripts>
```

Sample:
- For running index.js on the board -
  ```
  $> tessel push index.js
  ```
- For loading 'wifiTest' directory to board which contains index.js as entry point
  ```
  $> tessel push wifiTest
  ```

2.4.2.2.2 Unloading script(s) from board

Command: `$>tessel erase`
This clears the board's flash memory.

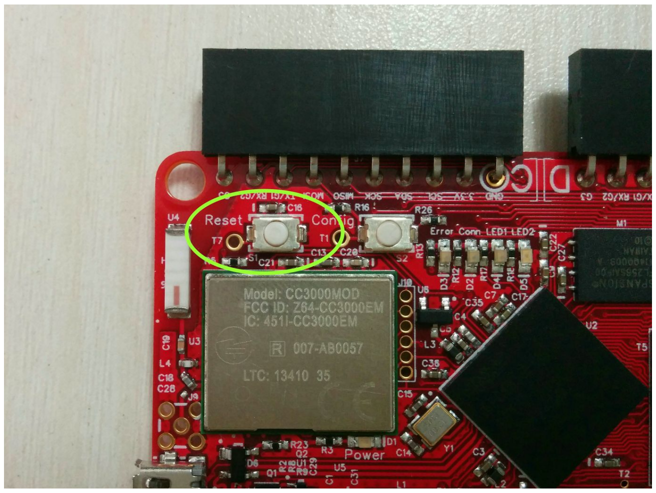
2.4.2.2.3 Reading logs at execution

When scripts are run in this mode, it might be required to read any status or any log messages from the device. To achieve this, we can use 'logs' command from tessel CLI once the device is connected to the system.

Command: `$>tessel logs`

This allows any log messages generated from the board's script execution to appear on the system's console

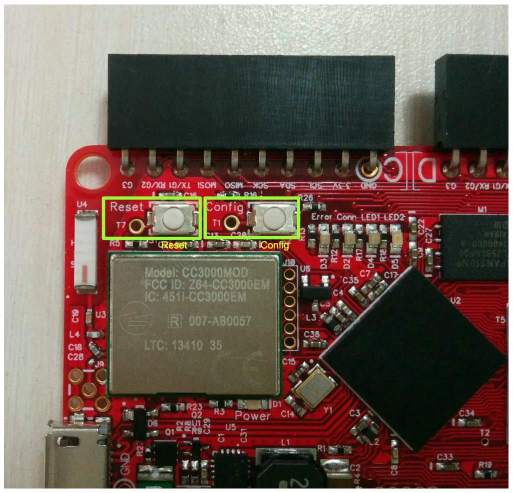
2.4.3 Restarting board without Power Disconnection



Board is provided with a small white circular button named 'reset' next to the processor. This is useful for restarting the device. However, this doesn't reset the device.

2.4.4 Resetting the board

Board can be reset to its initial status by flushing its scripts from Flash (if any) by pressing simultaneously the little white circular buttons on the board - 'reset' and 'config'



2.4.5 Troubleshooting scripts

To troubleshoot the device, various scripts are provided along the tessel source to check independent functions. The probe needs to be in "attached" mode, attached to a system with tessel CLI installed (instructions available here).

These scripts need to be executed in "RAM" mode as described above.

2.4.5.1 Board Tester

This script would check if it is capable of running a script and appropriately respond to it. This would flash onboard LEDs continuously until the script is stopped. If the LEDs do not flash then board is either busy or damaged.

To test, follow these steps -

1. Fetch scripts from here
2. On command prompt, execute -
   Command:
   ```
   $> cd <top_level_directory_of_fetched_scripts>
   $> tessel run blinky.js
   ```
3. If it runs successfully, the LEDs on the board would start flashing.
4. In case of any errors, appropriate messages would appear on the console
5. Press Ctrl+C to stop script execution

2.4.5.2 Climate Modules Tester

This script would test if the device can take readings of surrounding and continuously post them on the console after specified timeout (1s).
This would test if the board can connect to climate module and its ability to read measurements from it.
It won't need to connect to wifi. However, it needs the climate module be connected to the board at respective port ('A').

To test, follow these steps -

1. Fetch scripts from here
2. On command prompt, execute -
   Command:
   ```
   $> cd <top_level_directory_of_fetched_scripts>
   $> npm install
   $> tessel run .
   ```
3. If it runs successfully, it would output readings (temperature and humidity) on console every second. If there are any problems, either respective error message would appear on console.
4. In case of any errors, appropriate messages would appear on the console, giving hint about the problem
5. Press Ctrl+C to stop execution of the script

2.4.5.3 Web Socket Connection Tester

This would help to check if the board is capable of creating socket connections to remote app or not.
For testing, it would be required to -

1. Have the probe configured for wifi credentials and remote app ip addresses (configurations to be done in 'configs.json' available in the same package)
2. Probe be connected to configured wifi
3. Remote App to be installed and running on respective system
4. Remote App to be reachable over the wifi network used by probe

To test, follow these steps -

1. Fetch scripts from here
2. Configure configs.json with corresponding settings for wifi and remote system's ip and port running the remote app
3. On command prompt, execute -
   Command:
   ```
   $> cd <top_level_directory_of_fetched_scripts>
   $> npm install
   $> tessel run .
   ```
4. If it runs successfully, the script would confirm on console via message about successful connection.
5. In case of any errors, appropriate messages would appear on the console, giving hint about the problem
6. Press Ctrl+C to stop execution of the script