ECEN 5023-02

ECEN5023-02 FINAL PROJECT REPORT (LITTLERASCALS)

GUNASEKARAN, BERLIA

Course Project Report

ECEN 5023-002:

Low Power Embedded Design Techniques

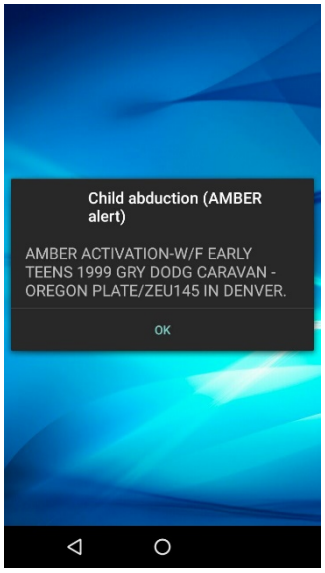
Rishabh Berlia
Shivasankar Gunasekaran

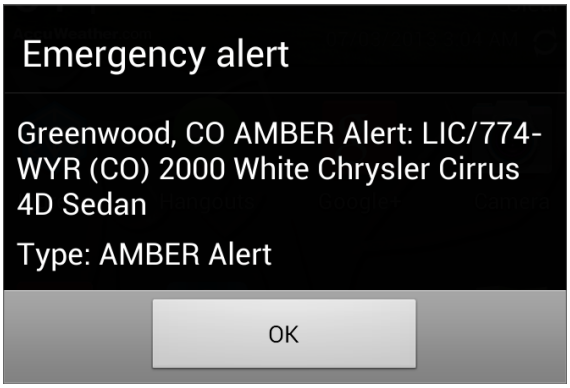
Table of Contents

Motivation: 4
Overview: 5
Hardware: 6
Block Diagram of the System: 7
Detailed Block Diagram: 7
Components 8
Software/App: 9
Firmware: 12
Web Application: 16
Web Application Commands/Features: 18
Created Modules for Database handling: 19
Security Features in Web Application: 21
Mobile Application: 23
Planned Development Schedule: 25
Difficulties Encountered: 26
Summary: 27
Conceptual Lessons Learned: 28
Appendix: 29

Motivation:

You might have seen alerts like the following.





“Amber Alerts” as they call it are used to notify the public about a nearby Child Abduction with a few details as to identify the criminal. However, Amber Alerts fail to be very effective due to the delays associated.
We aim to develop a device to tackle this issue effectively and without delays. So we can ensure the safety of our children.

Overview:

With the advent of the Internet of Things and tiny Low-Energy end devices, the possibilities have become endless. Low power battery operated devices have extended the Wearable electronic space.

To develop a complete “Children Tracking Device” to help parents securely and effectively track their kids when they go out of the specified range.

The Child tracker would use a BLE radio to communicate with the mobile app. The “RSSI” signal strength would also serve as the proximity sensor. Once your child moves away from the selected range, the SIM 808 on the tracker would activate.

The SIM 808 (GPS), pulls the coordinates and pushes it to the central Web Server using the Internet connection through SIM808 module (via GPRS).

We plan to build our own Web Application to serve the requests from the tracker. The application would be based on the Flask Web Framework in Python. We will have a simple mobile application for pairing with phone and display push notifications.

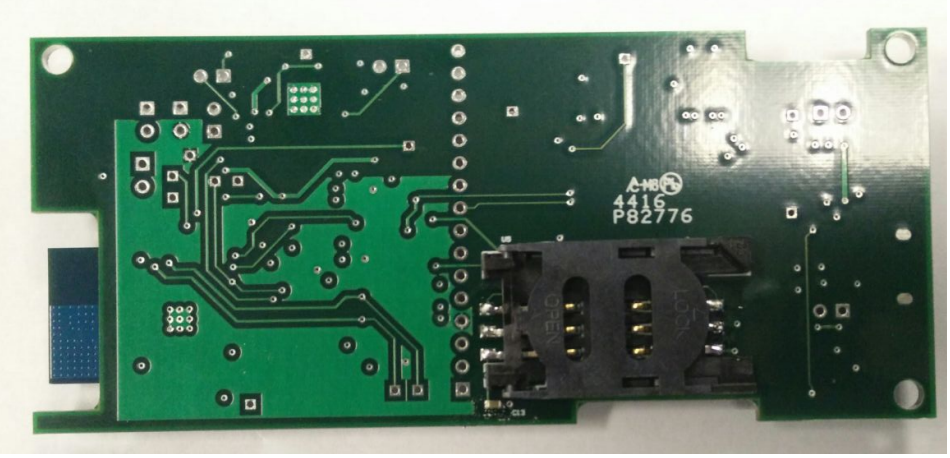
Hardware:

The hardware consists of a Bluetooth LE SoC (SAMB11) interfaced to a SIM 808 module (GSM and a GPS)

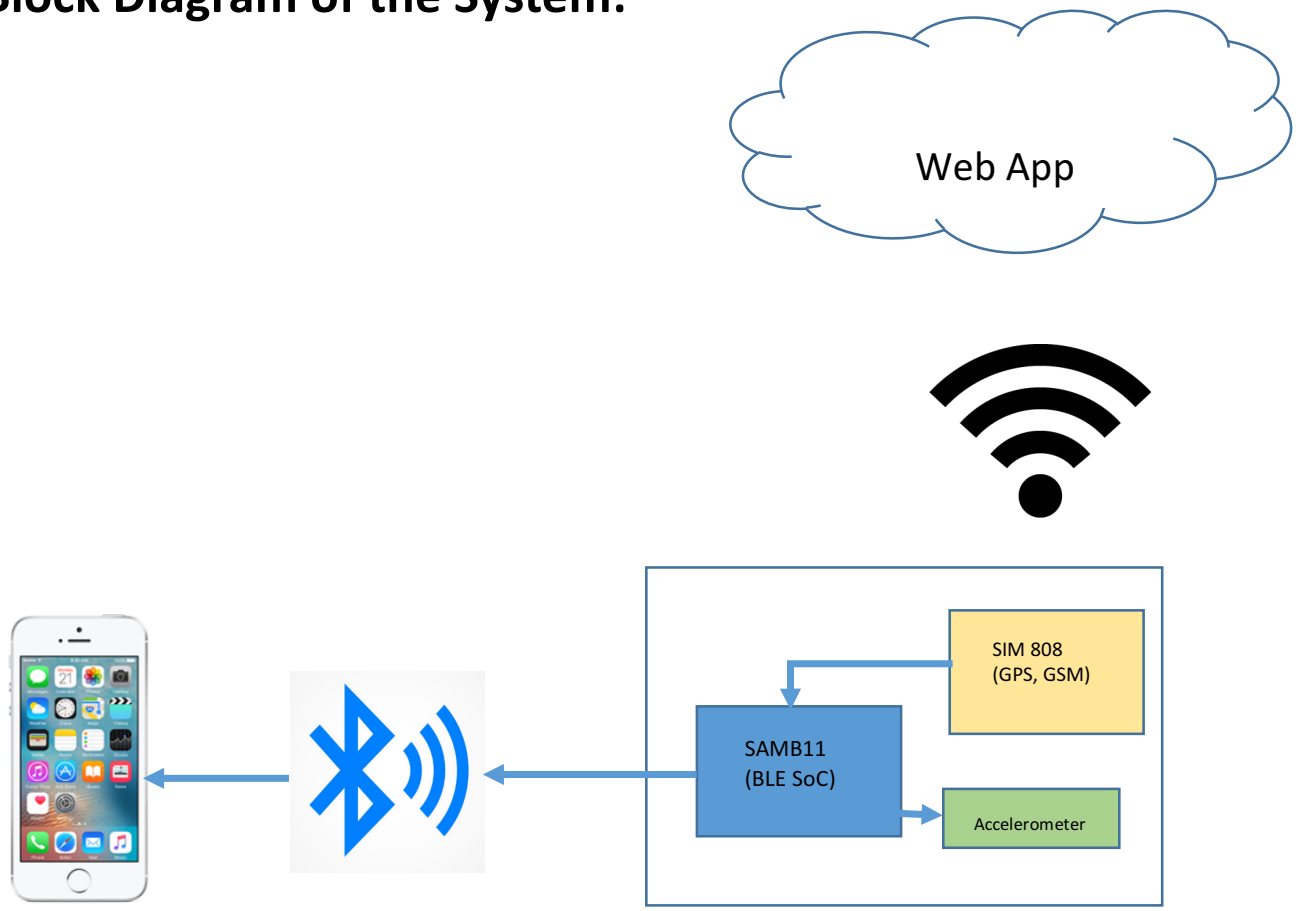
According to our use case, the best way to harvest energy would be using Piezo-electric (Kinetic) harvesting. Kids are usually more active than adults and we anticipate some energy could be harvested using this.

The GPS and GSM modules on board of the tracker are the most power hungry devices on the board. These would only switch on when the tracker loses connection with the mobile device i.e. when your child is out of the desired range.

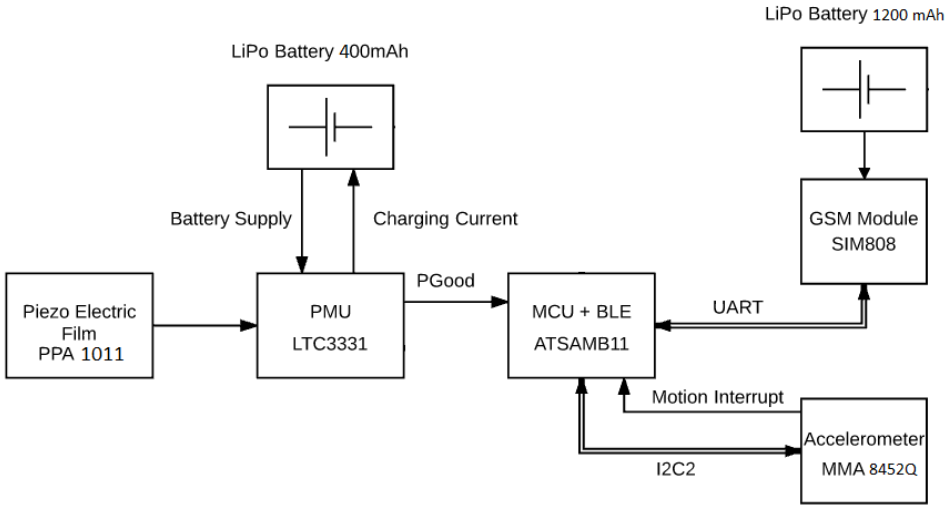
Figure: Our Board after initial bring up (Top and Bottom View)



Block Diagram of the System:



Detailed Block Diagram:



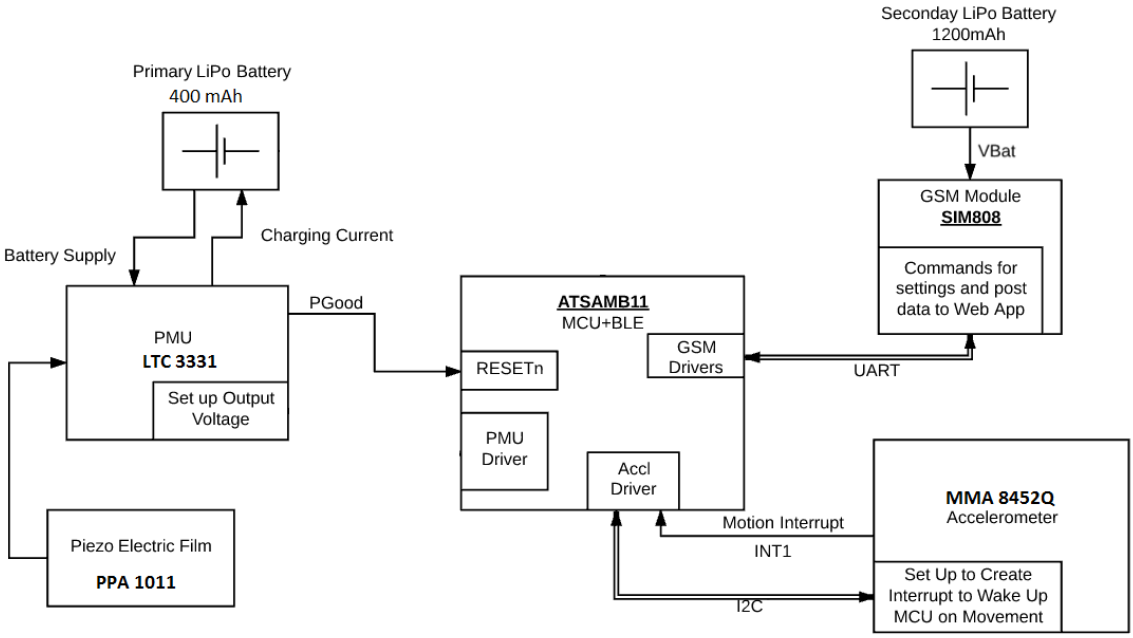
Components Used:

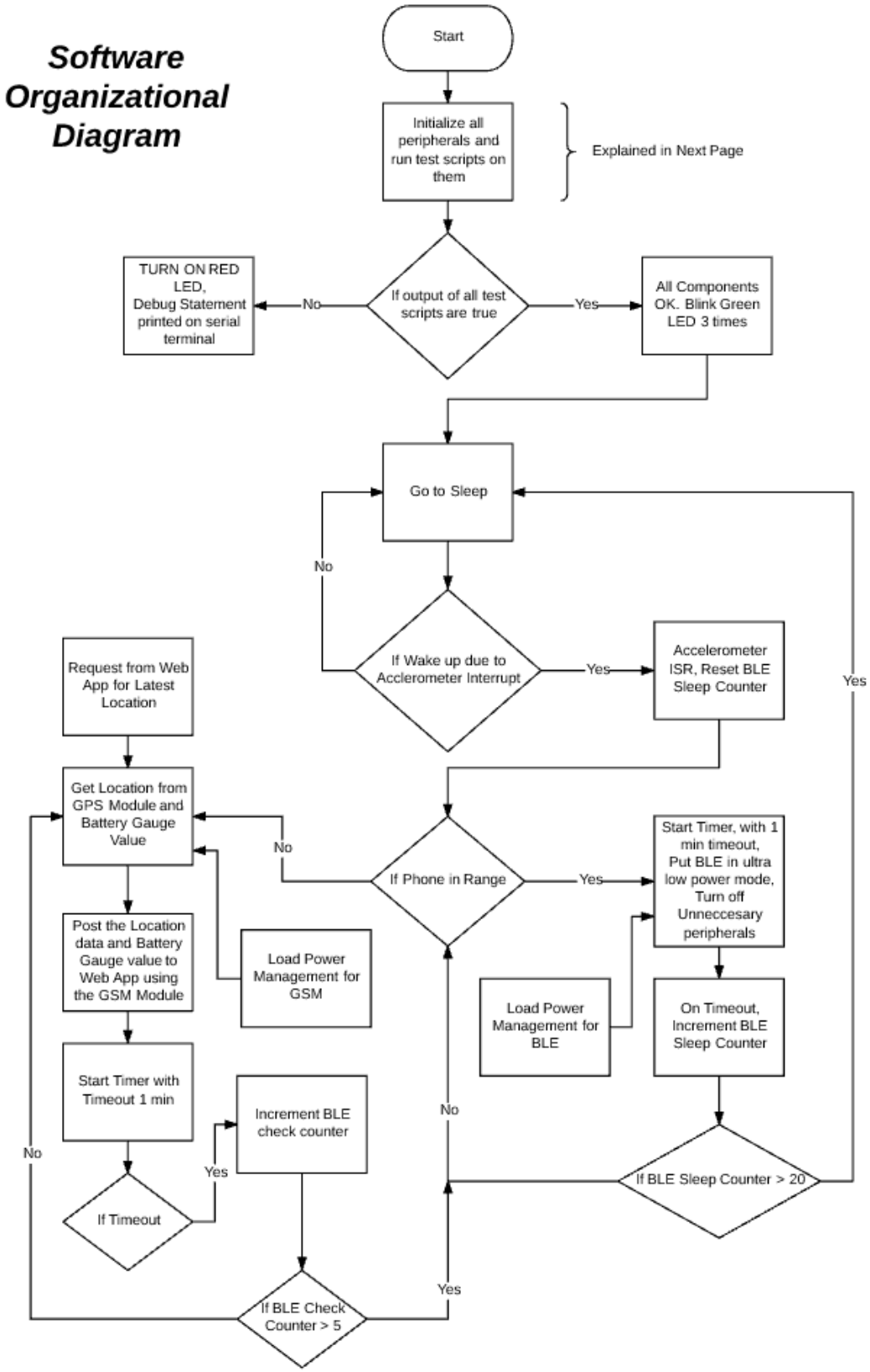
1. MCU:
    - ATMEL SAM B11 (ATSAMB11MR210CA)
2. External sensors and/or actuators:
    - Accelerometer: MMA8452Q
    - Geolocation Module: SIM 808
3. PMU:
    - LTC 3331
4. Energy Storage Device:
    - Battery
5. Energy Harvesting Device
    - PPA-1011 (Piezo Mide Sensor)

Software/Application:

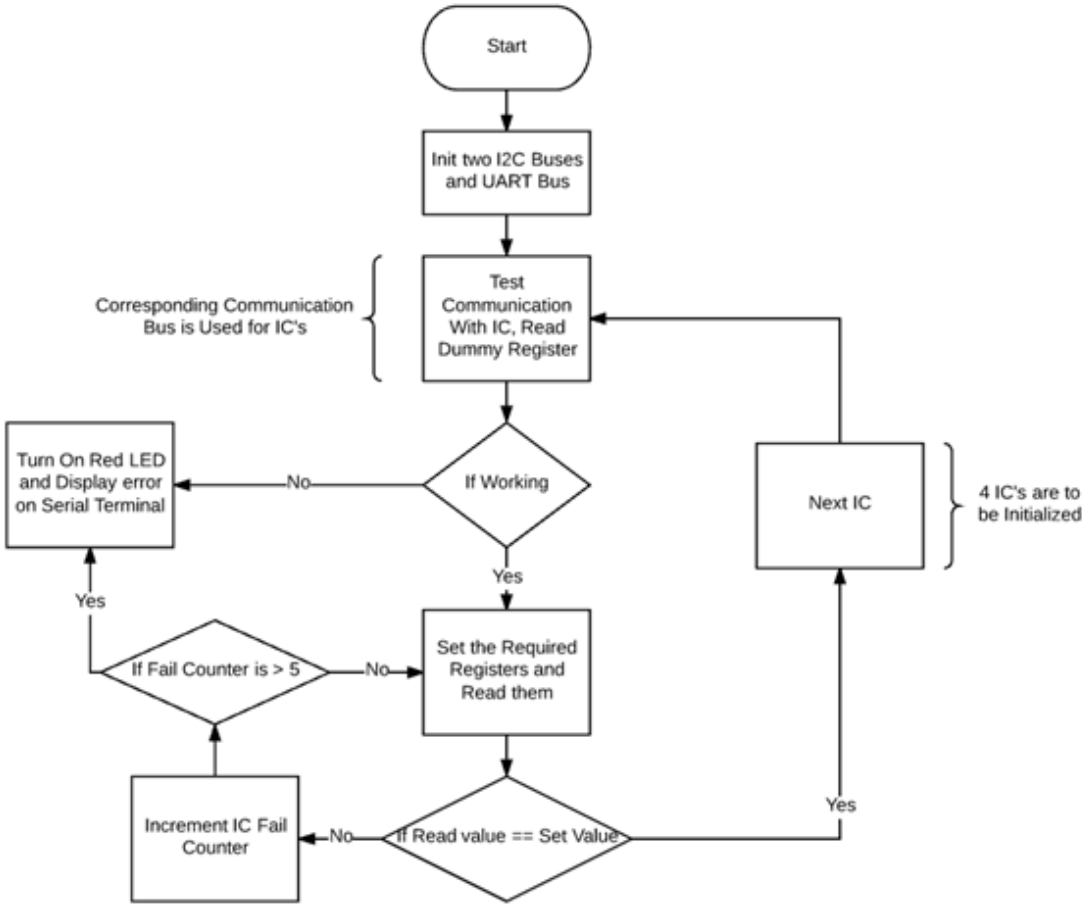
The software to support this is aimed to primarily be a Web based application. We would create our server to support a REST API and handle the requests appropriately.
The server would serve the front-end of the Web-App where parents can log-in to their accounts and see the data of their trackers in real time.

Software Flow Diagram




Software Organizational Diagram
Start
Initialize all peripherals and run test scripts on them
Explained in Next Page
If output of all test scripts are true
No
TURN ON RED LED, Debug Statement printed on serial terminal
Yes
All Components OK. Blink Green LED 3 times
Go to Sleep
No
If Wake up due to Acclerometer Interrupt
Yes
Accelerometer ISR, Reset BLE Sleep Counter
Yes
Request from Web App for Latest Location
Get Location from GPS Module and Battery Gauge Value
No
If Phone in Range
Yes
Start Timer, with 1 min timeout, Put BLE in ultra low power mode, Turn off Unneccesary peripherals
Post the Location data and Battery Gauge value to Web App using the GSM Module
Load Power Management for GSM
Load Power Management for BLE
On Timeout, Increment BLE Sleep Counter
Start Timer with Timeout 1 min
Increment BLE check counter
No
If BLE Sleep Counter > 20
No
Yes
If Timeout
Yes
If BLE Check Counter > 5
Yes

Initialization and Testing Sequence



Firmware:

Accelerometer Configuration:

CTRL_REG1_Standby: 0xF8
Set the sampling rate to 1.56Hz, and last bit is 0 to put it in standby mode.

CTRL_REG1_Active: 0xF9
Set the sampling rate to 1.56Hz, and last bit is 0 to put it in active mode.

CTRL_REG2: 0x40
To do a software reset to the module.

CTRL_REG4: 0x04
To enable Freefall/Motion detect interrupt.

CTRL_REG5: 0x04
Route the FF/MT interrupt to INT1 pin.

FF_MT_CFG: 0x78
Enable Motion detection and enable all three axis for motion detection.

FF_MT_THS: 0x10
Set the threshold for motion detection interrupt

FF_MT_COUNT:0x0A
Set the Count value for de-bounce for the motion interrupt.

SIM 808 AT Commands:

"AT\n"
Command sent to set the baud rate of the communication between SIM808 and SAMB11. SIM808 does an auto detect of baud rate based on the AT command.

"ATE0\n"
Command to disable echo for the responses of the SIM808.

"AT+CCID\n"
Command to get the SIM number of the SIM card used. This verifies that the SIM card is connected and data can be read from it.

"AT+CGNSPWR=1\n"
This turns on the GPS section of the SIM808 module on.

"AT+CGNSPWR?\n"
Command used to read the current value of GPS Power.

"AT+CGNSINF\n"
Command to retrieve the GNSS location of the module.

"AT+SAPBR=3,1,\"Contype\",\"GPRS\"\n"
Command to set the APN settings. This is to set the connection type as GPRS.

"AT+SAPBR=3,1,\"APN\",\"wholesale\"\n"
Command to set the APN settings. This is to set the APN as wholesale.

"AT+SAPBR=1,1\n"
Command to set all the APN settings.

"AT+HTTPINIT\r\n"
Command to initialize the HTTP stack.

"AT+HTTPPARA=\"CID\",1\n"
Command to set the HTTP Connection parameter “CID”.

"AT+HTTPPARA=\"URL\",\"http://littlerascals.informu.io/postData\"\n"
Command to set the HTTP connection parameter “URL”.

"AT+HTTPPARA=\"CONTENT\",\"application/json\"\n"
Command to set the HTTP connection parameter “CONTENT”.

"AT+HTTPDATA=80,10000\n"
Command to set the maximum length of data that will be sent, and the timeout to wait before which the data should be sent.

"AT+HTTPACTION=1\n"
Command to commit to the HTTP request.

"AT+HTTPREAD\n"
Command to read the response of sending the HTTP request.

"AT+HTTPTERM\n"
Command to close the HTTP stack connection.

"AT+CSQ\n"
Command to obtain the Signal Quality of SIM module.

"AT+CBC\n"
Command to obtain the Battery Level and Battery Voltage of the module.

BLE Firmware:

The BLE is setup to advertise for defined Advertise Timeout and then go to sleep if no connection happens. A connected event handler and disconnected event handler is used to know when BLE module is connected to the Phone. So BLE acts as a proximity sensor, or an E-Leash.

Web Application:

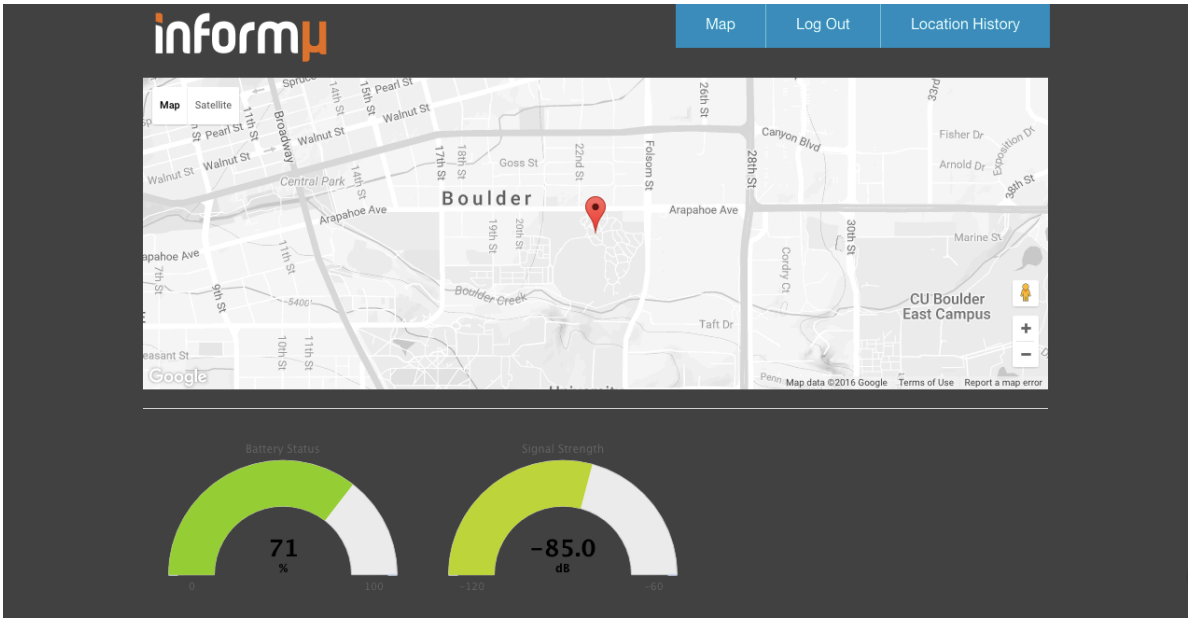


Figure 1 Web Application Preview On Desktop

The Web Application is written in Python on Flask which is a lightweight Python web framework based on Werkzeug and Jinja 2.

We are hosting the Web App on a Soft Layer (IBM) Web Instance running Ubuntu 14.04. We also used DNS forwarding to point the public IP of our server i.e. 50.23.237.26 and pointed it to littlerascals.informu.io

We have used uWSGI application server to launch the application and Nginx to act as a front end reverse proxy. This helps our web application to handle more request at a given time through multi-threading.

uWSGI

uWSGI is a deployment option on servers like nginx, lighttpd, and cherokee;. uWSGI is both a protocol and an application server; the application server can serve uWSGI, FastCGI, and HTTP protocols.The most popular uWSGI server is uwsgi, which we have used for our project.

Nginx

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.

The Web Application is optimized for Mobile Devices as well.

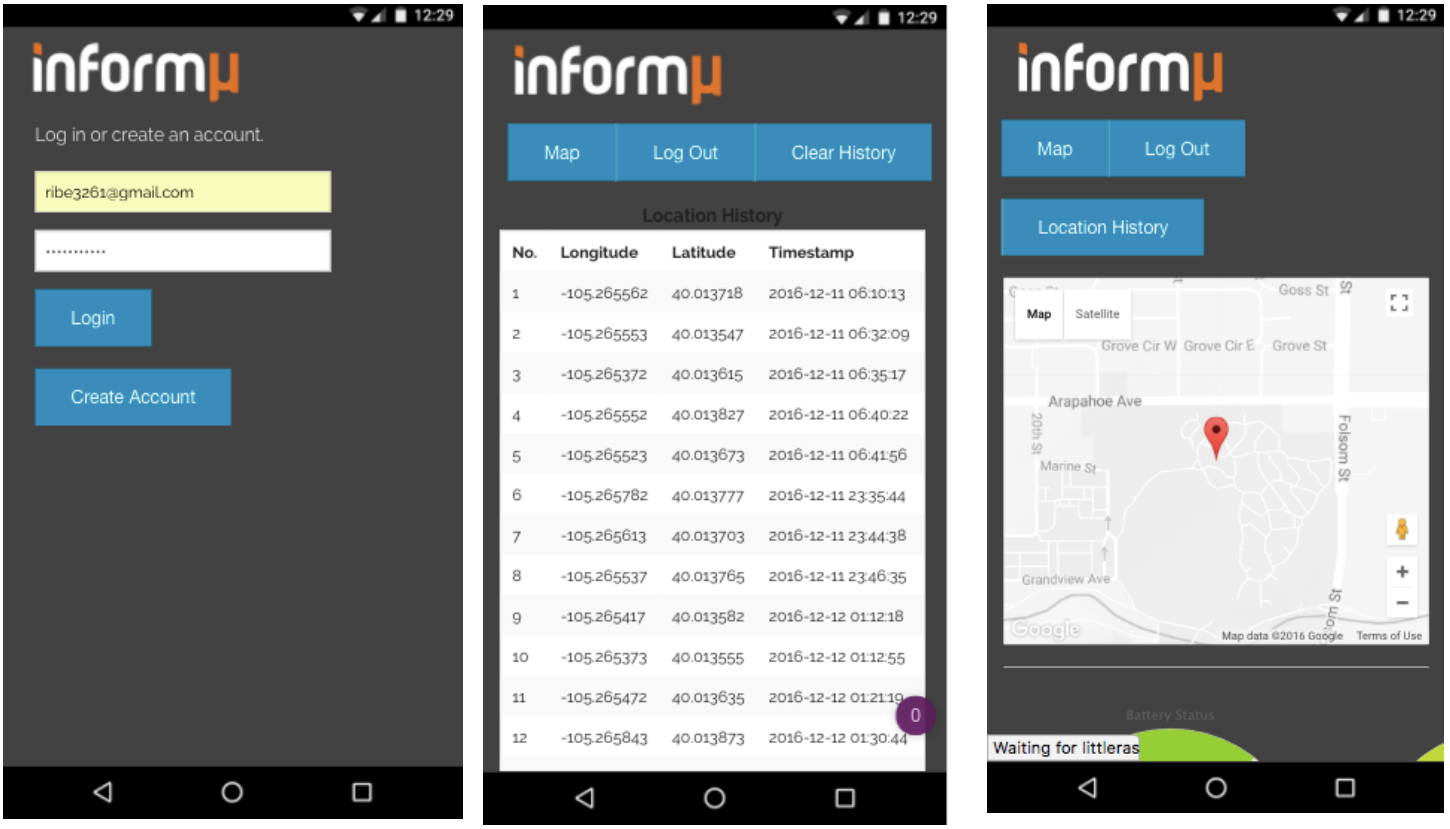


Figure 2 Web Application Preview On Mobile

Web Application Commands/Features:

The Application has the following end points and their functionality:

/ - Home Page (This is the Page for the Project Introduction)
/signup – When the user clicks 'Create Account' button this user credentials are entered in the database. There is a check for existing username.
/login – Every time, the user is not logged in or authenticated we will redirect to this page to authenticate the user. The entered credentials are checked against the stored ones. (Note: Password is SHA256 of actual password)
/map – This is the main Dashboard of the application. This loads the Google Map API with the last posted Latitude and Longitude. We also show a gauge for Battery and Signal Quality of the SIM Module. This will help the user know how long before the Battery will die and if the Signal Strength is too low.
/locationHistory – This end point queries the Data Base and returns the entire history (location trail) of the device. This helps to track the movements with timestamp.
/clearHistory – This end point gives the logged in user the capability to clear the database if required. To start fresh.
/logout – This end point gives the user the ability to log out and clears the session cookie to stop the session.
/postData – This end point allows the device to POST data to the server with appropriate key. The data is in JSON format and has 4 key-value pairs:
{"lat":"", "lng":"" , "bat":"", "rssi":""}

Created Modules for Database handling:

We have used sqlite3 as our database. We created a two different modules to handle the login (login.db) and location (places.sq3) databases separately.

Module dbtest.py (This module is imported as location handler)

insertRecord (latitude, longitude, batteryStatus, powerStatus):
This method is called when the data received needs to be inserted into the database.
getLatLong ():
This method is used to fetch the latest latitude and longitude from the database.
getlocationHistory ():
This method is used to query the entire database to get the location history with the timestamp.
getBatteryStatus ():
This method is used to get the latest Battery Status value from the database.
getPowerStatus ():
This method is used to get the latest Power Status (RSSI) value from the database.
clearHistory ():
This method will clear all the database and create a new table based on the existing schema.

Module models.py (This module is imported as login handler)

getHash (passText):
This method is used to convert the plain text password into the Hash value by a SHA256 Hash Function.

insertUser (username, password):
This method is called when the user credentials are accepted and needs to be in inserted into the database.

checkUsername (username):
This method is used to check the existing username and generate error if it does.

retrieveUsers (username, password):
This method is used to retrieve the given user credentials to match with the in the database to authenticate the user.

clearDataBase ():
This is an admin function which is used to clear the user's database. To delete all the existing users.

Security Features in Web Application:

We have added some basic security features in the application, since this information about their child's location is sensitive and we would not want it to be leaked.

List of Implemented Security Features:

1. **Create Account:** An account needs to be created before you access the service. This allows us to track the IP of the user which made the request as to track and block dubious IPs.
2. **Login:** We do not store the user's passwords in clear-text but we store the SHA256 hash of the user's password. This ensures if the database is ever breached the attacker would not be able to see the passwords directly.
3. **CSRF Defenses:** For Cross Site Request Forgeries Defense we have implemented a Random session cookie to be set on the user's browser. This session cookie is checked when the requests are made, therefore an attacker cannot make the user click on a link and make him do an unintended command.
4. **XSS Defenses:** We have tried to sanities the user inputs to the database which are echoed out to the webpage. This way an attacker cannot inject JS into our webpage to run his malicious code.
5. **Session and Logouts:** If the user logs out or closes his browser now the session cookie will be deleted and he will need to log in again. This ensures that no one else uses their account after they leave their computers for a while.
6. **Check on POST Requests:** To stop someone from making dubious post request to the server and act as the littleRascals device, we have implemented a 'key' check, only the device with the correct

key will be allowed to make the POST request, all other devices will be denied permission.

Requests over HTTPS: This was one important security feature which we could not implement. This was due to the fact that the device (SIM 808) can only make HTTP requests. We need to configure nginx to partially serve the page of HTTP (for POST Requests) and rest over HTTPS. This will make sure the Client and the Server is end to end secure.

Mobile Application:

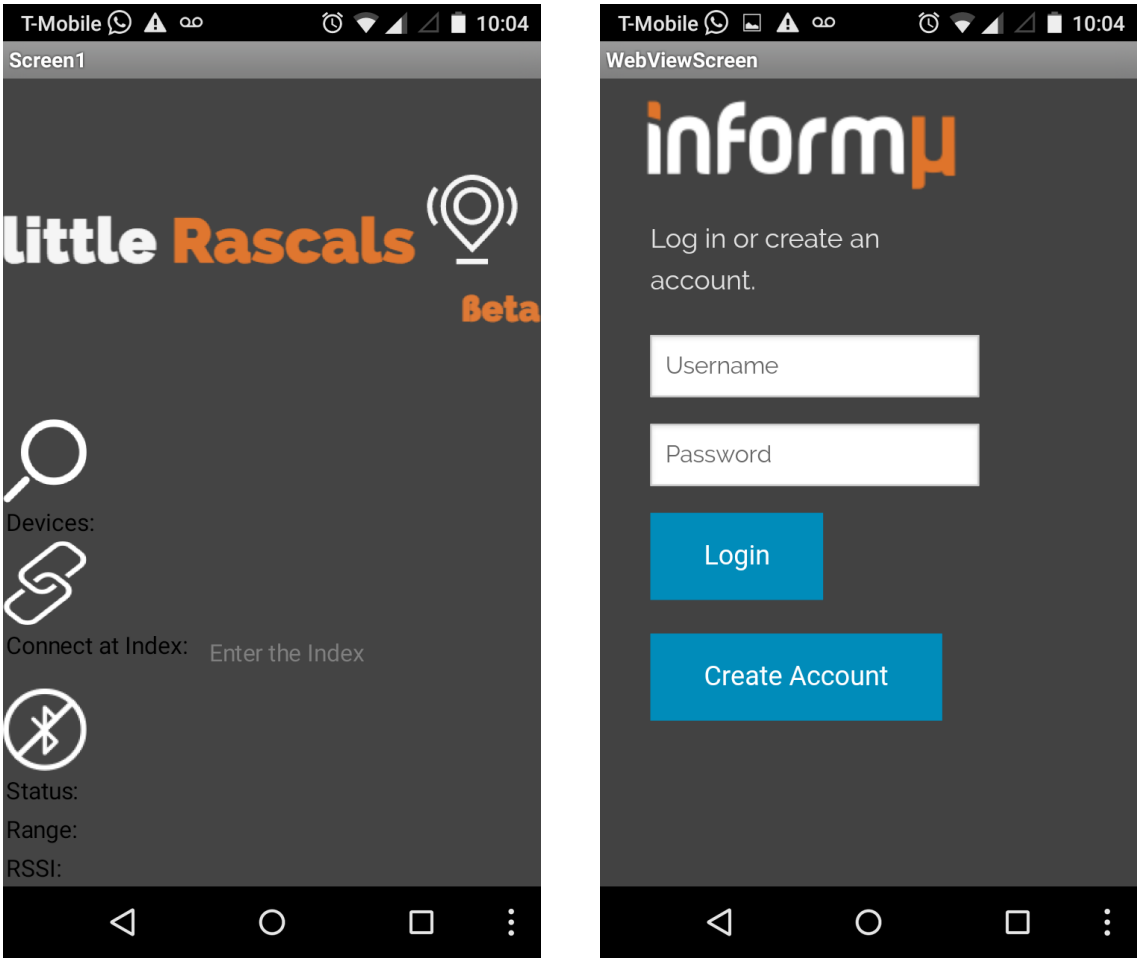


Figure 3 Main Screen and the Web View Screen of the Mobile App

We wrote a simple mobile Application on the MIT App Inventor platform. The application is for android platform.
We used the Bluetooth Low Energy API to scan for BLE devices and connect to the BLE Device that we desire. (That is our board)

Once, connected the App remains in the connected event block. As soon as the app is disconnected due to the distance of the mobile and device, the BLE Disconnect event is raised. This will notify the device as well and also open the Web Application in a Web View on the Mobile App directly.

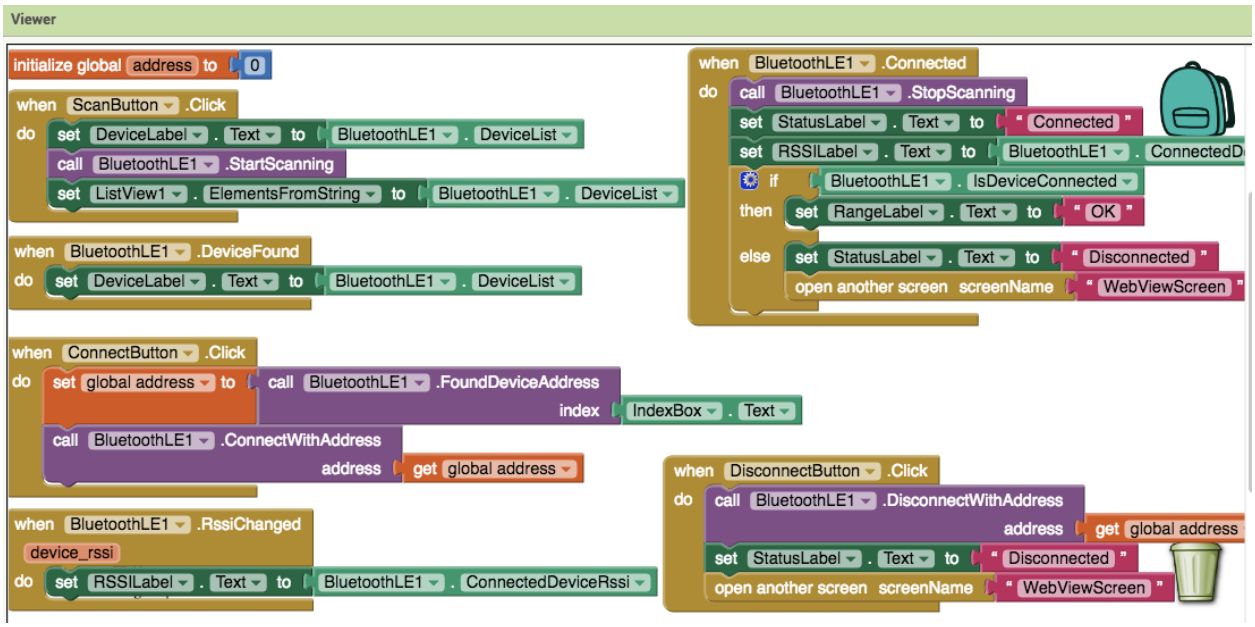


Figure 4 Functional Blocks for the Main Screen

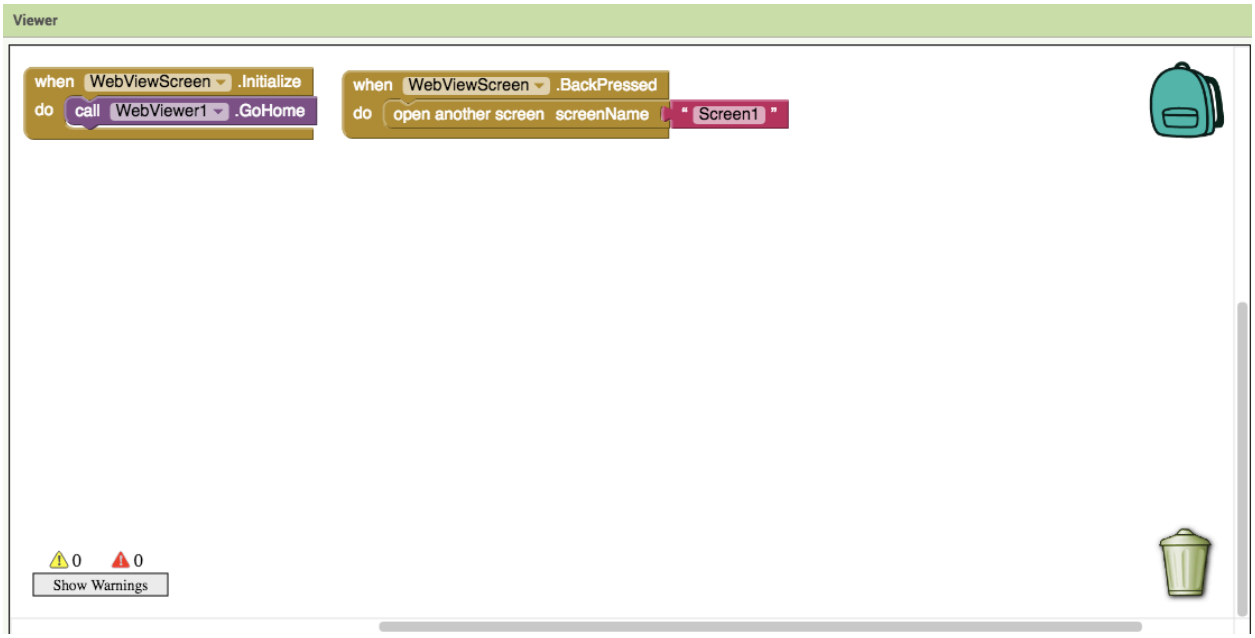


Figure 5 Functional Blocks for the Web View Screen

Planned Development Schedule:

Software/Hardware Module	**Planned Completion Date**	**When Completed**
Assembly	11/9	11/9
Continuity Check	11/10	11/10
Power Supply Test (**SIM 808**)	11/11	11/11
Power Supply Test (**LTC3331**)	11/11	11/11
Programming SAMB11	11/18	11/25
SIM 808 **GPS**	11/12	11/12
SIM 808 **GPRS**	11/15	11/25
SAMB11 **I2C** MMA 8452Q	11/23	11/29
SAMB11 **UART** SIM 808	11/21	12/03
Web Application	11/20	11/30
Mobile Application	11/30	12/08

Difficulties Encountered:

1. The major difficulty that we encountered was the PMIC (LTC 3331). We burnt a few IC's to learn how to use the PMIC correctly.
2. Having 3 radios on the board was a challenge but using existing documentation and Vivek's help for multiple rounds of review helped us bring up the board properly.
3. Mobile Application design using Visual studio was difficult as very less time was there. That is a whole new platform and language, we would love to have spent more time on it.
4. HTTP vs HTTPS problem due to the different protocols required and supported by the Google API and our SIM808 capability.
5. I2C on SAMB11 was a problem until we figured out that this is an issue with SAMB11's dye itself.
6. Watchdog timer issue on SAMB11 also was a speed bump.

Summary:

We have achieved almost all the functionality of the project as we set to achieve for this semester.

We have our board functioning properly:

1. All the radios work independently to perform their functions.
2. The power circuit works as designed, with energy harvesting. Although since we were not able to make the board fit in a shoe, we could not test how much does the device charge in one walk cycle.
3. The Web Application is complete and functions as we expected it.
4. We have built a Basic Mobile Application as well, which was not a part of the scope.
5. Our Firmware also is complete as per our Software Flow Diagrams, we could not optimize the energy to ultra-low levels due to hardware constraints like the Read function of the I2C doesn't work for the accelerometer which is required to put it to ultra-low power mode.

We could not achieve to put the device in a shoe, to be able to test it under the use case conditions properly.

If we could do a few things differently it would be starting the Schematics and Board Design earlier. That is one of the most important part of the project, if started earlier we might have been able to get the small errors with the board on time.
Also it would have given us more time to work on the mobile application. Better time management is one of the mains things we would do differently.

Conceptual Lessons Learned:

1. While examining the SIM808 schematic (Adafruit's) we learned how a Schottky Diode can be used as current limiter which eventually acts as a level shifter.
2. We learnt about High Speed Digital lines and how to route them properly without creating interference amongst each other. It is important to place these components with utmost care with data sheet specifications.
3. Dead copper can eventually become a source for EMI as eddy currents can generate from nearby power lines. Removing all dead copper is essential.
4. Energy harvesting is a new concept we learnt to implement. Using the PMIC (LTC 3331) we learnt about Buck and Buck-Boost convertors and how they work.

Appendix:

Programming MCU:

We used SWDIO and SWDCLK to program and debug the ATMEL SAM B11 module through the CORTEX DEBUG Header (10 pin).
We would connect to the CORTEX DEBUG Header using the 10 pin JTAG Cable.

We are programming our SAM B11 module from an external programmer i.e. Atmel Power Debugger.

We used the ATMEL Studio IDE for programming.

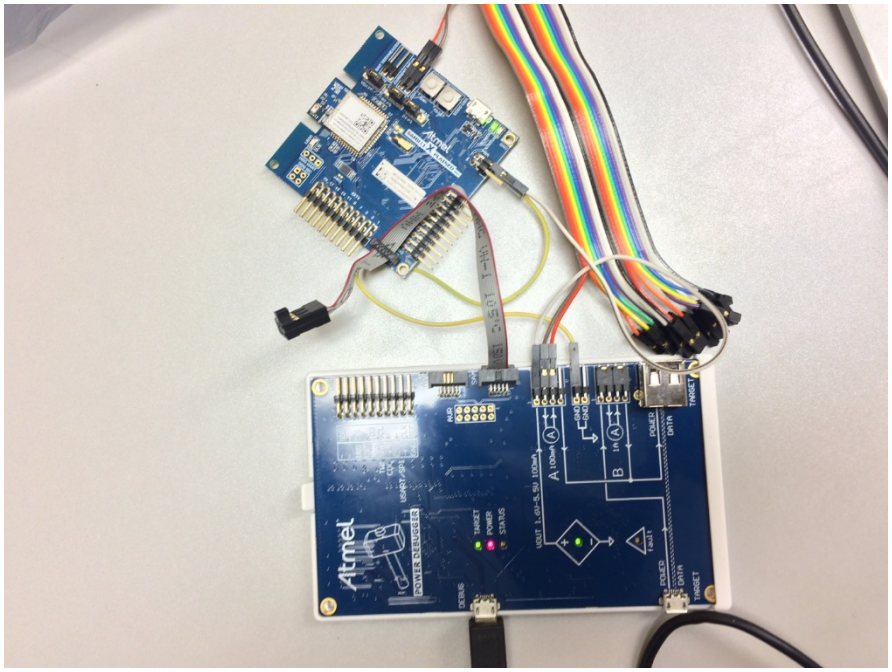


Figure 1 Programming SAM B11 with the External Debugger

Current Considerations for MCU (Worst possible scenario):

*We have not considered in this the intermediate energy available which would help us to reduce the current consumption further.

- Minimum Current Consumption - 1mA (Sleep)
- Maximum Current Consumption - 7mA (Active)

The average current would be ((1mA * 50 seconds) + (7mA*5s))/60 = 1.42mA
A battery of 400mAh should last for 282 hours.

Current Considerations for SIM 808
*We have not considered in this the intermediate energy available which would help us to reduce the current consumption further.

- Maximum Peak Current (Tx Burst) – 2A
- Typical Current Consumption – 1mA
- During GPRS Transmit – 200-300mA
- Current consumption during Normal mode – 1.4 mA
- Power down current consumption – 800 uA

We would be only switching on this module if the device is out of range. Once it is out of range, the module should update the new location every 1 minute.

module and damage it.

Notes: The module do not support for charging Ni-Cd or Ni-MH battery.

When battery is used, the total impedance between battery and VBAT pins should be less than 150mΩ.
The following figure shows the VBAT voltage drop at the maximum power transmit phase, and the test condition is as following:

VBAT=4.0V,
A VBAT bypass capacitor C_A=100μF tantalum capacitor (ESR=0.7Ω),
Another VBAT bypass capacitor C_B=1μF.

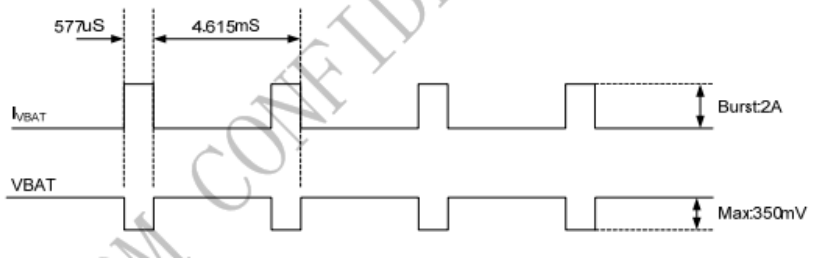


Figure 2 VBAT voltage drop during Tx Bursts

ON duty cycle for Tx Burts is 0.125% and OFF duty cycle is 0.875% therefore if we consider the base current to be the typical current in Normal Mode – **1.4mA.**

We get an Average current of **250mA**. This is the average current the Module would consume when it is ON. As mentioned earlier we are planning to update the location every minute.

Assuming that to update the location it takes 5 seconds,
Average Current is ((250mA * 5) + (1mA * 55))/60 = **21.75mA**.
This is a lot of current, but we have a separate bigger power supply for this (1200-2000mAh battery) with USB power. So the battery life for 1200mAh battery would be **55 hours** approx.

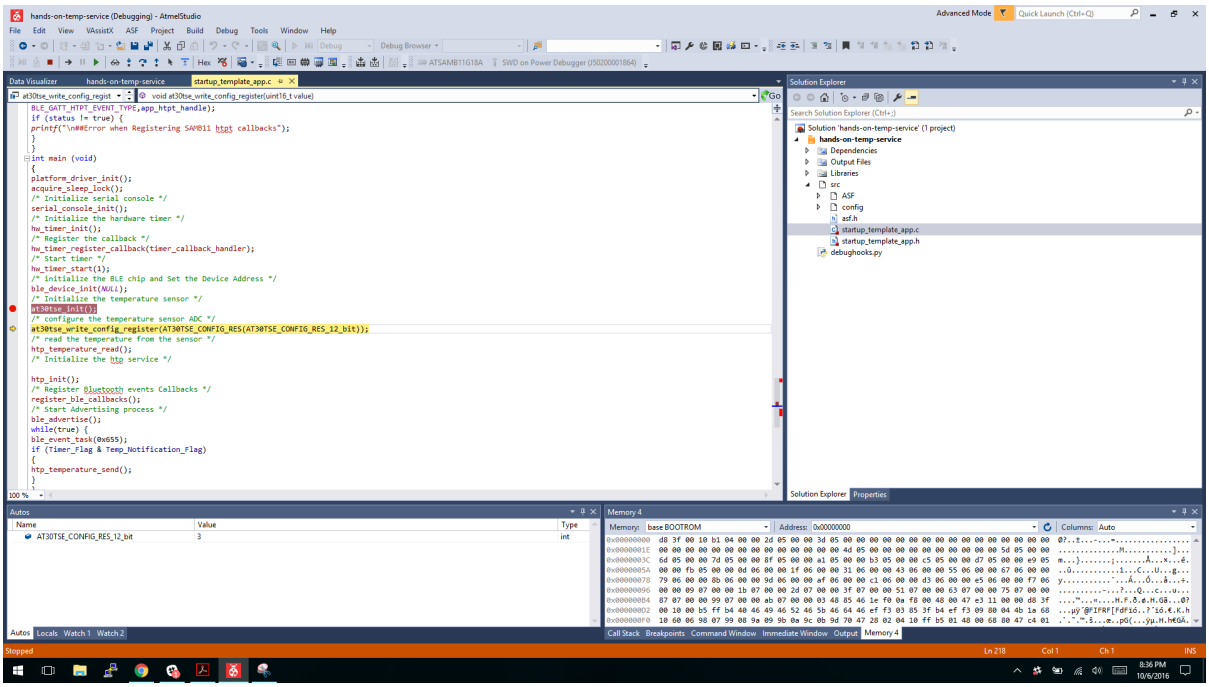


Figure 3 Screenshot of Setting break-points using the ATMEL Power Debugger

We verified whether the power debugger could be used to have control over the execution of the code and we were able to set breakpoints and debug the code as shown in the figure above.

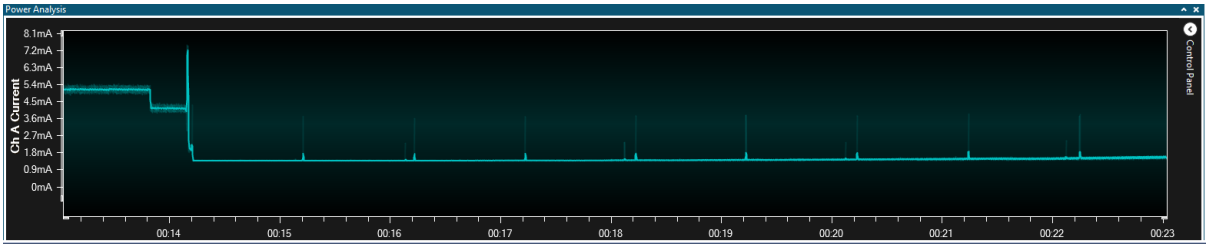


Figure 4 On Board Current Measurement using ATMEL Data Visualizer

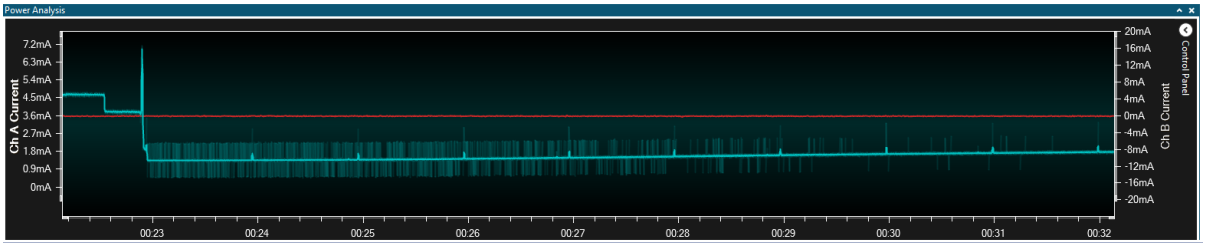


Figure 5 Current Measurement using ATMEL Power Debugger (ext)

Note: See the difference in Figure 2 and 3. We are getting noise on the data visualizer which we believe is due to the loose jumper wire connections, it also could be due to the long length of the wires as compared to the traces.

We have a header that will connect to the Piezo element. This is the energy harvester and should be used to charge the input Capacitor (25V). Our LTC3331 has a UVLO rising threshold of 12V i.e. it requires the input capacitor to charge to 12V before it generates an output (Vout) voltage.
Since charging it to 12V initially would take some time with the piezo before the product can be used. Therefore, we would connect a AC signal generator to charge the CAP through the Piezo terminals.

1mA active current [Ref: SAMB11 datasheet]

8.4 Flash Memory

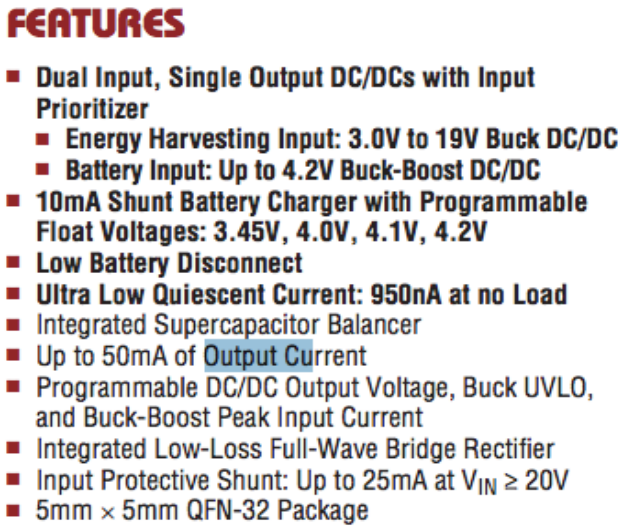
SAMB11 has 256kB of Flash memory, stacked on top of the MCU+BLE System on Chip. It is accessed through the SPI Flash controller and uses the 26MHz clock.

Flash memory features are:

- 256-bytes per programmable page
- Uniform 4kB Sectors, 32kB & 64kB Blocks
- Sector Erase (4K-byte)
- Block Erase (32K or 64K-byte)
- Page program up to 256 bytes <1ms
- More than 100,000 erase/write cycles and more than 20-year data retention
- 2.3V to 3.6V supply range
- 1mA active current, <1µA Power-down

Figure 6 ATMEL SAMB11 Flash Memory Specifications

50mA of output current [Ref: LTC3331 datasheet]



FEATURES

- **Dual Input, Single Output DC/DCs with Input Prioritizer**
  - **Energy Harvesting Input: 3.0V to 19V Buck DC/DC**
  - **Battery Input: Up to 4.2V Buck-Boost DC/DC**
- **10mA Shunt Battery Charger with Programmable Float Voltages: 3.45V, 4.0V, 4.1V, 4.2V**
- **Low Battery Disconnect**
- **Ultra Low Quiescent Current: 950nA at no Load**
- Integrated Supercapacitor Balancer
- Up to 50mA of Output Current
- Programmable DC/DC Output Voltage, Buck UVLO, and Buck-Boost Peak Input Current
- Integrated Low-Loss Full-Wave Bridge Rectifier
- Input Protective Shunt: Up to 25mA at $V_{IN} \geq 20V$
- 5mm × 5mm QFN-32 Package

Figure 7 LTC3331 Specifications

We have provided a jumper to remove the Power Circuit (LTC3331) in case it malfunctions.
We also have provided external header for both the input side of our MCU battery and the output side of the LTC Regulator(PMIC).

We can connect the external regulated power supply to any either of these terminals to power up our MCU.

We would use ATMEL power debugger to power the digital portion of the board. (Min. requirement of 50mA of current, which power debugger can provide.)

Power Consumptions:

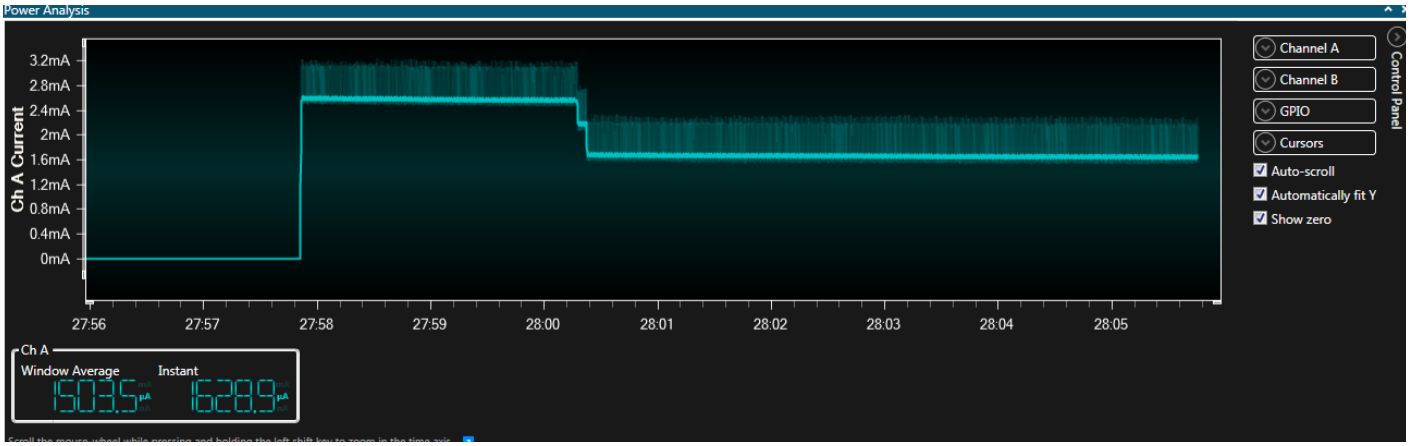


Figure 1 After SIM is turned off and MCU in sleep it waits for BLE/Accelerometer interrupt.

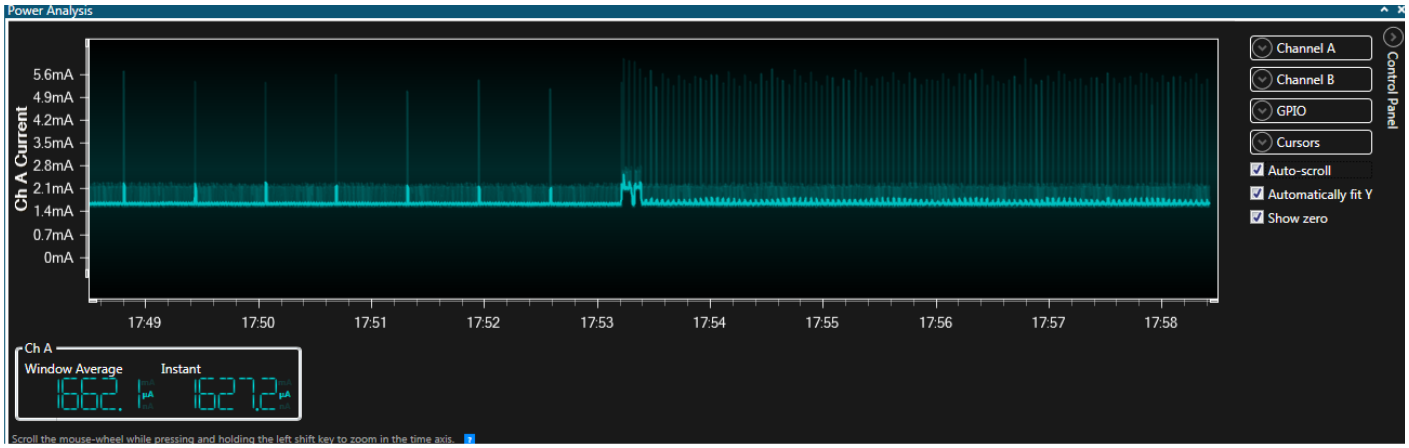


Figure 2 BLE Advertising to connecting phase. The Spike visible is where the connection begins.

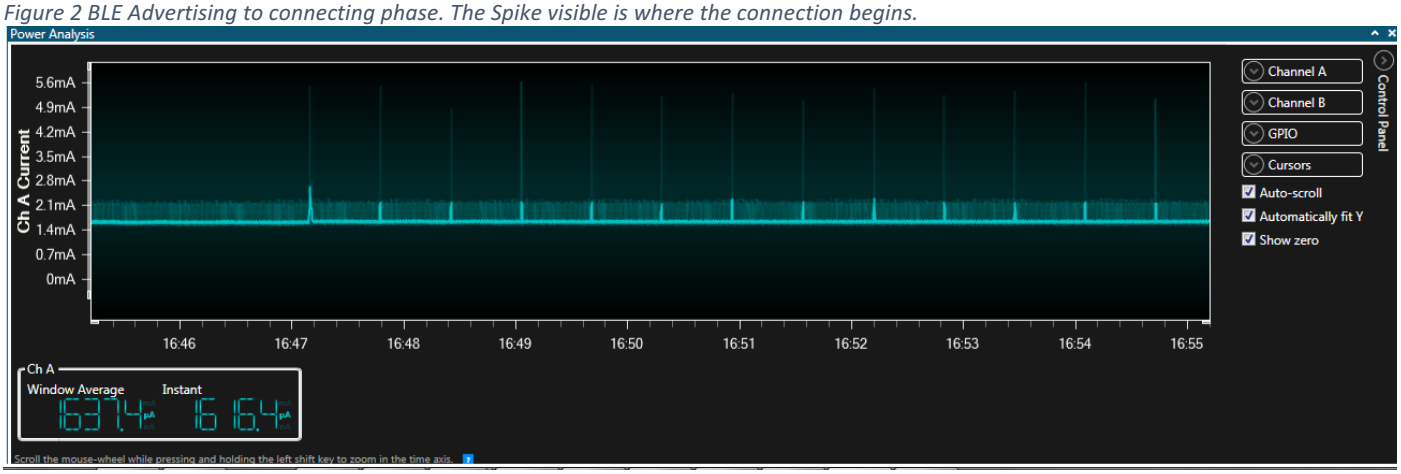


Figure 3 BLE Advertising

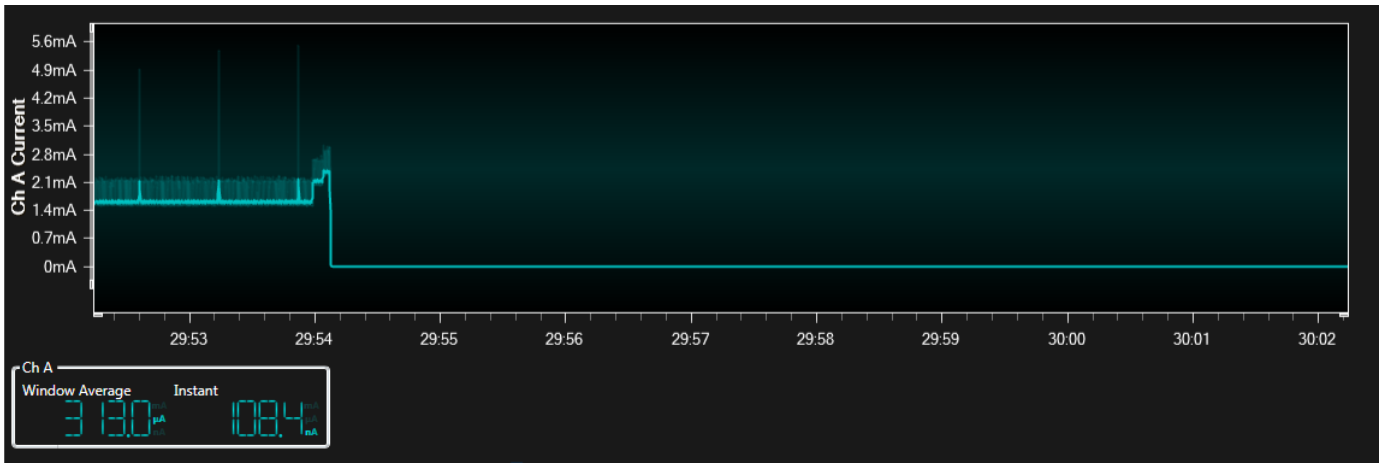


Figure 4 SAMB11 in low power mode and communicating with SIM808 UART to send commands to Post location data.

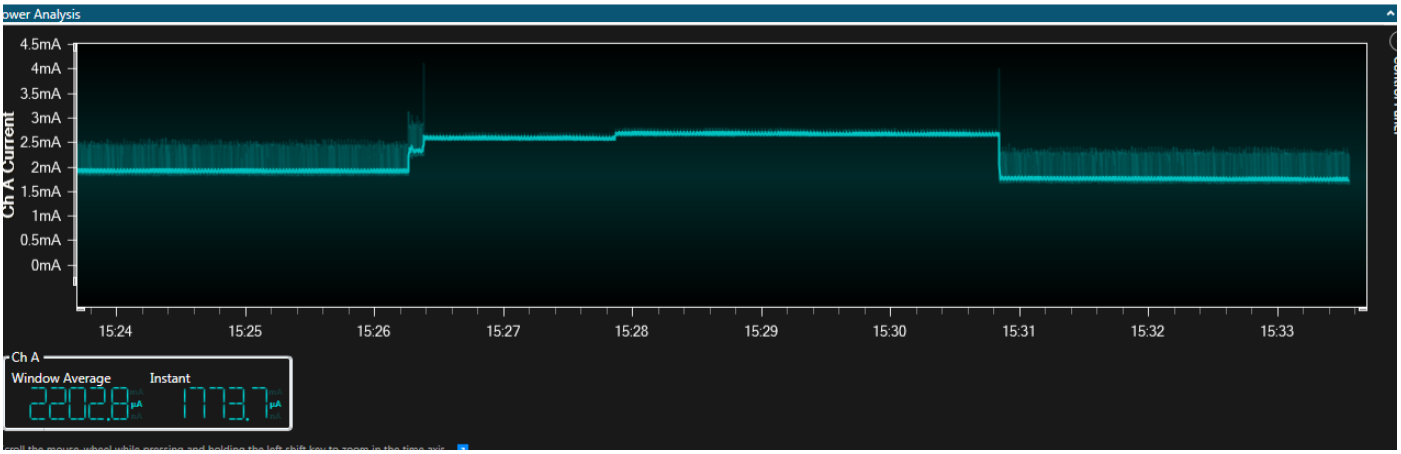


Figure 5 Initialization Sequence

SAM B11's **highest Power** Consumption is **2.2mA** during the initialization sequence.

Thereafter it consumes **1.6mA** on **average on advertising** and **1.8mA** during **connecting phases**.

The SAM B11 power drops to the order of **313uA** while BLE is off and UART commands are sent to activate SIM 808.

Time	Battery Level
0.00	56
3.00	56
6.00	56
8.00	56
11.00	55
14.00	55
17.00	56
20.00	56
22.00	56
25.00	56
28.00	55
31.00	55
33.00	55
36.00	55
39.00	55
41.00	55
44.00	55
47.00	55
49.00	55
52.00	55
55.00	55
58.00	55
60.00	55

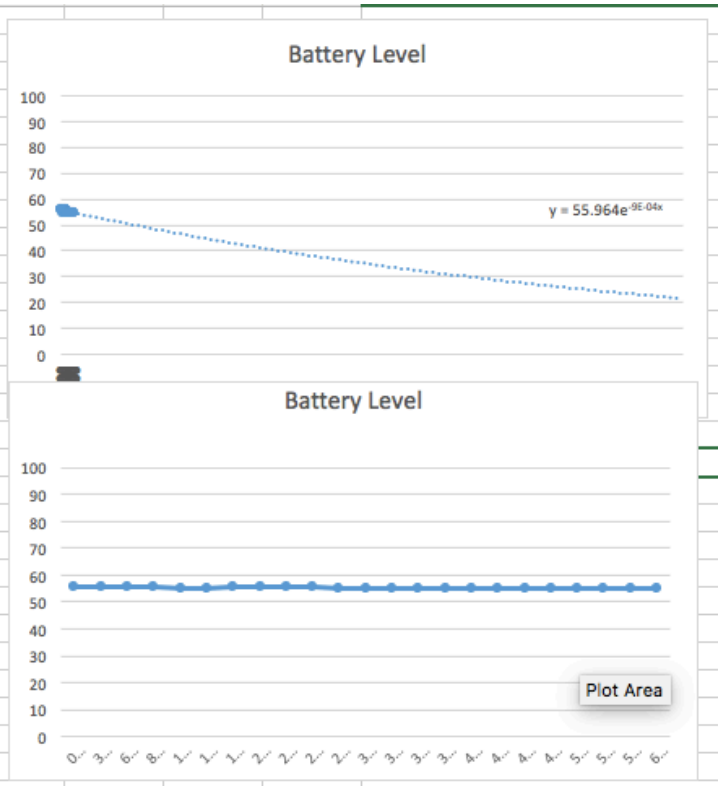


Figure 9 SIM 808 Power Calculations and Extrapolation to find the battery low point.

This is the equation obtained after extrapolating the battery decay of SIM 808 after a few data points.

$Y = 55.964e^{-9E-04x}$

Where Y is the %of Battery and x is the time elapsed in minutes.

This equation tells us that after 1000 minutes (16.67 hours) have passed the battery will come down to 22.75%

This means the device can currently supply up to **18 hours** of non-stop tracking once it goes out of range. This is using exponential decay which is worst possible case. A good battery never will have exponential decay, it is more of linear discharge from 90% to 40%.

We hope the 1200mAh battery should last at least **80-100 hours**.