University of Colorado
Boulder

Project Report
On
VaXinator - A Payload monitor

Spring 2022

ECEN 5833 Low Power Embedded Design Techniques

Submitted By:

Team 4: Nexus

Rishab Shah, Mukta Darekar

Guided By:

Prof. Randall Spalding

Table of Contents

Project Definition

Goals

Our Project goal is to track the complete status of the vaccines which are transported from the manufacturing facility to the distribution facility to ensure its potency and efficacy. This project goal is a subset of a generic in-transit Asset tracking system.

Motivation - Problem we are trying to solve

Vaccines require to be in a particular temperature range for it to be administered to the living-being. Exposure to inappropriate temperatures can reduce vaccine potency and efficacy, increasing the risk that living-beings are not provided with maximum protection against preventable diseases. Mishandling of the vaccines can lead to breakage and eventually shortage especially during tough times, it becomes very important to not let even one vaccine go to waste. It also becomes important to track vaccines so that unauthorized administration of vaccines does not occur.We have employed Low power design techniques to make this product a success.

Project Overview

VaXinator is used to monitor the status of vaccines during shipping. Our product is mounted on the vaccine container or any container in general as per user application needs.

It analyzes the Temperature and Humidity of the storage facility at particular intervals to understand if the vaccines got transported in the prescribed temperature ranges. Also, an IMU sensor is used to determine if the handling of the vaccines was done with extra care and the container didn't face any shocks during transportation.

In case of any shock or abnormal condition detection, those values will be stored in data storage along with timestamp. Additionally, the GPS is used to collect the timestamp and location information to help in the error logging process.

On arrival of the container at the destination, it can be verified for its validity by collecting error logs stored in VaXinator using Bluetooth technology. If a smartphone app shows status as in Good Condition then the container's content will be passed on. In case if the app shows a status as in Bad condition then the container will be sent for further scrutiny.

VaXinator can be powered up using a battery and USB for charging the battery for having portable usage during in-transit.

Features

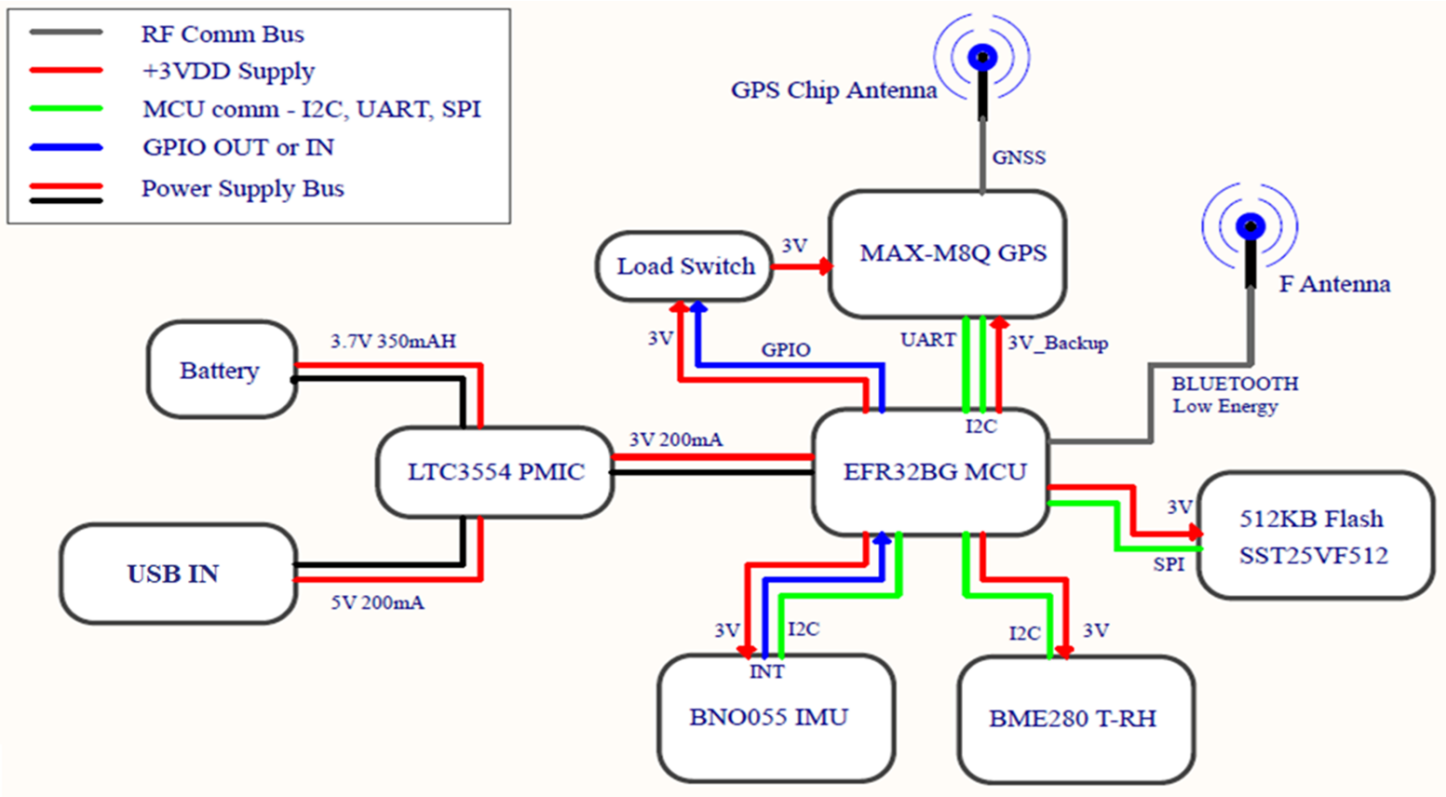
1. Temperature & Humidity Monitoring: Track the health status of the asset.
2. Vibration Detection: To detect if any mishandling of the package is done
3. Location and time based monitoring - to keep track of all abnormal event occurrence with their location as well as time
4. Error Logging - Flash memory of 512KB to store the information OF ERRORS
5. Transferring error log through bluetooth to provide an ease to access the data.
6. Battery and USB provision - To charge the product in-transit as well as ensure the device can operate over a duration of 24 hours without battery.

Specifications

Below are out project specifications for VaXinator:

1. Physical Dimensions
    - Board Size -> 2 x 2.5 inch
2. Power Requirements
    - Operating Voltage - 3.0 V
    - Current - targeting max. 7 mA in no activity and advertising mode
3. Operating Conditions
    - Temperature Range -> 2 to 7 Degree Celsius
    - Humidity -> 55% or lower at +2-8°C
    - Bluetooth Range -> 2 to 7 feet
4. Capacity
    - Data storage maximum capacity - 24Hr - every 1 minute error logging - provision of max 512KB storage
    - Battery life - at typical 5mA per Hr total life of 24Hr with 120mAH 3.7V rechargeable battery
    - Warranty - depends on the most vulnerable part i.e. the battery chosen.

System Block Diagram



Hardware Requirements

Controller selection

We decided to use EFR32BG as our application required bluetooth compatibility, capability to interface different sensors working on I2C, SPI, ADC. EFR32 also provides good Energy power management through various EM1 to EM3 modes.

Sensor (or Actuator) selection

For Temperature and Humidity detection we used BME280 and for vibration detection we used BNO055 sensors which consume less energy and achieve Low power goal. GPS - MAX M8Q is used along with load power management. Error log information is stored in the FLASH memory.

Power Supply options:

As this product is portable and it fits on shipping containers, it will be hard to provide continuous supply from the power grid or any major power source. Our product is thus battery dependent during transit. The product is recharged using a USB power supply.

Energy Consumption

As per our calculations (Please refer to the xlsx sheet in PowerEnergyCalculations_VaXinator.xlsx), below are the calculated weighted power values per use case over a 24 hour operation cycle.

State1:	Sleep mode	1.34 mW	99.99% (Time during Transit)
State2:	Run mode	139.91 mW	0.1% (Time during Transit)
State3:	Bluetooth mode	532.955 mW	
State4:	GPS + Run	469.9137 mW	0.001% (Time during Transit)

Energy Model

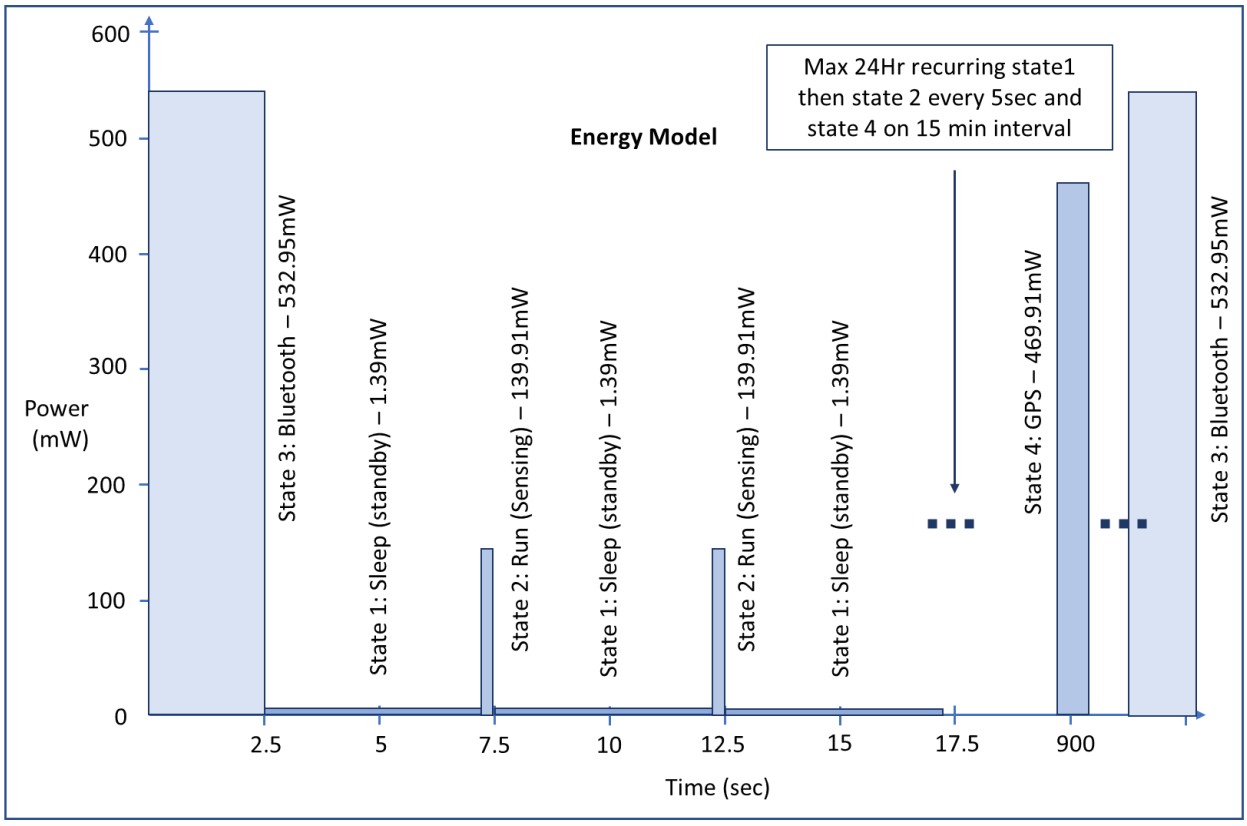


Figure 4: Energy Model of VaXinator

Power Calculations Summary

In the VoltageLevels sheet (Please refer to the xlsx sheet in folder PowerEnergyCalculations_VaXinator.xlsx), our total requirement is 172.9428 mAH. Such an amount of mAH would not be provided by a capacitor and the size constraints are also not conducive in choosing a capacitor. Hence, we have decided to go with a battery - 2750 which has 0.5 cm height and a capacity of 350 mAH. This battery would be suitable for more applications. Even though the charge would be about an hour but it provides a working life of more than 24 hours.

Selection of Energy Storage

The energy storage device selected is a LiPo battery with 350 mAH capacity (Deduced from above).

a. C Rate: 1 C
b. Peak Discharge rate: 525 mA at 1.5C
c. Lowest Nominal Voltage : 3.0 V
d. Storage Voltage in use: 3.7 - 3.85 V
e. Maximum Voltage: 4.2 V
f. Minimum Voltage/Battery Cutoff Voltage : 3 V
g. tolerance analysis: Min to max Nominal capacity = 332 mAH to 350 mAH

Selection of Charging methods

The product would be in a confined space for most of its operation. The major source of energy which would be feasible is USB supported charging technology. Hence, we decided to use a combination of USB and battery with sufficient capacity to last at least 24 hours at minimum for a run based on the calculations.

Selection of PMIC

We decided to go ahead with LTC3554 for the features mentioned in PowerEnergyCalculations_VaXinator.xlsx-> Energy Calculation.

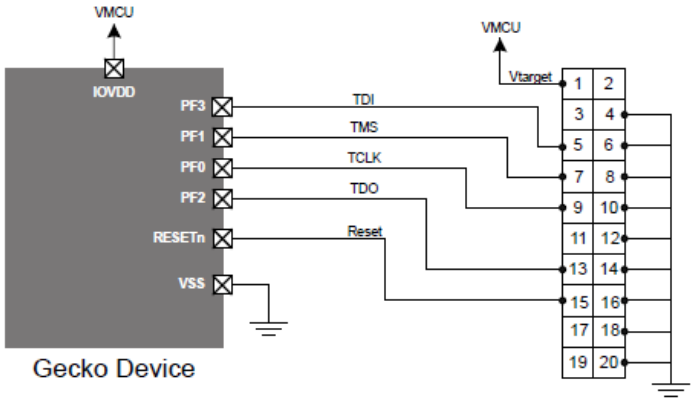
As V_{BUS} Undervoltage Lockout = 3.5V, VBUS to BAT Differential Undervoltage Lockout = 0mV And our battery has a cutoff voltage = 3V (from datasheet).

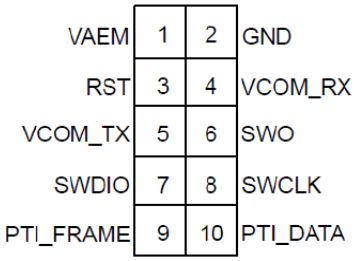
So we can easily program our PMU to its minimum cut-off voltage of 3.0 as it will work well with battery limitations.

Important Considerations

Compile and Download Process

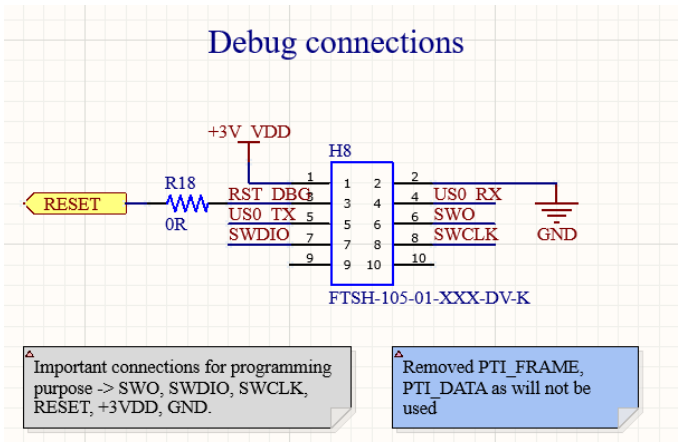
We added a mini debug connector to the EFR32BG13. This is a hardware connection which helped us to connect to the MCU of the prototype board using SWD through the mini debug connector as shown in below figure.
We did not make use of PTI frames. Hence, were not used in the schematic.
https://www.silabs.com/documents/public/application-notes/an958-mcu-stk-wstk-guide.pdf
https://community.silabs.com/s/article/kba-bt-1210-connect-custom-board-to-simplicity-studio-using-jtag-swd-x?language=en_US

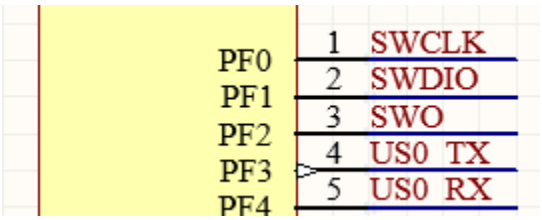




ARM 20 Pin Header

Connecting the EFM32 and EFR32 Wireless Gecko Series 1 Device to an ARM 20-pin Debug Header

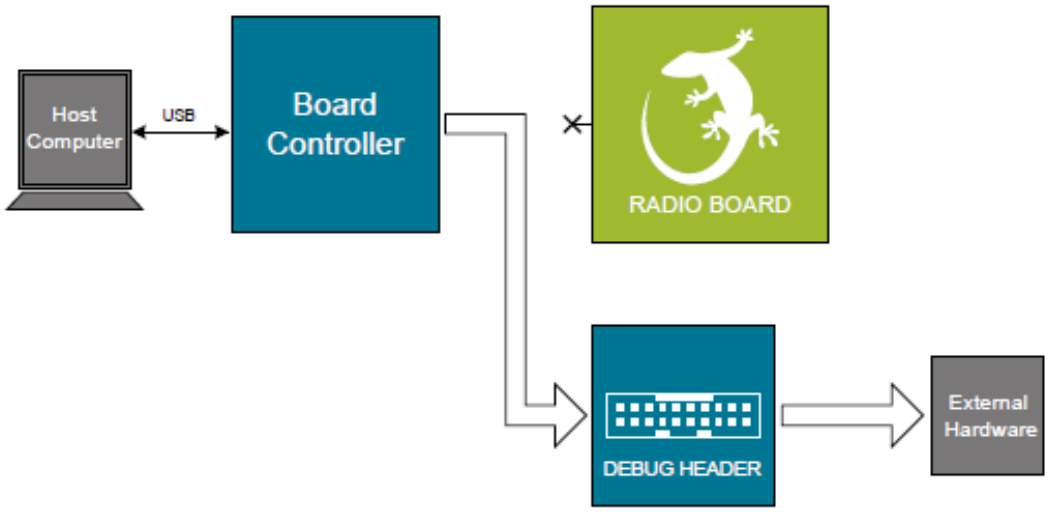




IDE: Simplicity Studio
We developed our code on simplicity studio and programmed it on the Evaluation board. Once the software was working with an evaluation board we made changes in the settings under the

Debug section of IDE in Device configuration. The appropriate chip number was entered and in the adapter settings of MCU it was changed to OUT from MCU and utilized the mini debug connector to program the prototype MCU.
https://www.silabs.com/documents/public/user-guides/ug279-brd4104a-user-guide.pdf

Debug OUT: In this mode, the on-board debugger can be used to debug a supported Silicon Labs device mounted on a custom board as shown in below image.

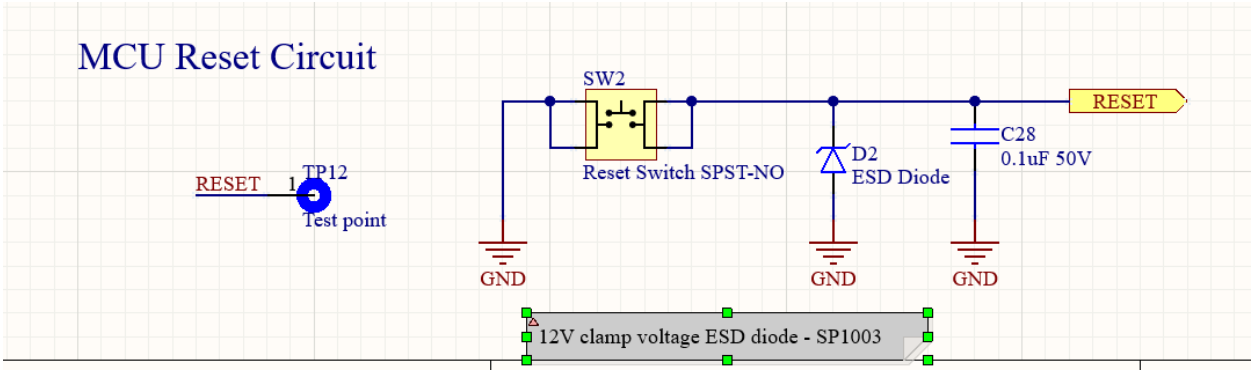


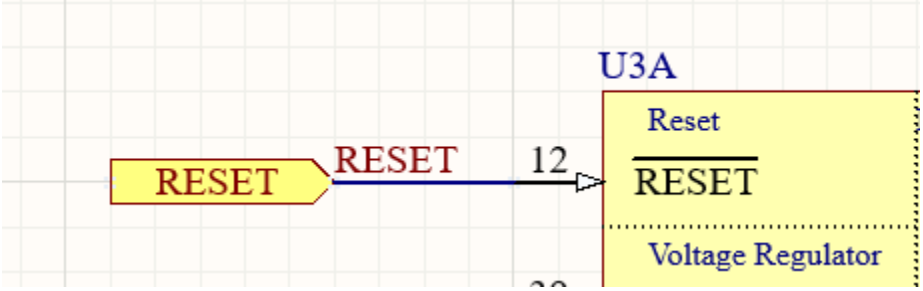
Test Point Signals

- Test points for +3V_OUT, +3V_VDD and GND.
- GPIO signal for profiling. To debug parts of the time sensitive code where it is not convenient to use a debugger or print logs.
- I2C -> SCL and SDA lines to capture logic analyzer traces. (BME280 and BNO-055)
- Flash memory SPI Lines -> MOSI, MISO,CS and SCK
- LEUART -> TX and RX (for GPS)
- Load switch Out
- +5V USB, +VBATT

RESET circuit

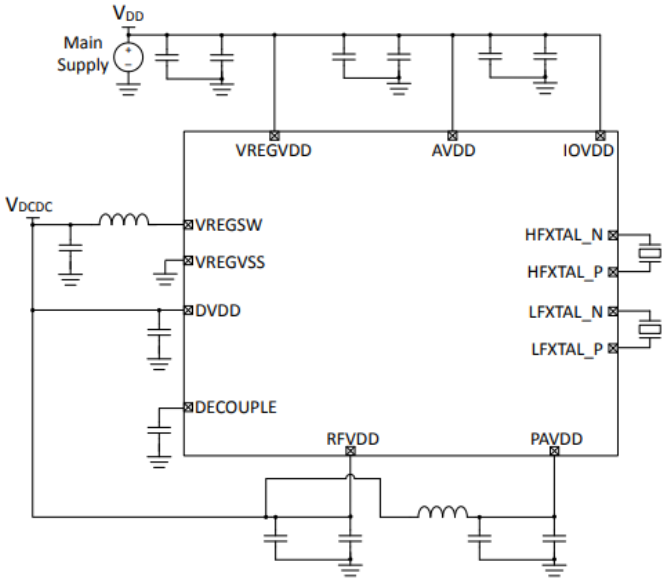
RESET by I/O option was utilized. A power on reset circuit ensures the system power supply stabilizes at the correct levels, the clocks of the processors settle accurately, and that the loading of the internal registers is complete before the device actually starts working or gets powered up. A typical diagram for RESET circuit ias below:





Power and Crystal connections of Controller

https://www.silabs.com/documents/public/data-sheets/efr32bg13-datasheet.pdf



EFR32BG13 Typical Application Circuit: Configuration with DC-DC converter (PAVDD from VDCDC)

Clock generation:

System utilized following different clocks

1) Sleep / Idle state: EM3 mode - LFRCO - 1KHz
2) Sensor working: EM1 mode - LFXO - 32.768 KHz
3) Bluetooth / RF communication: HFXO - 38 MHz - for antenna

Reference:
https://www.silabs.com/documents/public/application-notes/an0016.1-efm32-series-1-oscillator-design-considerations.pdf

Below tables cover all clock types and different crystals available for high frequency of 38MHz and low frequency of 32.768KHz.

Clocks in EFR32BG

Oscillator	Frequency	Optional?	External components	Description
HFXO	38 MHz - 40 MHz	No	Crystal	High accuracy, low jitter high frequency crystal oscillator. Tunable crystal loading capacitors are fully integrated. The HFXO is required for all types of RF communication to be active.
HFRCO	1 MHz - 38 MHz	No	-	Medium accuracy RC oscillator, typically used for timing during startup of the HFXO and as a clock source as long as no RF communication is active.
AUXHFRCO	1 MHz - 38 MHz	No	-	Medium accuracy RC oscillator, typically used as alternative clock source for Analog to Digital Converter or Debug Trace.
PLFRCO	32.768 kHz	No	-	Self-Calibrating RC oscillator, typically used for RTCC and BLE sleep timing.
LFRCO	32768 Hz	No	-	Medium accuracy frequency reference typically used for medium accuracy RTCC timing.
LFXO	32768 Hz	Yes	Crystal	High accuracy frequency reference typically used for high accuracy RTCC timing. Tunable crystal loading capacitors are fully integrated.
ULFRCO	1000 Hz	No	-	Ultra low frequency oscillator typically used for the watchdog timer.

32768Hz Crystals: (selected FC-135 FC-13532.7680KA-AG3)

Mfg	Part	ESR (kΩ)	C_0 (typ) (pF)	Temp (°C)	Tolerance (ppm)	C_L (pF)	Footprint (mm)
Abracon Corporation	ABS07-32.768KHZ-7	70	1.2	-40 to +85	± 20	7	3.2 x 1.5
Epson	FC-135	65	1.0	-40 to +85	± 20	9	3.2 x 1.5
KDS	DST210AC	70	1.3	-40 to +85	± 20	7.0	2.0 x 1.2
KDS	DST310S	70	1.3	-40 to +85	± 20	12.5	3.2 x 1.5
Micro Crystal	CM8V-T1A	55	1.1	-40 to +85	± 20	13	2.0 x 1.2

38.4MHz Crystals: (Selected CX2016DB38400F0FSRC1)

Mfg	Part	ESR (Ω)	C_0 (max) (pF)	Temp (°C)	Temp Tolerance (ppm)	Mfg Tolerance (ppm)	C_L (pF)	Footprint (mm)
KDS	DSX211SH 1ZZHAE38400AB0A	50	2	-40 to +85	± 20	± 10	10	1.6 x 2.0
KDS	DSX211SH 1ZZHAE38400AB0B	50	2	-40 to +105	± 20	± 10	10	1.6 x 2.0
Kyocera	CX2016DB38400F0FSRC1	40	1	-40 to +125	± 40	± 10	10	1.6 x 2.0
Kyocera	CX2520DB38400F0FSRC2	50	1.2	-40 to +125	± 40	± 10	10	2.0 x 2.5
Kyocera	CX2520DB38400F0FSRC3	40	1.2	-40 to +125	± 40	± 10	10	2.0 x 2.5
NDK	EXS00A-CS08361	40	1.26	-40 to +125	± 32	± 10	10	2.0 x 2.5
NDK	NX2016SA-38.4MHz-EXS00A-CS08973	60	0.61	-30 to +85	± 12	+-10	10	1.6 x 2.0
TaiSaw	TZ0909E	50	3	-40 to +125	± 30	± 10	10	2.0 x 2.5
TaiSaw	TZ2205E	40	3	-40 to +125	± 30	± 10	10	1.6 x 2.0

Alternate energy source apart from harvester

USB is our main source of charging the battery and provides a life-span of 24 hours..

Jump Start method to charge the energy storage element

Our battery takes about 1.5 hour to charge from 2.8V to 3.6V. Hence, we will be using, as per suggestion, a lab CC power supply to regulate the current and charge the battery at 4.2 volts as maximum charging voltage is 4.2 Volts. CC power supply will be utilized to ensure that the battery is charged cautiously.

Maximum Charging Current of PMU circuit

The maximum current chagrin current allowed by the PMU circuitry is 500mA. With usage of a 1.8K resistor, A charge current of 400 mA can be achieved.

Maximum charging current allowed by the energy storage

350 mA is the maximum charging current of the energy storage element i.e. the battery selected for our project. The maximum current of the CC power supply is set to 349 mA. Positive of PS to Positive of Battery and Negative of PS to negative of battery. When charging is almost complete, the PS switches from CC to CV.

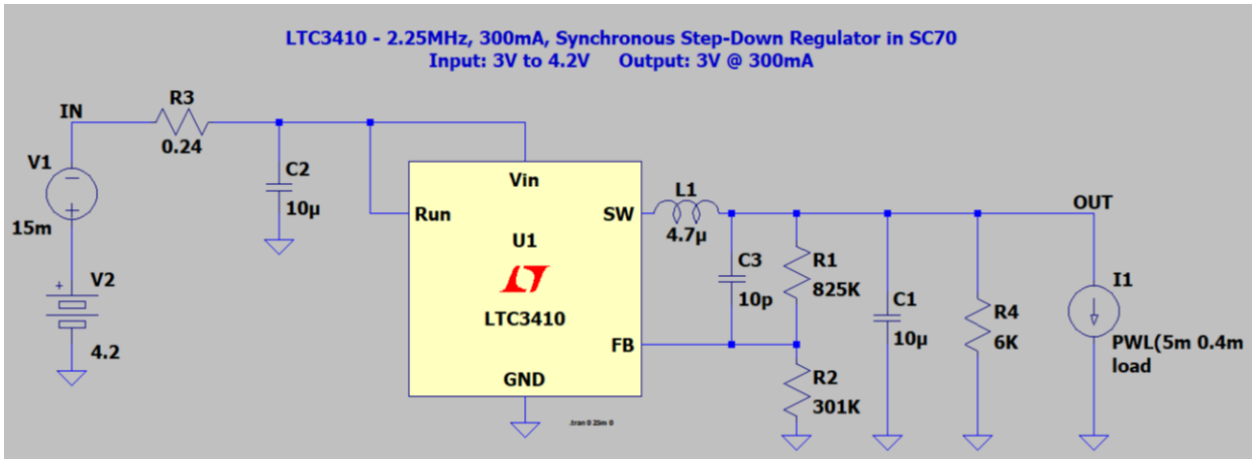
Energy Current requirement to program the FLASH of the MCU

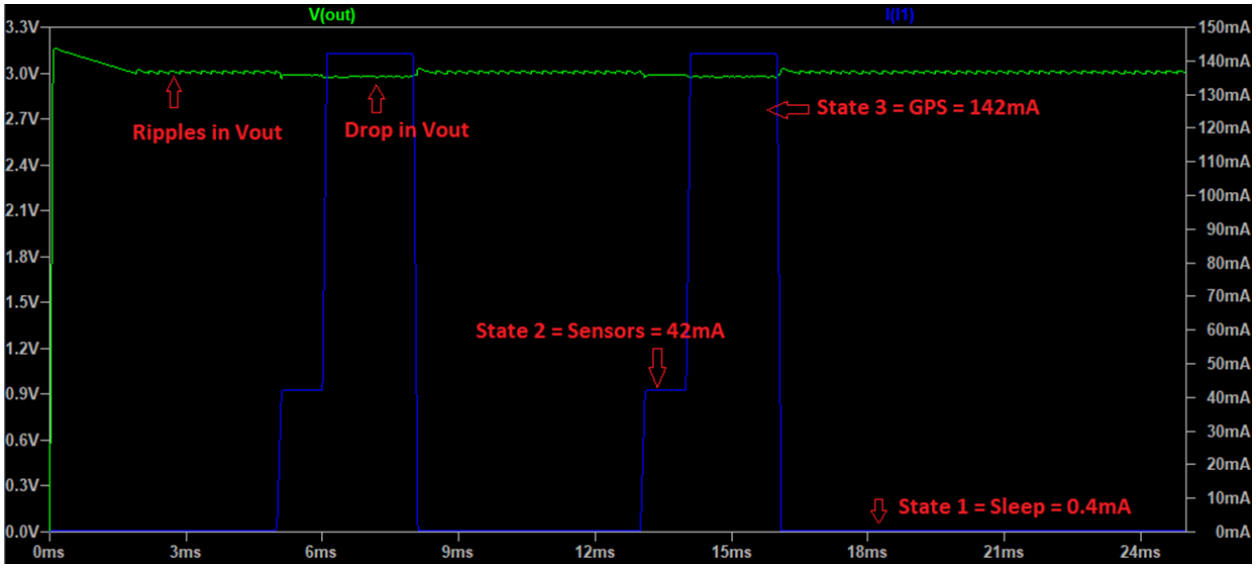
Programming of MCU would require a max of **3.5mA** refer below table and energy Storage element (Battery) current supplying capacity is nominal (0.2*C) **70 mA** and maximum is **350mA.** Typical current that can be supplied by the PMU circuit is **200 mA**.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Flash erase cycles before failure	EC_{FLASH}		10000	—	—	cycles
Flash data retention	RET_{FLASH}	T ≤ 85 °C	10	—	—	years
		T ≤ 125 °C	10	—	—	years
Word (32-bit) programming time	t_{W_PROG}	Burst write, 128 words, average time per word	20	26.3	30	µs
		Single word	62	68.9	80	µs
Page erase time[2]	t_{PERASE}		20	29.5	40	ms
Mass erase time[3]	t_{MERASE}		20	30	40	ms
Device erase time[4 5]	t_{DERASE}	T ≤ 85 °C	—	56.2	70	ms
		T ≤ 125 °C	—	56.2	75	ms
Erase current[6]	I_{ERASE}	Page Erase	—	—	2.0	mA
Write current[6]	I_{WRITE}		—	—	3.5	mA
Supply voltage during flash erase and write	V_{FLASH}		1.62	—	3.6	V

Bulk Capacitance Analysis

We used the LTC 3410 Buck regulator to demonstrate the buck inside LTC 3554 using LTspice and to anticipate its performance. Max drop in voltage at 3V on state change is 2.85V – All components work perfectly at this voltage as minimum voltage rating for all chips in our project is around 2.7V. So having below circuit design and at 3V vout will be perfect for our application as it will have max utilization of battery till its cutoff voltage of 3V.





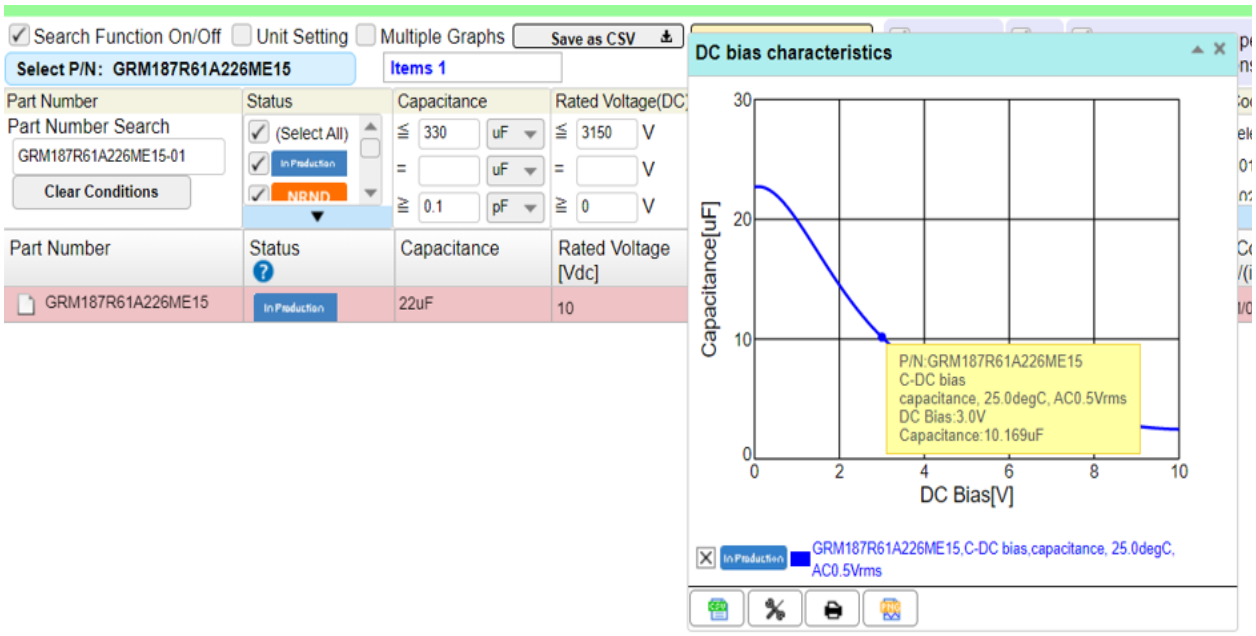
Bulk capacitance selected previously was 10uF 10V as per given as recommandation in LTC3554 PMIC datasheet. But after the simulation of the PMIC circuit and simulating practical performance of PMIC, the actual bulk capacitance required for our power supply is 22uF.

Selected Component as per analysis

Component: CAP CER 22UF 10V X5R 0603
Part Number: GRM187R61A226ME15-01
https://www.digikey.com/en/products/detail/murata-electronics/GRM187R61A226ME15D/5877406

After doing analysis using MURATA's online tool we found that the 22uF capacitor component as mentioned above gives an effective capacitance of 10.17uF which will satisfy PMIC datasheet recommendation under DC bias condition at 3V from below image

The effective capacitance at voltage DC bias of our system:



ESD Protection

As ESD protection is necessary wherever there are chances of human touch causing ESD spark, we have placed ESD diodes across USB input, Reset push button and PMIC push button with required clamping voltage.

Schematic and Layout Considerations

Following are some of the schematic guidelines:

1. A switch between PMIC output and rest of the circuit to power on.
2. 0 Ohm resistor provision wherever required for provision of section isolation
3. Test points for voltages an
4. Adding proper netlabels and keeping consistency in naming convention as well as flow of the power from Left to right
5. Ensured that the pages are divided by module. Hence, design is scalable.

Layout Guidelines referred:

1. Section wise segregation of layout - Power section, MCU, RF, Sensors
2. Common Ground plane as bottom layer
3. Short in size and less number of cross unders and Ground stitching across cross unders
4. Via stitching of GND plane near RF sections
5. Mounting holes - remove copper for holes in path of antenna direction
6. Nothing should come in the path of F-antenna or chip antenna
7. Solder mask under antenna sections

8. 20mil or more trace for power nets or use of copper pours
9. F antenna and GPS chip antenna should point in opposite directions

Key Components

Finalized Component list

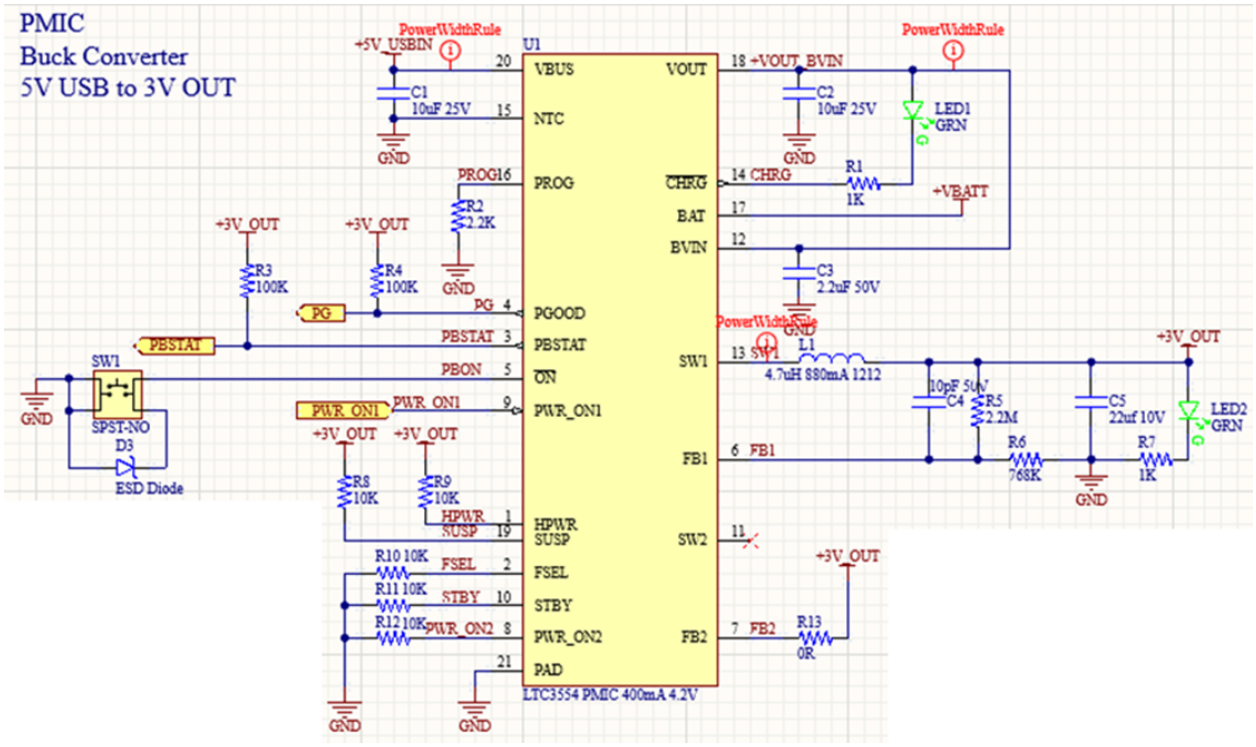
Component	Value	Reason for selection	Datasheet
Micrccontroller	EFR32BG	Supports Bluetooth and different t energy modes especially Low power mode of operation through EM0, EM1, EM2 and EM3 mode where different peripherals are active in different modes.	https://www.silabs.com/documents/public/data-sheets/efr32bg13-datasheet.pdf
T-RH Sensor	BME280	**Purpose:** Provides temperature and humidity values to monitor the storage conditions. **Features**: 1.8 µA @ 1 Hz is very useful for battery operated devices. -40 to 85 degree C operation range. Can be kept inside the storage unit. **Protocol:** Device uses the I2C protocol.	https://cdn-shop.adafruit.com/datasheets/BST-BME280_DS001-10.pdf
IMU Sensor	BNO055	**Purpose**: Detect the vibrations to understand if the payload is experiencing turbulence. **Features:** Supports Low power operations where if motion is detected the accelerometer enters the Normal mode **Protocol:** The device supports I2C protocol.	https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf
GPS	MAX-M8-Q	**Purpose**: Find the latitude and longitude information to understand location to understand where the vibrations were sensed **Features:** Power save mode feature is present in this module as well. It is suitable for consumer applications because it makes use of halogen free gas. **Protocol:** The device supports UART protocol.	https://www.u-blox.com/sites/default/files/MAX-M8-FW3_DataSheet_%28UBX-15031506%29.pdf

Flash device	SST25VF512	**Purpose:** To log the GPS location and error condition (Vibration or temperature) for analysis **Features:** – Active Read Current: 7 mA (typical) – Standby Current: 8 µA (typical) – Endurance: 100,000 Cycles (typical) **Protocol:** The memory supports UART protocol	https://www.digikey.com/en/products/detail/microchip-technology/SST25VF512A-33-4C-SAE-T/2486542
Battery - 350 mAH	2750	**Purpose:** Provide a regulated power supply to the Microcontroller and other sensors/devices interfaced in the circuit. **Features:** Voltage: 3.7 V Max. Charging Current 350mA Max. Charging Voltage 4.2V Min. Supply Voltage: 3V **Protocol:** N/A	https://www.digikey.com/en/products/detail/adafruit-industries-llc/2750/6612470
PMIC	LTC3554	**Purpose:** Charge the LiPo battery in an optimized manner so as to increase the lifetime of the battery and thus the product. **Features:** ProjectUpdate3_VaXinator -> PowerEnergyCalculations_VaXinator.xlsx **Protocols:** N/A	https://www.analog.com/media/en/technical-documentation/data-sheets/355412ff.pdf
Load Switch	TPS22942DCKR	**Purpose:** Charge the LiPo battery in an optimized manner so as to increase the lifetime of the battery and thus the product. **Features:** ProjectUpdate3_VaXinator -> PowerEnergyCalculations_VaXinator.xlsx **Protocols:** N/A	https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftps22942
Chip Antenna	1575AT43A0040E	**Purpose:** GPS, Galileo, BeiDou (BDS), and Glonass (GLNSS), Quad-Band, Low Cost, Linearly Polarized, Omnidirectional Chip Antenna **Protocols:** N/A	https://www.johansontechnology.com/datasheets/1575AT43A0040/1575AT43A0040.pdf

Hardware Design

Schematic has been updated as final demo submission and included along with this in the folder.

Power Section Schematic Design:



- L1 = 4.7uH as per datasheet for Vout in 2.5 to 3.3V range
- R5, R6 calculated to get 3V out in order to utilize battery fully also as whole circuits cutoff voltage limit is 2.7v
- R5, R6 Calculated from Vout = 0.8v x ((R7 / R8) -1)
- C5 = 10uF recommended, selected 22uF to get effective capacitance of 10uF at DC bias
- Rprog = R2 = 2.2 K for max charge current of 340mA based on I(CHRG) = 750V / Rprog

Verification plan

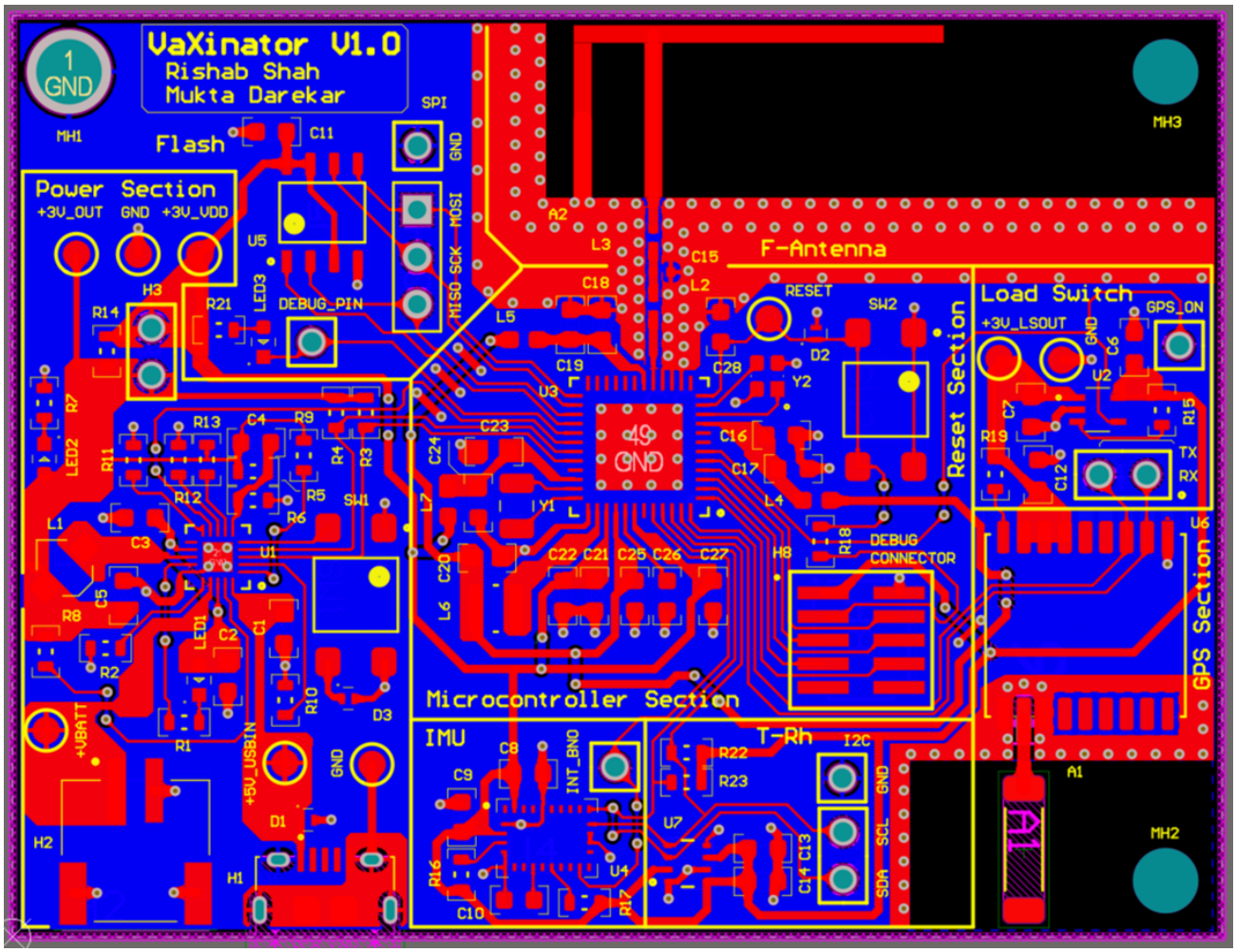
Detailed verification plan is present in folder in file VerificationPlan _VaXinator.xlsx

PCB verification

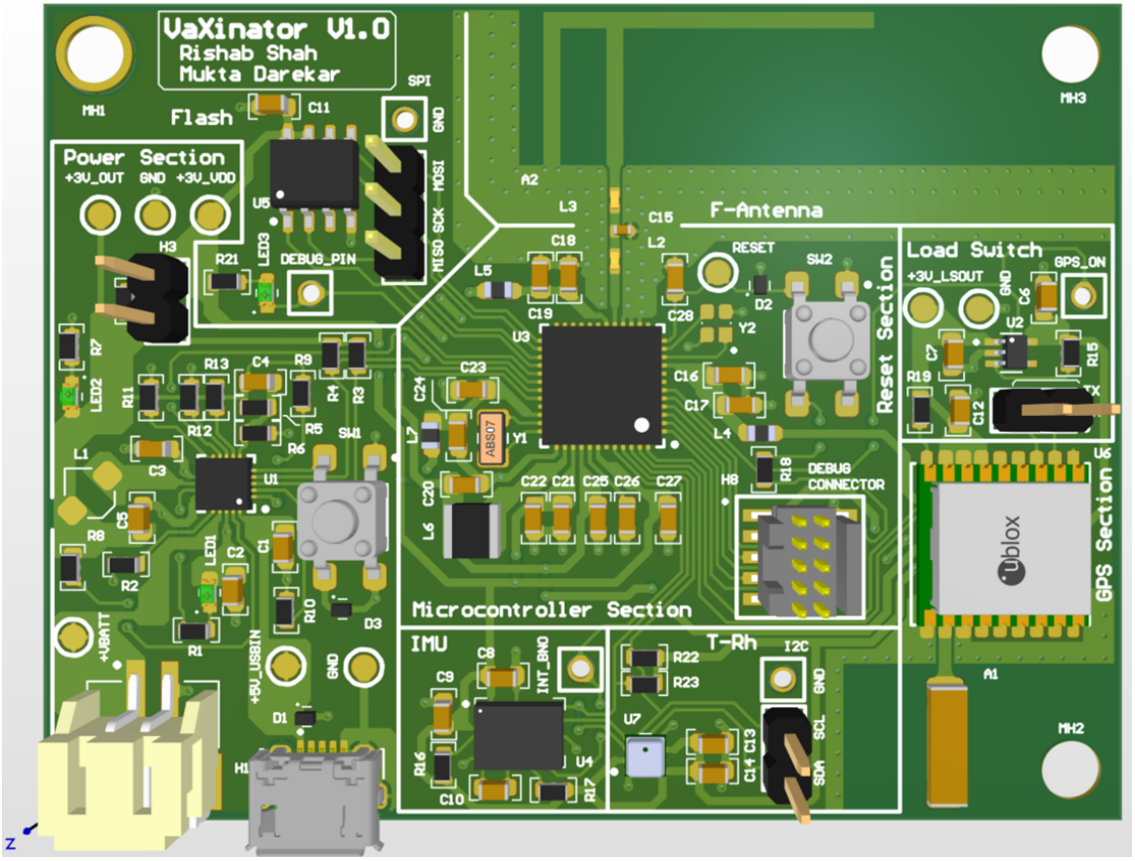
Received boards on Tuesday 5th April. Connectivity of the bare board was verified. All connections as per requirement. There is power ground short. Updated same in verification plan.

Layout Design

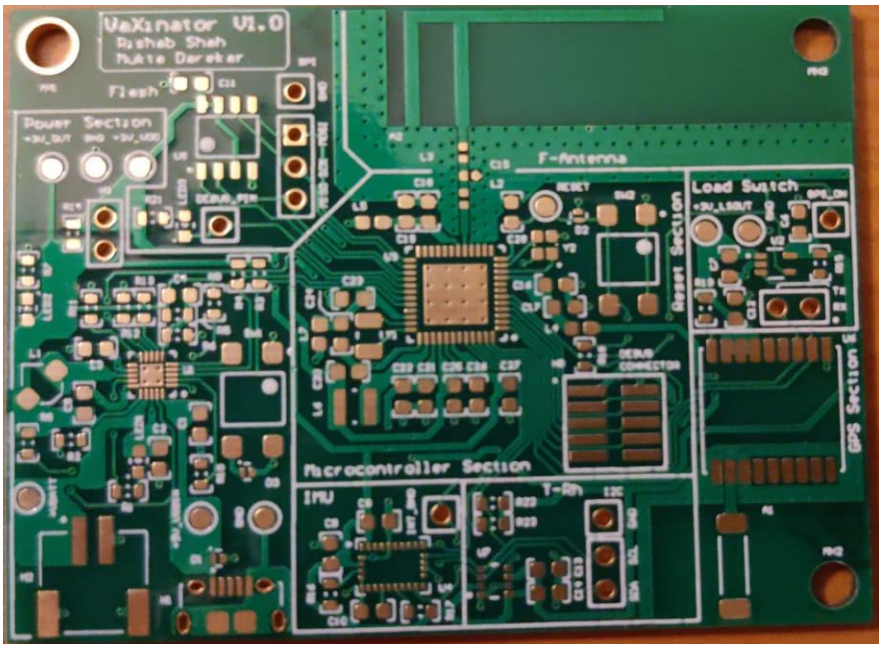
PCB 2D view Layout:



PCB 3D view Layout:



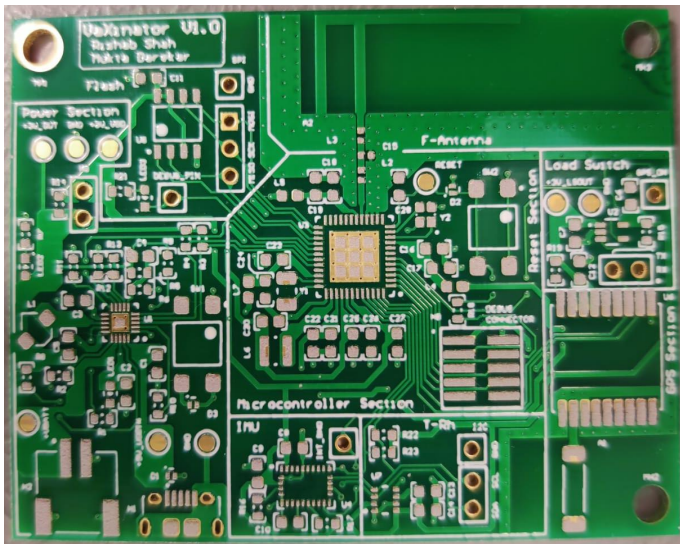
Bare board picture:



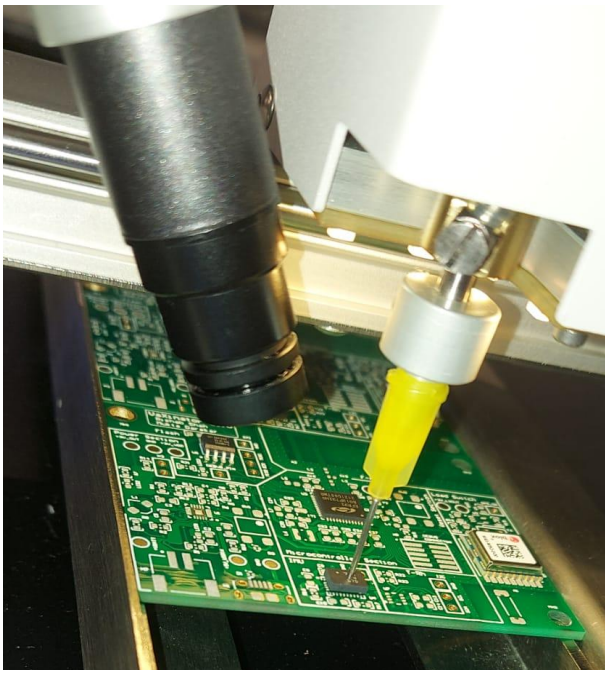
PCB Assembly

On Friday 8th April, critical component assembly on the pick and place machine was done. Rest of the assembly will be completed on monday.
We were able to get all the major components and we reflowed it so that the major ICs are well connected to the board. We plan to do a second reflow.
We could not get all the solder paste off before reflow. Mukta soldered these components on the board. Now, for some of the remaining components, we have to reapply solder paste by hand and assemble for a second reflow.

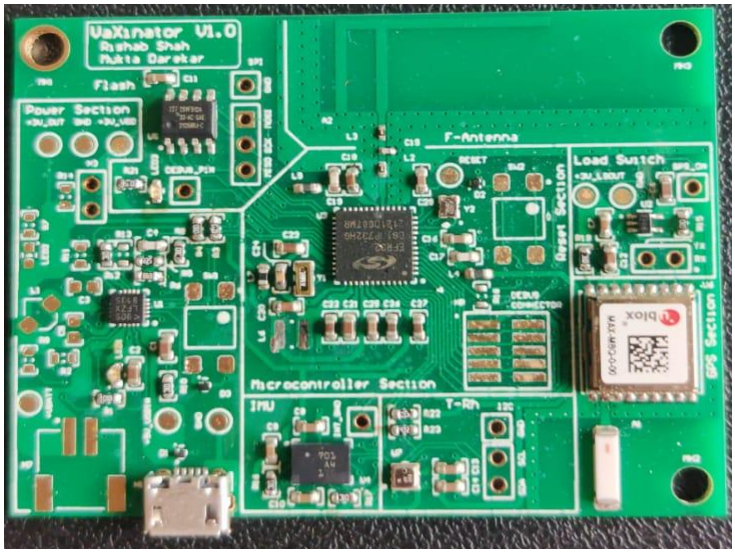
Solder paste applied board using Stencil



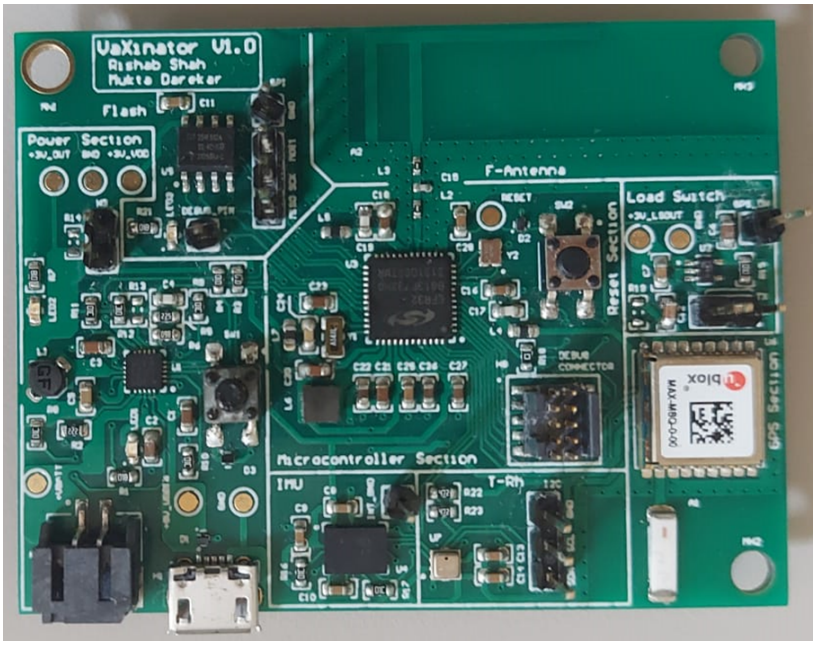
Assembling on pick and place machine



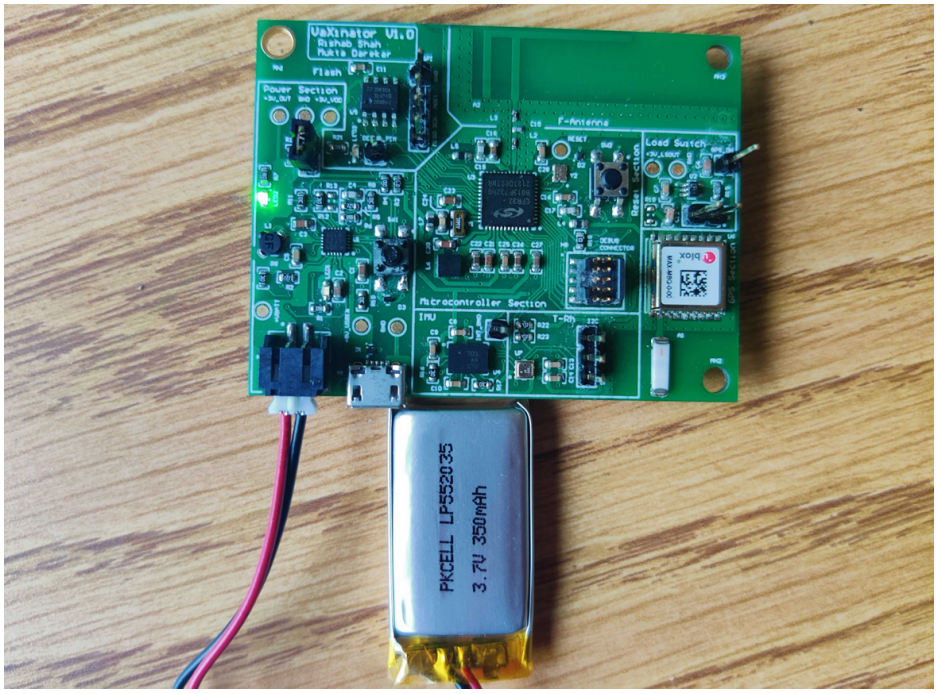
Partially assembled board picture



Fully assembled board picture



Working condition prototype



Battery Analysis

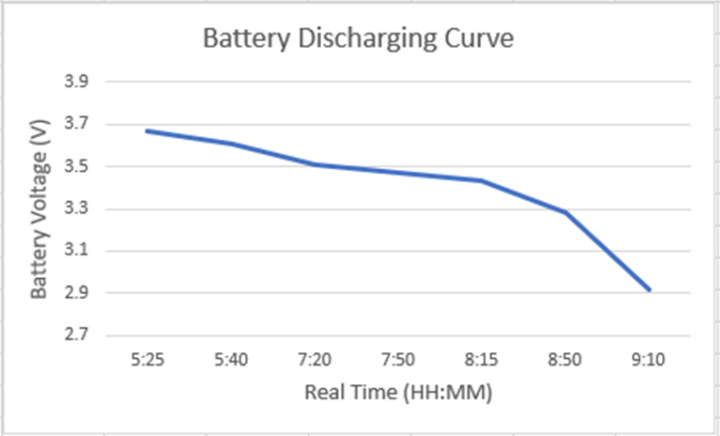
Discharged battery from its full voltage to the cutoff voltage after which it stopped working. The battery was brought down from 3.67V to 2.92V and it took 3hr 45min. So as per 0.2C C rate mentioned in the datasheet of the 350mAh battery, it should take 5hrs to discharge from 4.2V to fully discharged at max current of 350mA. But as we have tried it with keeping GPS on continuously at which project was consuming 42mA average current in EM0 mode of blue gecko. And as our application consumes an average current of 5.5mA, I believe this battery will satisfy our project statement about having a life span of at least 24hrs.

Also we charged our battery from 2.92V to 3.75V which took around 1hr 35min. This concludes that even after complete discharge, at the user end the battery can be recharged fast enough to be ready for next transit.

Battery discharge in State 4 - GPS

	Time(min)	Voltage
5:25	0	3.67
5:40	15	3.61
7:20	115	3.51
7:50	145	3.47
8:15	170	3.43
8:50	205	3.28
9:10	225	2.92

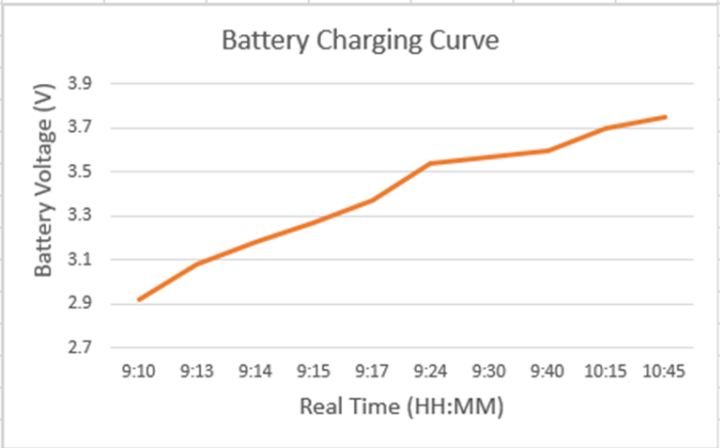
Total time: 3 hr 45 min

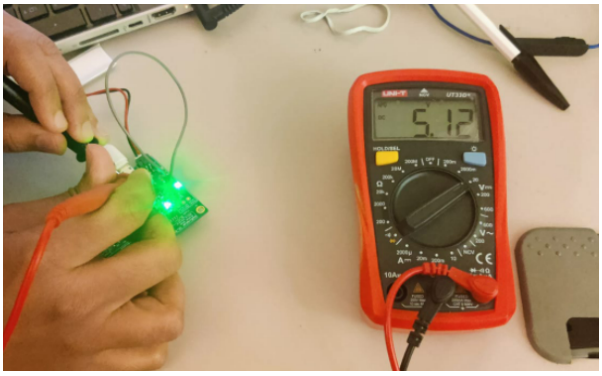


Battery charging - unit switched off

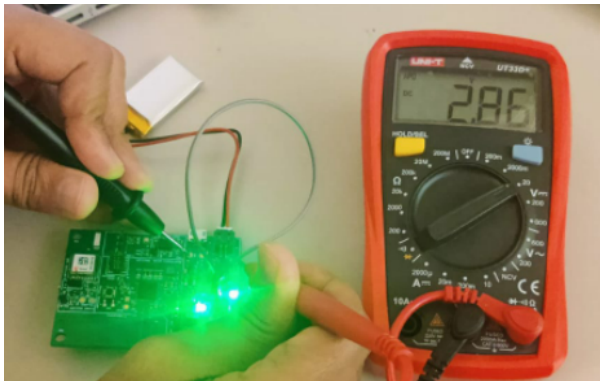
	Time(min)	Voltage
9:10	0	2.92
9:13	3	3.08
9:14	4	3.18
9:15	5	3.27
9:17	7	3.37
9:24	14	3.54
9:30	20	3.57
9:40	30	3.6
10:15	65	3.7
10:45	95	3.75

Total time: 1 hr 35 min





Battery Input - Charging condition - USB IN



Battery is charging from cut-off voltage

Low Power Mode Timeline

Theoretical rating of each use case : considered max possible current consumption per case

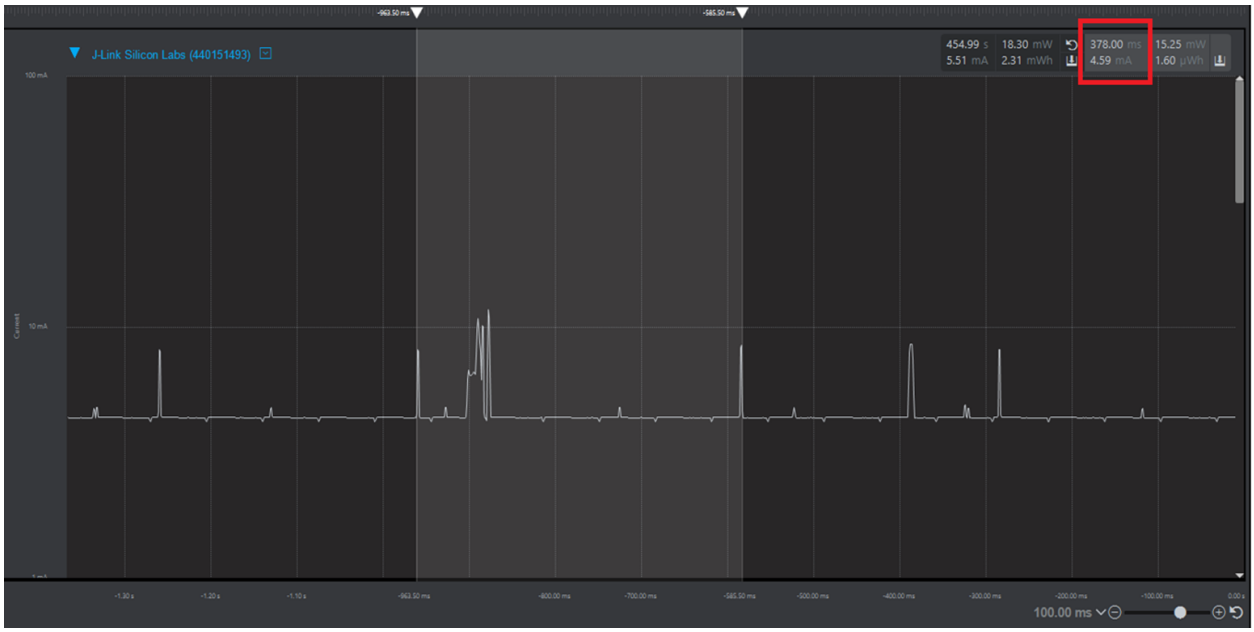
State 1 – Sleep
State 2 – measurement
State 3 – Bluetooth
State 4 – GPS

As per power calculations done for our project below are voltage and current ratings required to be supplied by the given circuit.

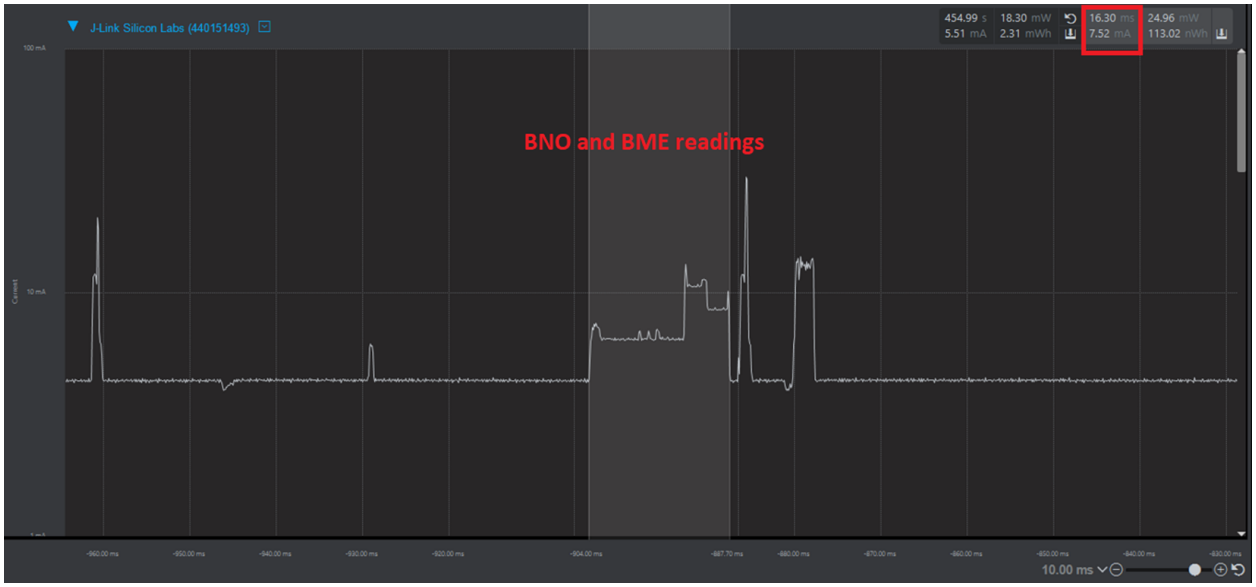
Parameter	State 1	State 2	State 3	State 4	Average
At Voltage (V)	3.3	3.3	3.3	3.3	3.3
Current (mA)	0.42	42	142	161	7.3
At Voltage (V)	3	3	3	3	3
Current (mA)	0.46	46	156	177	8

Theoretically considering 24hr duration and all the components in our circuit the average current comes around 8mA and practically achieved is 5.5mA as per the energy profiler outputs taken from Simplicity studio.

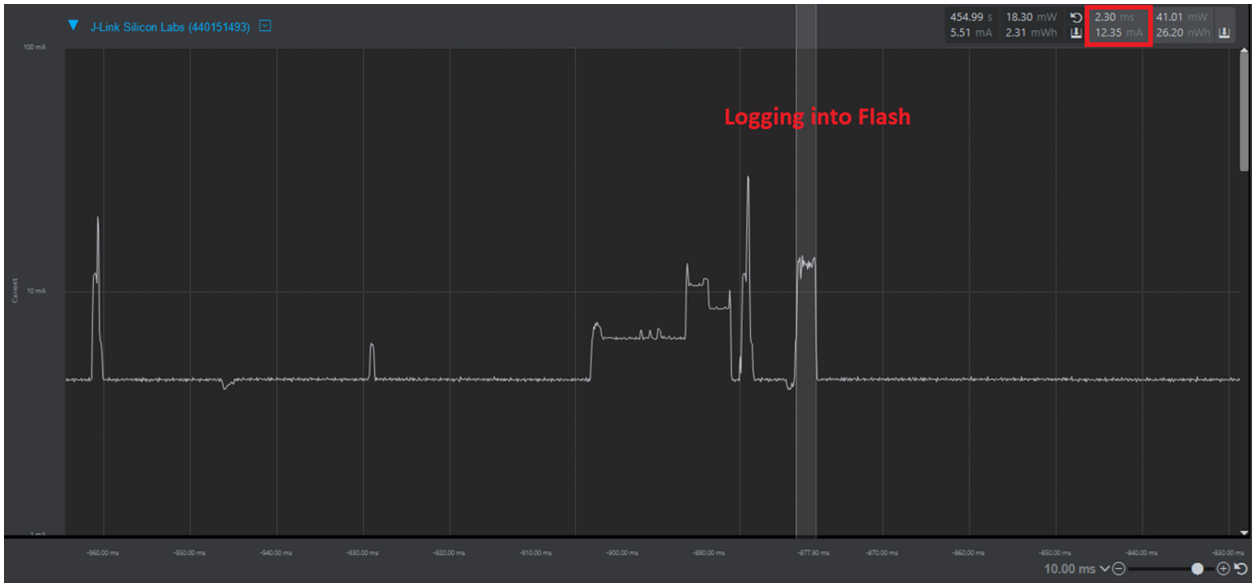
State 1: Average current consumed = 4.59mA - Sleeps in EM3



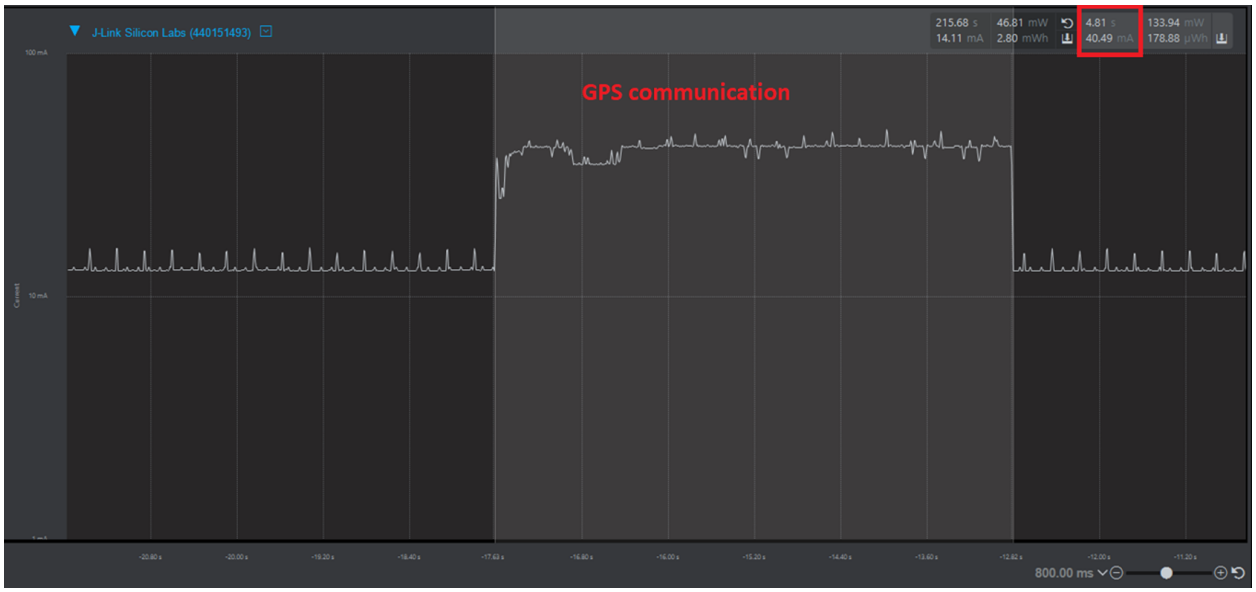
State 2.a: Average current consumed = 7.52mA - EFR sleeps in EM1 during this



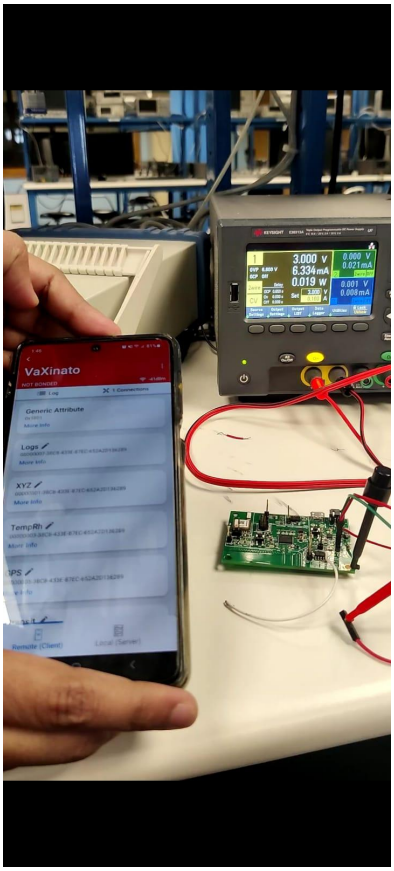
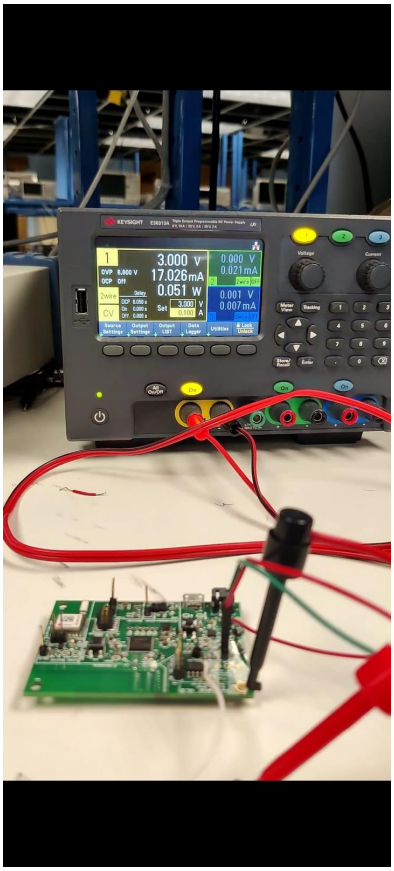
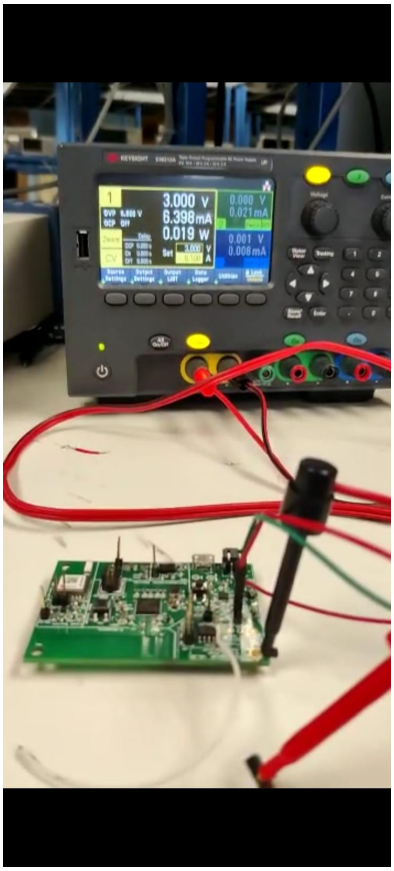
State 2.b: Average current consumed = 12.35mA - EFR sleeps in EM1 during this



State 4: Average current consumed = 40.49mA - EFR sleeps in EM1 during in between delays

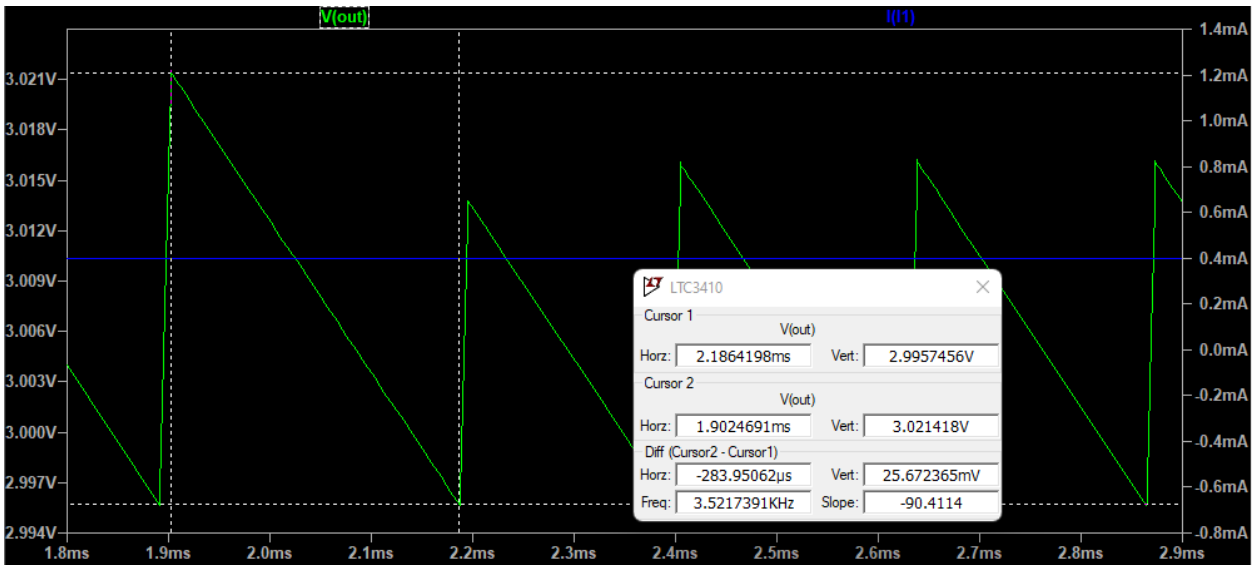


Similar verification from CC power supply:

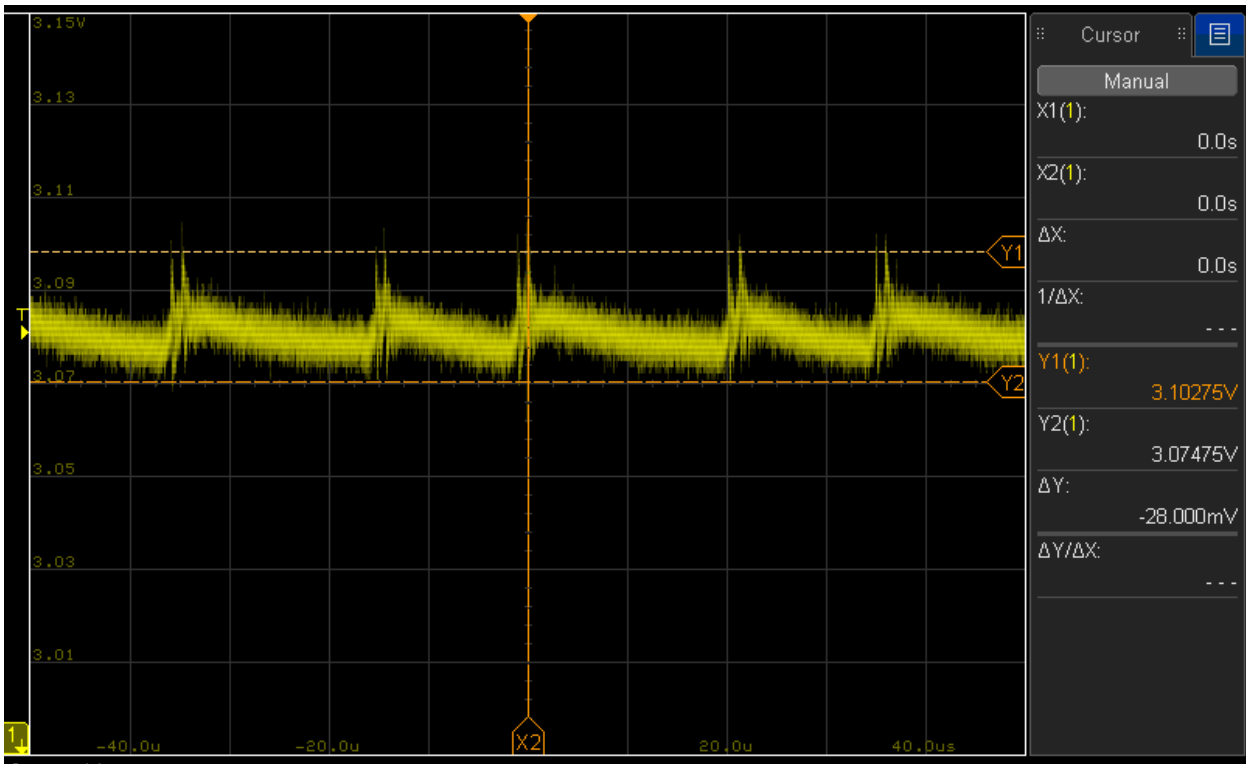
		
Connected State	Max. peak current in config	Sleep, Advertising state

DC/DC Verification

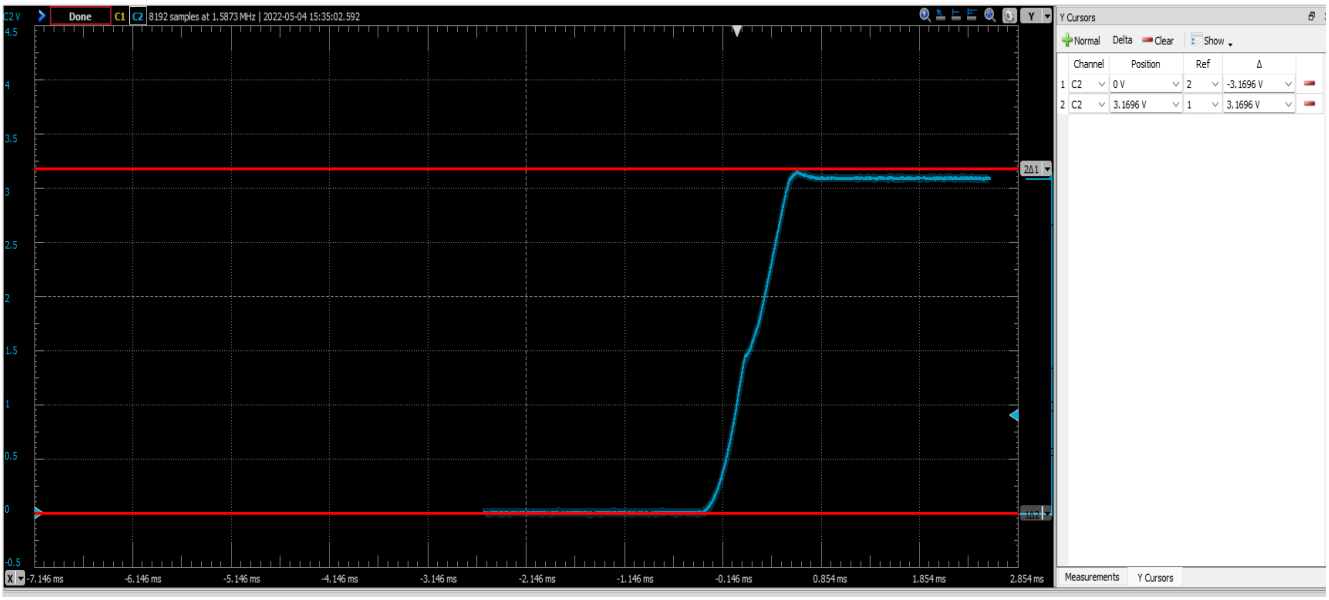
As you can see in the image below, when we did bulk capacitor simulation we got ripples in our dc/dc output (PMIC LTC3554 output) of around 26mV.



And when we practically did this on our project board we got ripples in voltage of around 28mV as shown in the image below.



On startup our voltage peak goes upto 3.169V as our PMIC circuit resistor divider values are set as to deliver voltage output of around 3.08V.



Software Design Overview

Following are the different components of the software.

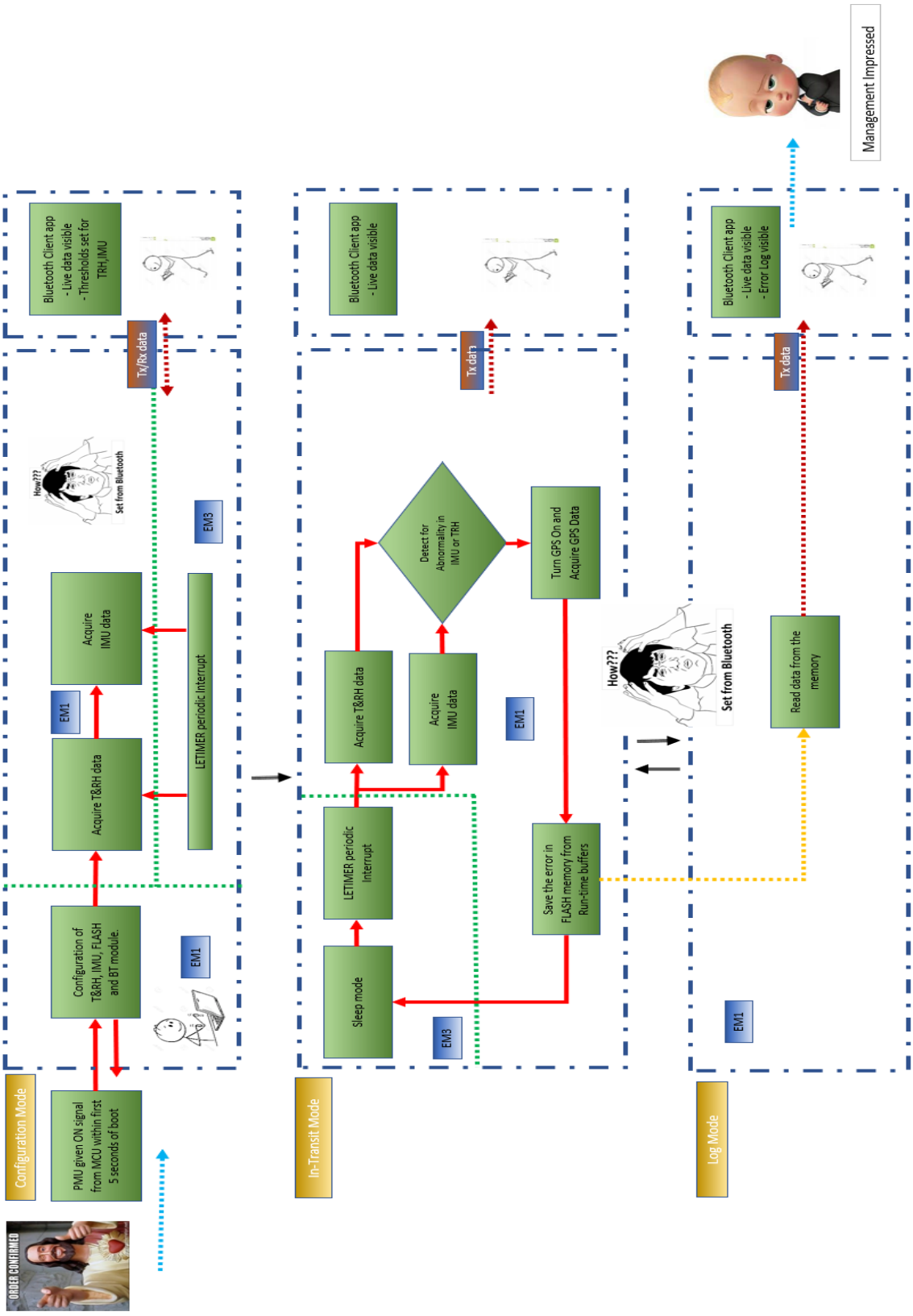
Board Firmware

Firmware design for Blue gecko EFR32BG13P**7**32F512GM48 has interfacing with BME280, BNO055 over shared I2C bus, Flash over SPI and GPS over LEUART.

Smartphone App - Silicon Labs

Capable to read/write the data to the custom board for control of all sensors and actuators.

Software Flow Diagram



Software Flow

In **Sleep mode**, All the peripherals are inactive. No communication (I2C, UART etc.) is taking place. Only LETIMER is functional and the respective LFXO clock generates periodic interrupts. The EFR32BG is in EM3 state to achieve the low power mode operation.

The application software runs in three different modes:

1. Configuration mode:
2. In-transit mode:
3. Log mode:

Configuration mode

All the initial configuration for sensors and actuators is done in this mode. The whole system is in EM1 mode for this operation. Once configuration is completed, the system periodically senses the environmental TRH and motion pace to stabilize the values and enter low power mode. The user then enters the TRH and IMU thresholds and then only the system makes a progress to the In-transit mode. The TRH and IMU are only operational and this live data is available on bluetooth application.

In-transit mode

In this mode, the user can have live data of the log as well as view the current values exhibited by the system. In this mode, any abnormal activity (threshold crossed) is captured by the system and is stored in the log by entering the EM1 mode from the EM3 sleep.

Log Mode

When the payload would reach its intended destination, the data stored in the FLASH memory would be extracted and transmitted over BLE to the client-app after authentication from the user. The device is operational in EM1 mode to transfer the data to the client after retrieving the data from the FLASH memory through SPI protocol.

Thus, VaXinator would continuously switch between Sleep mode and Run Mode (through Periodic interrupts) to acquire T&RH data i.e Transition occurs between EM3 and EM1.

In the above diagram, The various EM modes are mentioned through which the device makes transitions for proper operation as well as save huge margins of energy through low-power mode operation. The feature of ease in Energy management facilitated by EFR32BG through EM0,EM1,EM2,EM3 modes has made BG a remarkable device to manage power consumption.

```c
void AssetMonitoringSystem_StateMachine(sl_bt_msg_t* event) {
  asset_monitoring_state_t current_state;
  //static activity_monitoring_state_t next_state = HEARTBEAT_INIT;
  static asset_monitoring_state_t next_state = BME280_INIT_CONFIG;
  if (SL_BT_MSG_ID(event->header) != sl_bt_evt_system_external_signal_id) {
      return;
  }

  current_state = next_state;

  switch (current_state) {
    case BME280_INIT_CONFIG:
        next_state = init_bme280_machine(event);
        //LOG_INFO("BME280_INIT_CONFIG\r");
        break;

    case BNO055_INIT_CONFIG:
        next_state = init_bno055_machine(event);
        //LOG_INFO("BNO055_INIT_CONFIG\r");
        break;

    case BME280_READ:
        next_state = bme280_read_machine(event);
        //LOG_INFO("BME280_READ\r");
        break;

    case BNO055_READ:
        next_state = bno055_read_machine(event);
        //LOG_INFO("BNO055_READ\r");
        break;

    case DEBUG_READ:
        next_state = debug_machine(event);
        //LOG_INFO("DEBUG_READ\r");
        break;

    default:
      break;
  }
}
```

Init_bme280_machine and init_bno055_machine run in EM1 mode whereas bme280_read_machine and bno055_read_machine run in EM3 mode and at LETIMER triggers wakes up in EM1 mode

List of Commands (GATT_DB Commands and Characteristics)

Bluetooth Command Table for our Project which acts as a Server:

Event	**Description**	**Values to save**	**Commands to call in response to event**	**Command description**
sl_bt_evt_system_boot_id	This event indicates the device has started and the radio is ready. Do not call any stack API commands before receiving this boot event, including starting BT stack soft timers!		sl_bt_system_get_identity_address()	Returns the unique BT device address
			sl_bt_advertiser_create_set()	Creates an advertising set to use when the slave device wants to advertise its presence
			sl_bt_advertiser_set_timing()	Sets the timing to transmit advertising packets
			sl_bt_advertiser_start()	Tells the device to start sending advertising packets
sl_bt_evt_connection_opened_id	This event indicates that a new connection was opened.	Connection handle. May want to maintain a flag that tracks whether a connection is open or closed.	sl_bt_advertiser_stop()	Stop advertising
			sl_bt_connection_set_parameters()	Send a request with a set of parameters to the master
sl_bt_evt_connection_closed_id	This event indicates that a connection was closed.	May want to maintain a flag that tracks whether a connection is open or closed.	sl_bt_advertiser_start()	Tells the device to start sending advertising packets

sl_bt_evt_connection_parameters_id	Informational. Triggered whenever the connection parameters are changed and at any time a connection is established			
sl_bt_evt_gatt_server_characteristic_status_id	Indicates either that a local Client Characteristic Configuration descriptor (CCCD) was changed by the remote GATT client, or that a confirmation from the remote GATT client was received upon a successful reception of the indication	Track whether indications are enabled/disabled for each and every characteristic. Track whether an indication is in flight or not.		
sl_bt_evt_gatt_server_indication_timeout_id	Possible event from calling sl_bt_gatt_server_send_indication() - i.e. we never received a confirmation for a previously transmitted indication.	Track whether an indication is in flight or not.		

Bluetooth Services added for our Project Functionality (Indicate Only)

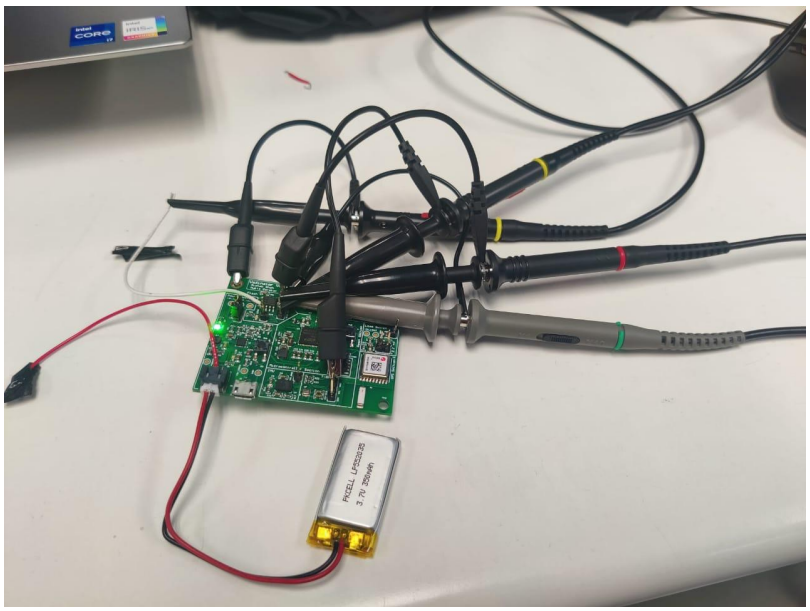
Service Name	Accelerometer Service	Temperature Humidity Service	GPS Service	Debug Service
Service UUID Value	00000001-38c8-433e-87ec-652a2d136289	00000003-38c8-433e-87ec-652a2d136289	00000005-38c8-433e-87ec-652a2d136289	00000007-38c8-433e-87ec-652a2d136289
Characteristics Name	XYZ Characteristic	TRH Characteristic	Coordinate Characteristic	Debug Info
Characteristics UUID Value	00000002-38c8-433e-87ec-652a2d136289	00000004-38c8-433e-87ec-652a2d136289	00000006-38c8-433e-87ec-652a2d136289	00000008-38c8-433e-87ec-652a2d136289
Characteristics Configuration	Indicate	Indicate	Indicate	Indicate
CCCD Name	client_characteristic_configuration_5	client_characteristic_configuration_6	client_characteristic_configuration_7	client_characteristic_configuration_8
CCCD UUID Value	2902	2902	2902	2902
CCCD Configuration	Read, Write	Read, Write	Read, Write	Read, Write
Description	Gives Accelerometer XYZ live values Format: X:aa.a Y:bb.b Z:cc.c	Gives Temperature and Humidity live values - in degC and percentage Format: xxC yyR	Gives Latitude and Longitude coordinates with Date and Time Format: +- LAT +-LON DD:MM:YY HH:MM:SS	Gives Live error logging info while in Transit and when in logging mode gives all Logged values one after other from start Format: Date Time Temp RH A0/A1 Lat Lon (A0 - No shock, A1 - Vibration experienced)

Bluetooth Services added for our Project Functionality (Read Write)

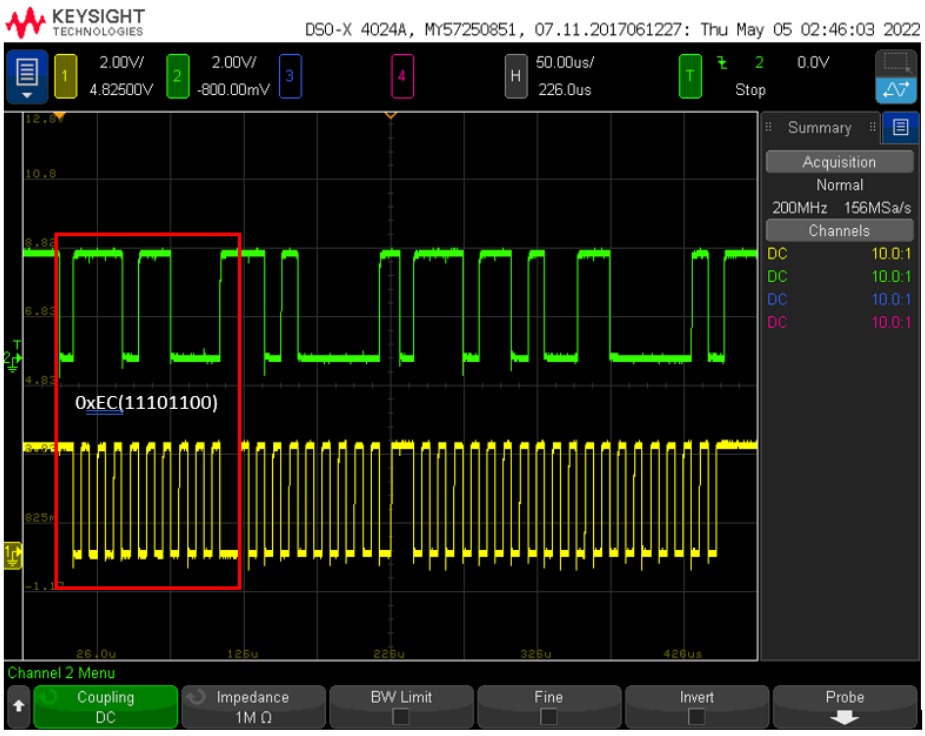
Service Name	Start Transit	Send Error log	Thresholds				
Service UUID Value	00000009-38c8-433e-87ec-652a2d136289	0000000b-38c8-433e-87ec-652a2d136289	0000000d-38c8-433e-87ec-652a2d136289				
Characteristics Name	Transit status	Enable Error log	Temp High Limit	Temp Low Limit	RH High Limit	RH Low Limit	accelerometer limit
Characteristics UUID Value	0000000a-38c8-433e-87ec-652a2d136289	0000000c-38c8-433e-87ec-652a2d136289	000000 1d-38c8 -433e-8 7ec-652 a2d136 289	000000 2d-38c8 -433e-8 7ec-652 a2d136 289	000000 3d-38c8 -433e-8 7ec-652 a2d136 289	0000004 d-38c8-4 33e-87e c-652a2 d136289	000000 5d-38c 8-433e- 87ec-6 52a2d1 36289
Characteristics Configuration	Read, Write, Indicate	Read, Write, Indicate	Read, Write, Indicate	Read, Write, Indicate	Read, Write, Indicate	Read, Write, Indicate	Read, Write, Indicate
CCCD Name	client_characteristic_configuration_9	client_characteristic_configuration_10	client_characteristic_configuration_11	client_characteristic_configuration_12	client_characteristic_configuration_13	client_characteristic_configuration_14	client_characteristic_configuration_15
CCCD UUID Value	2902	2902	2902	2902	2902	2902	2902
CCCD Configuration	Read, Write	Read, Write	Read, Write	Read, Write	Read, Write	Read, Write	Read, Write
Description	To start / stop Transit mode for client use Enter 1 to start, 0 to stop	To enter into Error logging stage Enter 1 to start sending error logs And 0 is updated by device after all logs are sent	To set high Temp value - in Ascii and enter in celsius	To set lowTemp value - in Ascii and enter in celsius	To set high RH value - in Ascii and enter in percentage	To set low RH value - in Ascii and enter in percentage	To set XYZ value difference - in Ascii

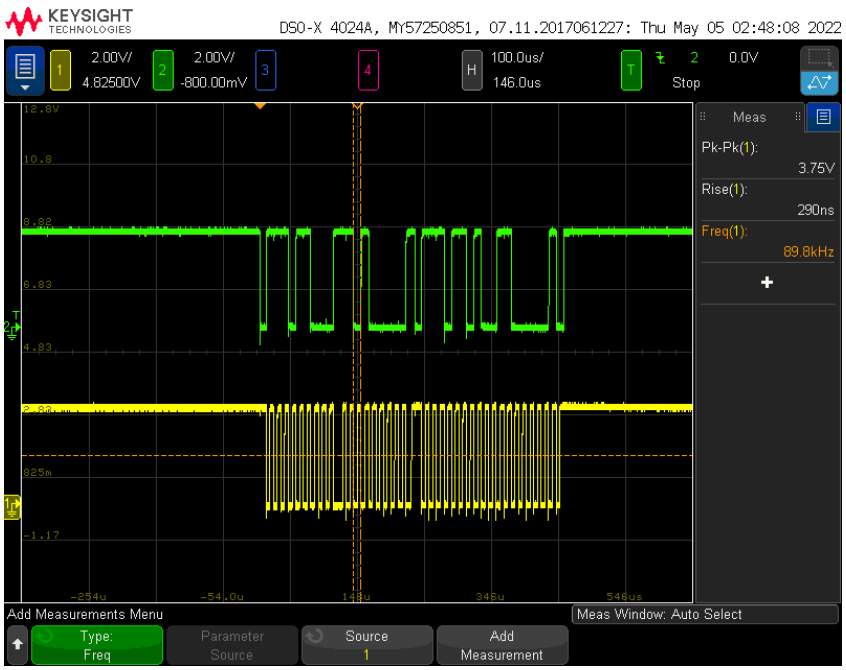
Signal traces

Following setup was used to take the traces:



a) I2C traces

BME280 was used for I2C signal integrity analysis using SDA high to low as a trigger signal	
	This is a trace to show that the I2C signal is very clean. It can read that the address sent on the bus is 0XEC which is 0x76 << 1 and 0x76 is the BME280 chip address SCL is Yellow SDA is Green



I2C CLock frequency:
In code 100KHz was used and we get a clock of 89.28 KHz frequency.

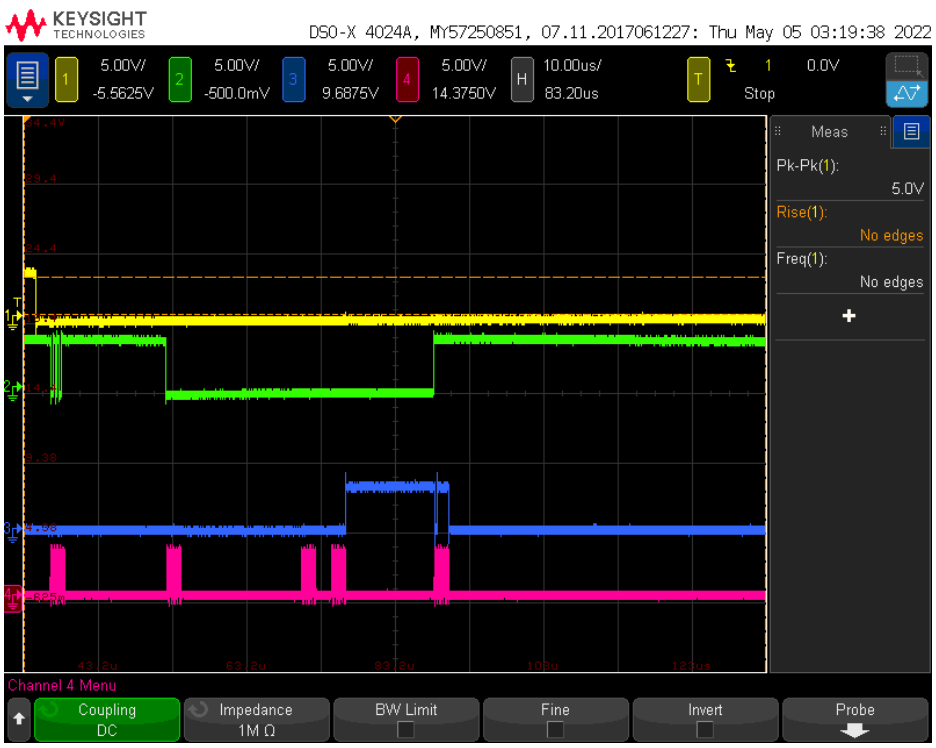
SCL is Yellow
SDA is Green

I2C Rise time of SCL signal: 316 nsec

SCL is Yellow
SDA is Green

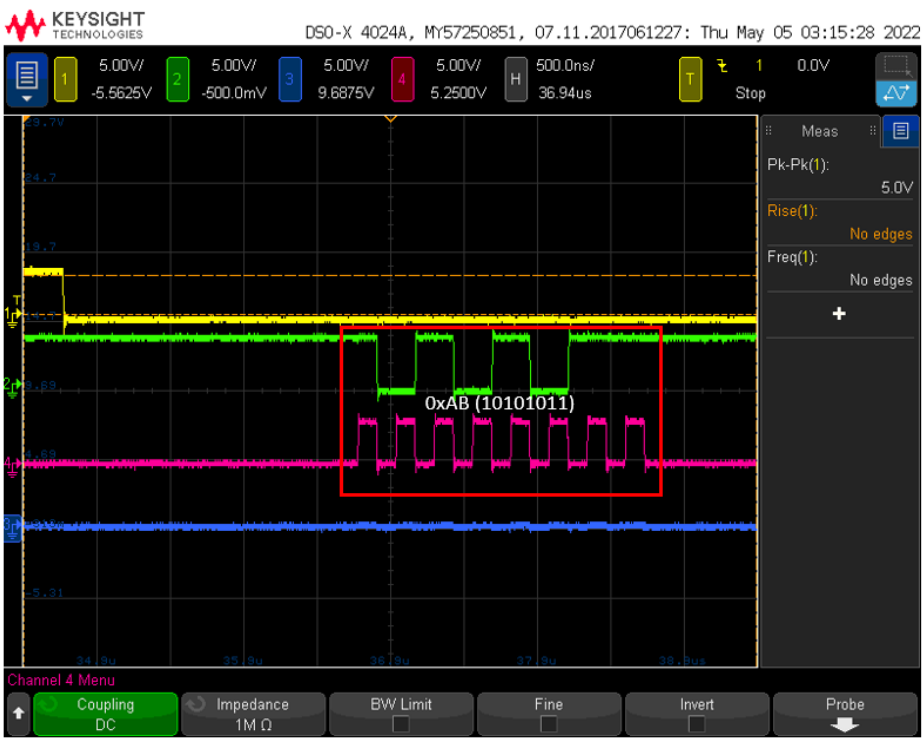
b) SPI traces

CS line from high to low transition was used to trigger the SPI signal going to flash



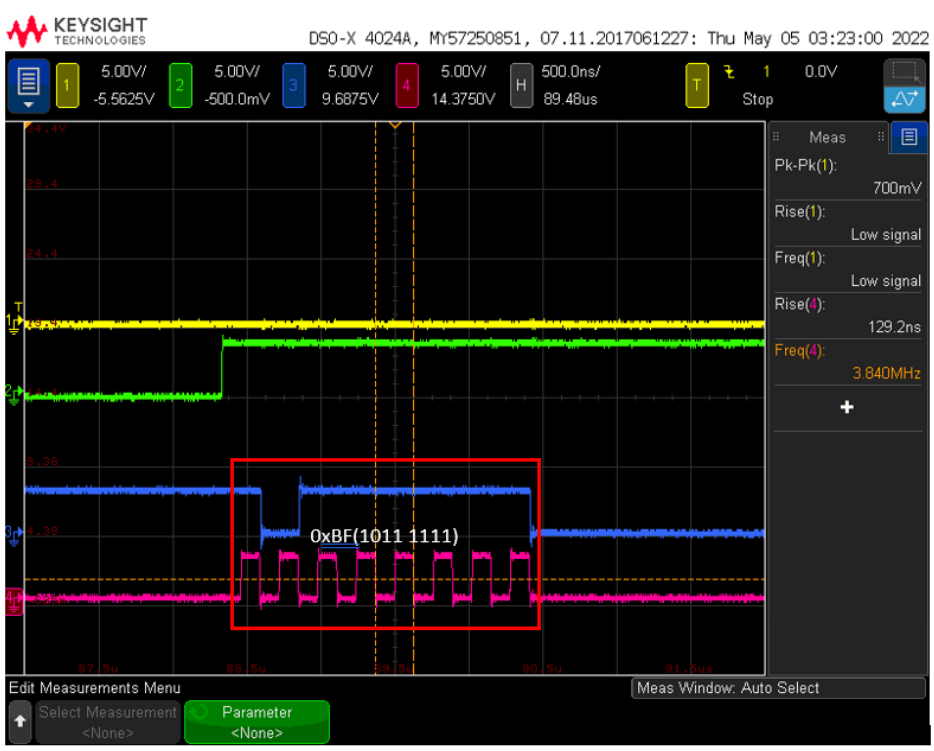
This is an overview of all the signals.
Yellow is Chip select
Green is MOSI
Pink is Clock
Blue is MISO

A whole transaction is captured



Yellow is Chip select
Green is MOSI
Pink is Clock
Blue is MISO

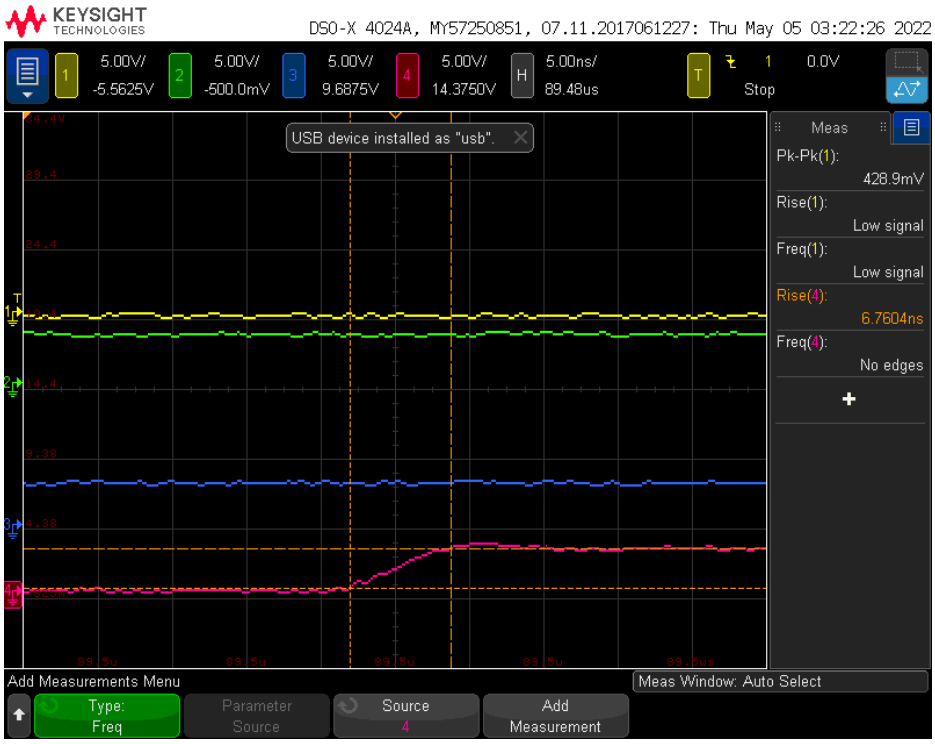
Comparing MOSI and CLK , we can read 0xAB sent in the line which is the FLASH chip address read command instruction



Yellow is Chip select
Green is MOSI
Pink is Clock
Blue is MISO

Flash Frequency is 3.84 Mhz and in code 4 Mhz was set.

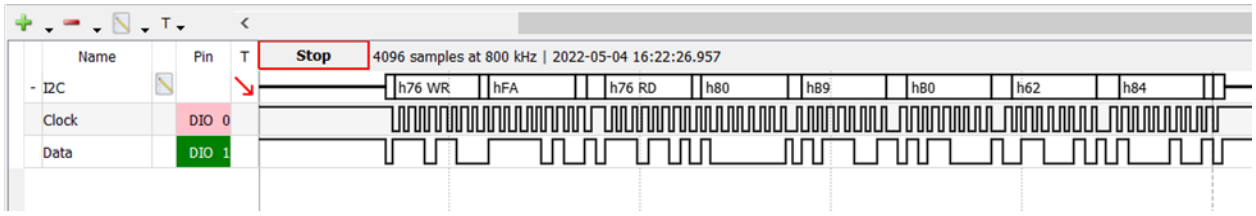
Also, we can see the manufacturer ID is read out as 0xBF



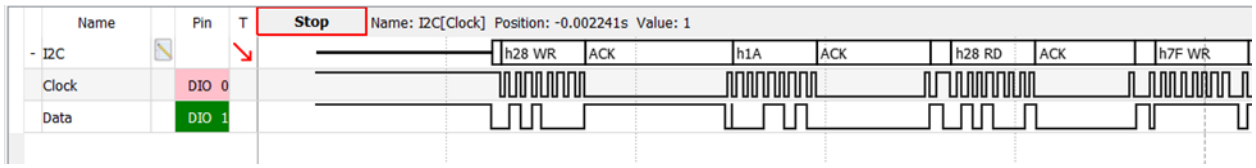
Yellow is Chip select
Green is MOSI
Pink is Clock
Blue is MISO

Flash rise time is 6.76 nsec

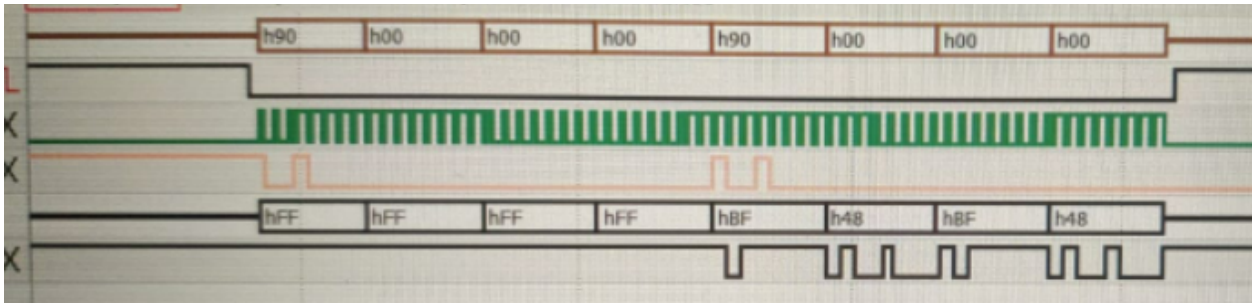
I2C AD2 traces for BME280



I2C AD2 traces for BNO055



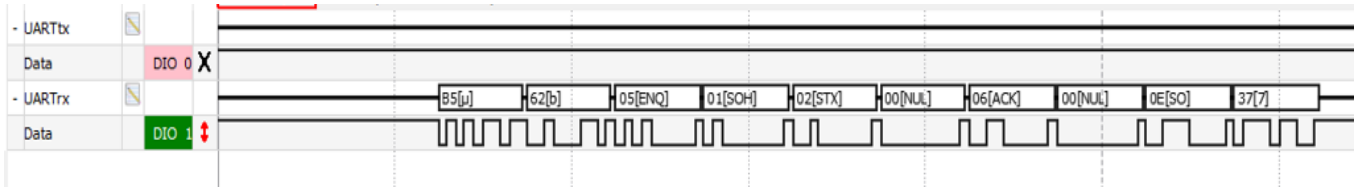
SPI AD2 traces for FLASH



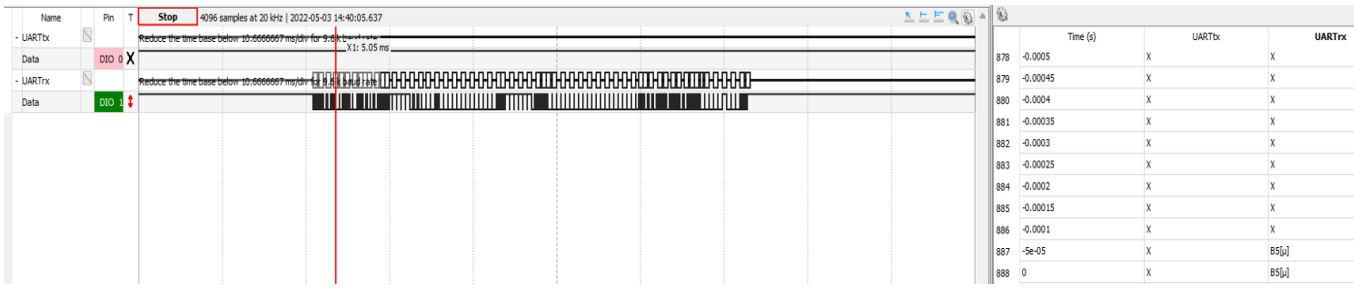
LEUART traces

GPS

ACK frame received on completion of settings



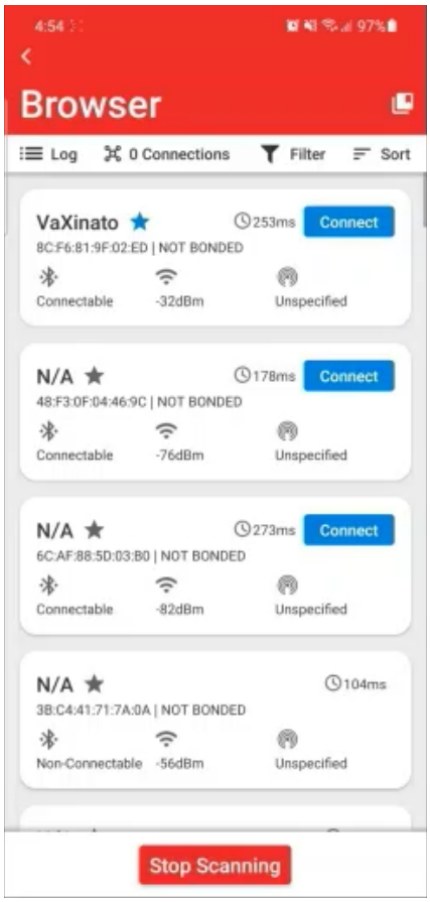
GPS return frame received as shown further in detail of the message received

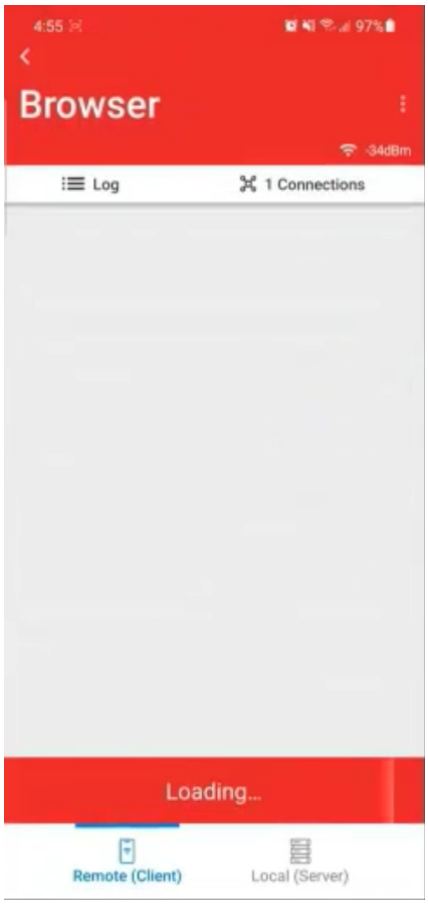


Project Working

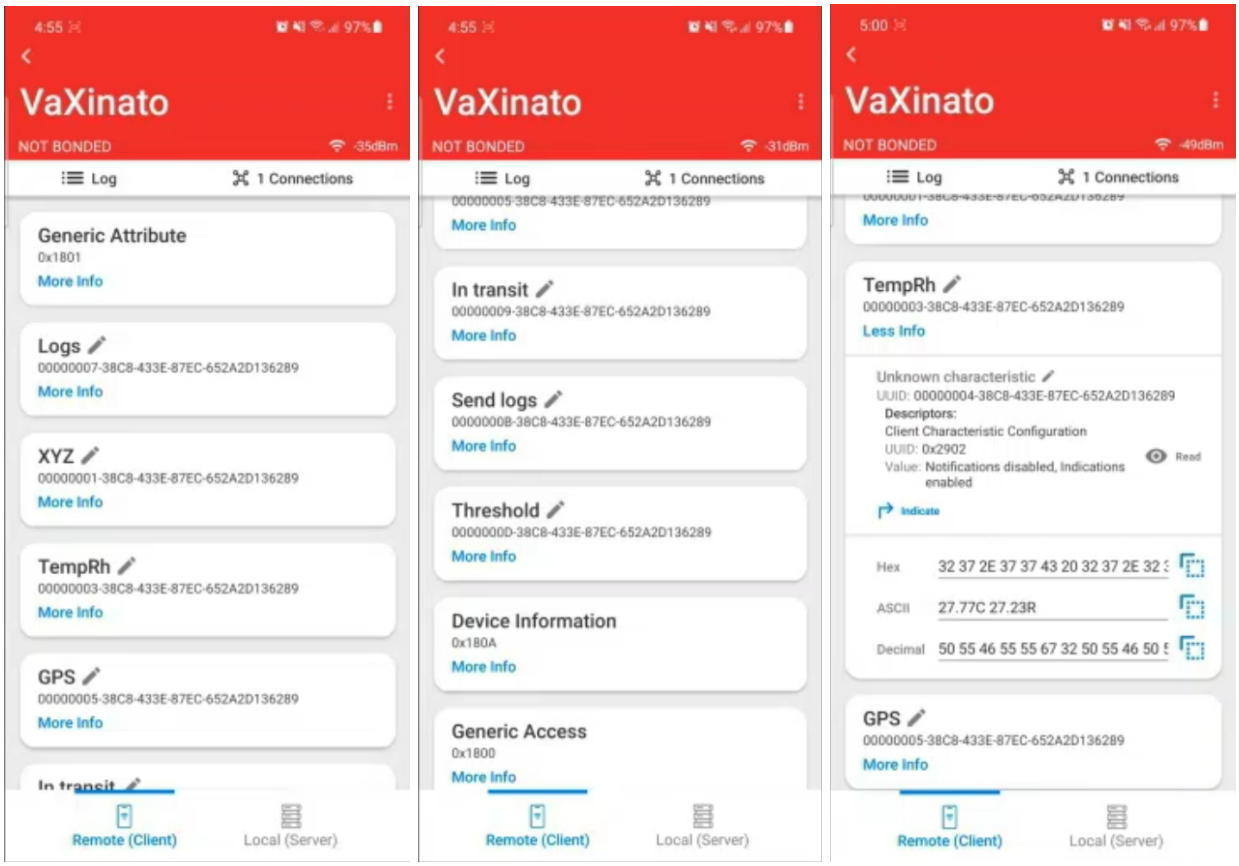
As explained in Software Flow the operation is divided into 3 modes and at the user end, the user has to make sure of which mode the device is in and what to set in order to get it working as per your requirement.

At power on, everything is initialized and all values are set to default as zero. After which device is made discoverable for any clients to connect to as well as the routine of getting values from all sensors is started at an interval of 5 sec in background.

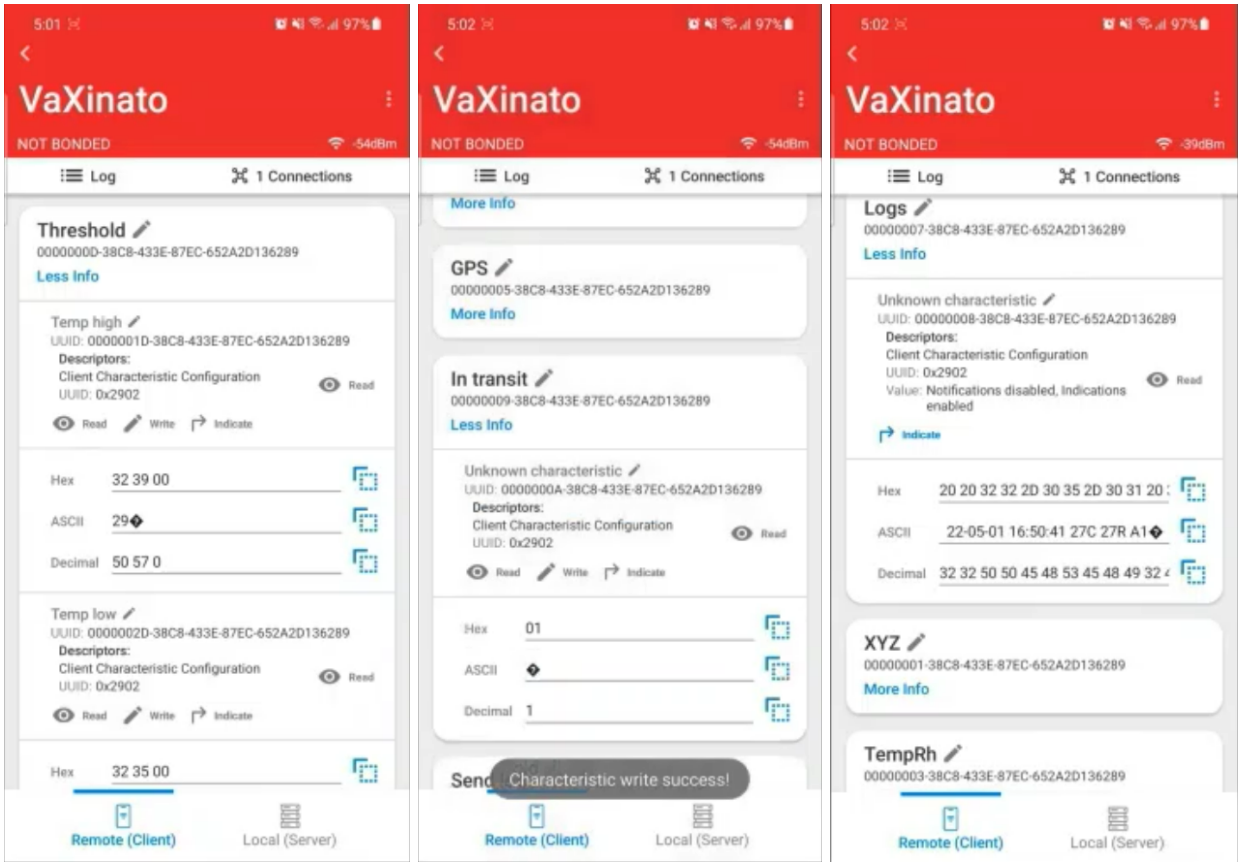




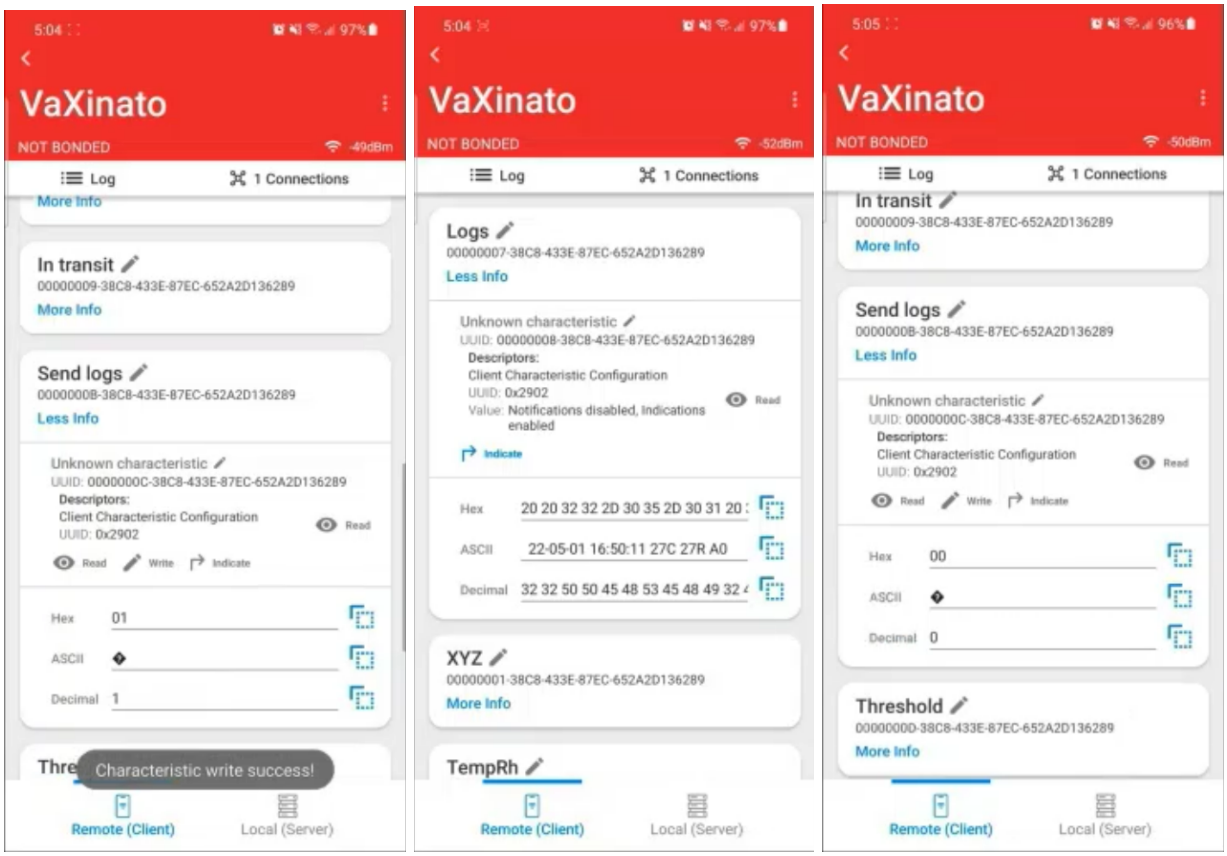
Once connected to the device in the mobile app you can view all available services as mentioned previously in bluetooth gatt command tables. The device at this stage is still in first configuration mode where you are able to view present sensed values by sensors



To proceed to the transit stage you first have to set threshold values and then set the in transit flag to high. After which in case of any movement accelerometer will detect it or in case of temperature and humidity is detected out of range of set threshold then will log it into flash memory along with sending the same to bluetooth if connected to any client as shown below.



After transit is complete, at destination if you want to check all logs from start then first you have to reset the transit flag and then set error flag to get log data read out of the flash and send it over to bluetooth to client. On completion of sending all logs and there are no more logs to send then device will reset the send log flag at client end through bluetooth in order to notify the client of the same.



Project Challenges

Hardware

1. Selection of components and Altium library creation

Some parts were required to be created from scratch as their footprint was not available such as BNO-055, BME280, EFR32, LTC3544. We received some help from SA in terms of giving the remaining pieces.

Only GPS was found online and that was partial. DImensions required to be changed.

Resistors and capacitors were chosen incorrectly without conderationfor derating capability. We then made a selection of resistors and capacitors from the same company so that a footprint for one can be used for all 0603 or 0402 parts without any further issue.

A debug switch would have been better to have. We had difficulty in procuring an inductor for PMIC as it was very unique in shape as well as value.

2. **ITLL Assembly**

Assembly of EFR32, BME280, LTC3544 and PMIC was very difficult to assemble. We could not start our Assembly process at 12 pm as expected because it was occupied by other teams in Spite of booking the facility. As everyone was trying to work on the same day. We had to do a partial reflow of our boards on friday as we could not keep it for 2 more days in the fridge. In the available time, we could only remove some solder paste without damaging the placement of other components and worked on hand-soldering the rest of the components. This consumed a lot of time which was more than expected.
Soldering was difficult as the components were tiny but we were able to get all major components for our first reflow
.

3. **Bring-up**

During bring-up, it would have been better to have a header for Battery connector. Before powering through the battery , a battery was simulated through the power supply and if there were test points we could have hooked up a power supply directly through jumper wires instead of soldering wires.

For doing a cross testing where Flash of the custom board was interfaced with evaluation board we wished we had a pin for Chip select. Similarly, we observed that all unused gpio pins should be provided just in case we want to use those pons later if we want to add any functionality dynamically or create more easy debugging combinations.

The pins should have also been placed a little apart then it would have been easier to place the probes side by side but this is an accepted tradeoff for minimizing the size.

4. **Legend Size**

Legend size was 20x6 mil and it was a little inconvenient as it did not serve the purpose for which it was added. It is better to go with 25x6 mil

Software

1. **Development IDE**

Development IDE was configured for SDK3.0 whereas Mukta's was for SDK 5.0 due to a windows upgrade on her system. Our code was not portable and we could not make any progress further in parallel fashion. One fine day, the SDK 3.0 **.sls** file was shared and then we got both laptops ready for parallel code development but by this time the course had picked up pace and we were in the schematics and layout phase which was more important. Also, .zip file is corrupted.

2. **Bring-up debug connector and Blinky cod**e

We had to really explore on the day of assembly before soldering the header and finding the orientation for programming pins coming from the gecko board to the custom board. We are thankful to Jake for providing the link for programming the EFR32 chip. It helped us to find a direction and program the data.

3. **BNO-055**

The BNO-055 was not set in the GPIO based interrupt functionality. Also, BNO-055 and evaluation board worked perfectly even the address in a polling operation happened correctly but reading of BNO-055 address on the custom board always ended up with a NACk and did not exit the infinite loop. This address checking for BNO-055 at start was removed and we were able to execute our state machine seamlessly then.

4. **FLASH**

The SPI driver support initially used was SPI with LDMA approach. In this mechanism, there was no control on the amount of data read from the chip in spite of many attempts. One fine attempt before fabrication we were able to read the chip address and were happy with it. However, writing further down, the firmware code was not possible to develop. Hence, wote own firmware driver to resolve this and the required configuration for the same. This approach finally paid off and the chip address was read with better control. Our read operation with this driver was successful but the write operation was not.Even write enable was given. The issue was clear after reading the datasheet in depth and it was seen that BPL (Block protection) bits were On) by default on power-on after professor's advice to check the memory protection. However, this became clear only when the Read Status Register was accessed from the FLASH memory. This was like having a debug view into the Flash system and being able to get FLASH working. All of the further driver functions which were developed from scratch were written in a day - a continuous coding session.

5. **GPS**

At start during before board bringup we used inbuilt LEUART driver provided by SDK along with LDMA to communicate to the GPS module. This combination worked on blue gecko eval board interfaced with GPS module giving below output,

The data received is "Recommended minimum data" - GNRMC request - which is required in our application.

Frame format -

$xxRMC,time,status,lat,NS,lon,EW,spd,cog,date,mv,mvEW,posMode,navStatus*cs<CR><LF>

Required fields for our application are Time, Latitude, Longitude, Date and we got them on the module.

But as we haven't tested leuart with all other peripherals turned on on eval board it started to create issues and hence wrote leuart direct driver for these. We were able to get UBX type message format on our custom board instead of NMEA string format messages. But the frame even after getting ACK signal for setting commands returns a frame with invalid data. Hence not used in our project. Instead found a way around and configured internal RTC to get the current time and date for timestamp in error logs.

Tried to get below message

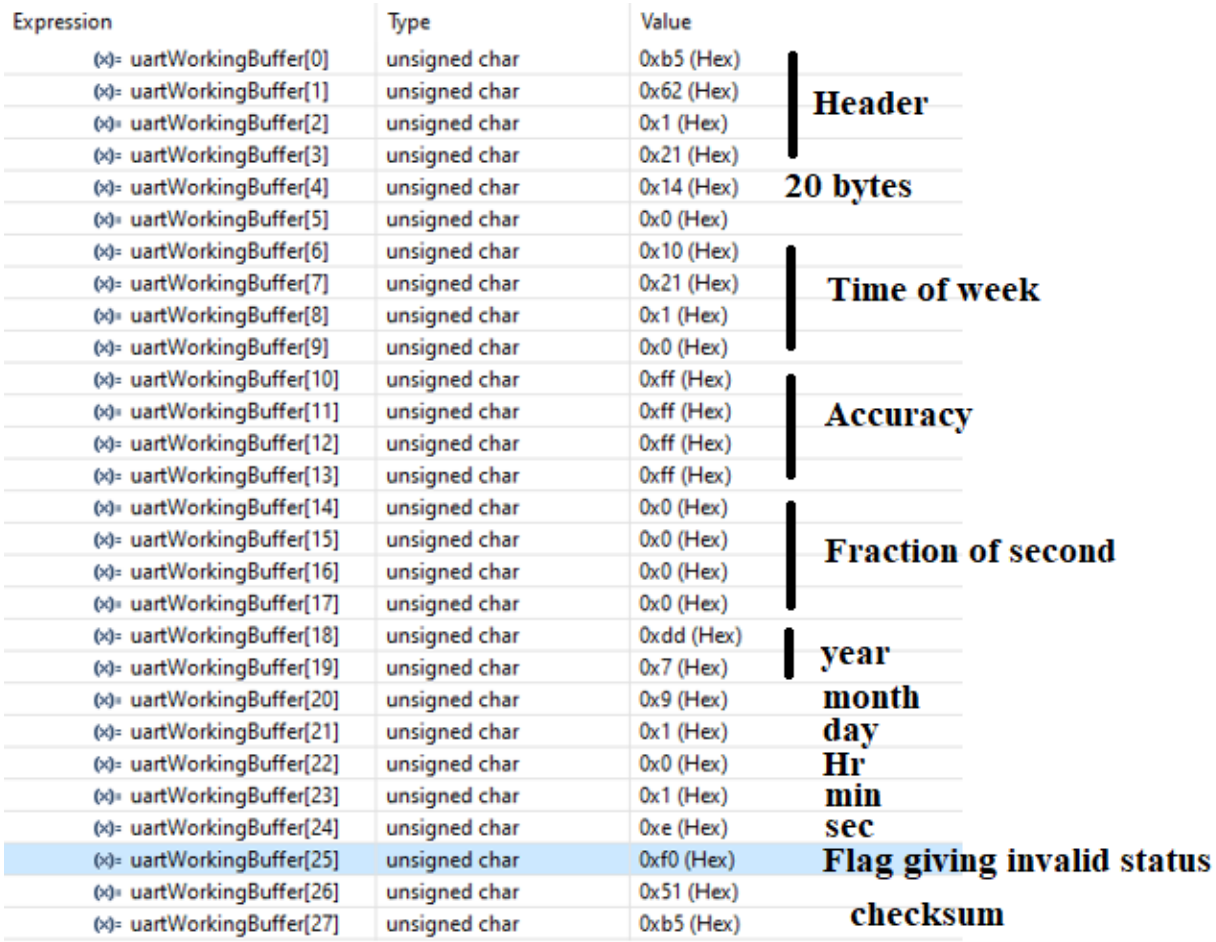
32.17.32.1 UTC time solution

Message	**UBX-NAV-TIMEUTC**					
Description	**UTC time solution**					
Firmware	Supported on: • u-blox 8 / u-blox M8 protocol versions 15, 15.01, 16, 17, 18, 19, 19.1, 19.2, 20, 20.01, 20.1, 20.2, 20.3, 22, 22.01, 23 and 23.01					
Type	Periodic/Polled					
Comment	Note that during a leap second there may be more or less than 60 seconds in a minute. See the description of leap seconds for details.					
Message Structure	Header	Class	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x01	0x21	20	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U4	-	`tAcc`	ns	Time accuracy estimate (UTC)
8	I4	-	`nano`	ns	Fraction of second, range -1e9 .. 1e9 (UTC)
12	U2	-	`year`	y	Year, range 1999..2099 (UTC)
14	U1	-	`month`	month	Month, range 1..12 (UTC)
15	U1	-	`day`	d	Day of month, range 1..31 (UTC)
16	U1	-	`hour`	h	Hour of day, range 0..23 (UTC)
17	U1	-	`min`	min	Minute of hour, range 0..59 (UTC)
18	U1	-	`sec`	s	Seconds of minute, range 0..60 (UTC)
19	X1	-	`valid`	-	Validity Flags (see graphic below)

Received:



Expression	Type	Value	
uartWorkingBuffer[0]	unsigned char	0xb5 (Hex)	
uartWorkingBuffer[1]	unsigned char	0x62 (Hex)	Header
uartWorkingBuffer[2]	unsigned char	0x1 (Hex)	
uartWorkingBuffer[3]	unsigned char	0x21 (Hex)	
uartWorkingBuffer[4]	unsigned char	0x14 (Hex)	20 bytes
uartWorkingBuffer[5]	unsigned char	0x0 (Hex)	
uartWorkingBuffer[6]	unsigned char	0x10 (Hex)	
uartWorkingBuffer[7]	unsigned char	0x21 (Hex)	Time of week
uartWorkingBuffer[8]	unsigned char	0x1 (Hex)	
uartWorkingBuffer[9]	unsigned char	0x0 (Hex)	
uartWorkingBuffer[10]	unsigned char	0xff (Hex)	
uartWorkingBuffer[11]	unsigned char	0xff (Hex)	Accuracy
uartWorkingBuffer[12]	unsigned char	0xff (Hex)	
uartWorkingBuffer[13]	unsigned char	0xff (Hex)	
uartWorkingBuffer[14]	unsigned char	0x0 (Hex)	
uartWorkingBuffer[15]	unsigned char	0x0 (Hex)	Fraction of second
uartWorkingBuffer[16]	unsigned char	0x0 (Hex)	
uartWorkingBuffer[17]	unsigned char	0x0 (Hex)	
uartWorkingBuffer[18]	unsigned char	0xdd (Hex)	
uartWorkingBuffer[19]	unsigned char	0x7 (Hex)	year
uartWorkingBuffer[20]	unsigned char	0x9 (Hex)	month
uartWorkingBuffer[21]	unsigned char	0x1 (Hex)	day
uartWorkingBuffer[22]	unsigned char	0x0 (Hex)	Hr
uartWorkingBuffer[23]	unsigned char	0x1 (Hex)	min
uartWorkingBuffer[24]	unsigned char	0xe (Hex)	sec
uartWorkingBuffer[25]	unsigned char	0xf0 (Hex)	Flag giving invalid status
uartWorkingBuffer[26]	unsigned char	0x51 (Hex)	
uartWorkingBuffer[27]	unsigned char	0xb5 (Hex)	checksum

6. **Energy power management**

Energy power management when added during initialization was not working. We then decided to scrap the energy management in the initial configuration phase and put power management in EM3 mode for In-transit and Log phase.

7. **Configuration tool incorrect target chip number**

This was probably the biggest and critical issue we faced. While programming the chip, we programmed for EFR32BG13P**6**32F512GM48 instead of EFR32BG13P**7**32F512GM48. Same family of chips work for basic blinky functionality as well as I2C but not for USART communication protocols such as SPI and UART when their ports are changed and that was the only difference between Evaluation board and custom board.After changing the chip part number. We dreaded this issue over a week and did not figure out This was a bottleneck for GPS for one week and a day before we were just able to receive some data but time was lost to work any further on GPS for demo.This was a bottleneck for FLASH for 1 day.

Every sensor contributed as a challenge as well as a new learning.

Summary: Final working status of the project

- **BME280**:
  BME280 fetches T and RH values through energy mode management and LETIMER interrupt events on custom board
- **BNO055**:
  BNO-055 fetches X,Y,Z values through energy mode management,Low power mode and LETIMER interrupt events on a custom board. Low power mode entered after 5 seconds of inactivity.
- **GPS**:
  Able to utilize the LEUART communication peripheral and is able to transmit a set of bytes of data as well as read a set of bytes of data to and from GPS. Just that data received is invalid only able to deduce that the GPC chip is not able to capture GPS data using chip antenna. But this custom board hardware is able to get data when connected to eval board and the rest of the custom project board is bypassed.
- **FLASH**:
  FLASH writes and reads values through energy mode management and LETIMER interrupt events only when abnormality is detected on the custom board.
- **Load Switch:**
  Load Switch behaves as per the expectation in the active low fashion on the custom board to turn ON/ OFF the GPS module.
- **PMIC**:
  PBSTAT is used to turn On/Off the whole microcontroller system. An On signal is sent within the first 5 seconds of boot to the PMIC.
- **Bluetooth:**
  Devices from all sensors - TRH, IMU,FLASH, and actuators are successfully displayed over bluetooth on the silicon labs bluetooth application as well as write operation to the custom board server software is successful.

Future Scope

- Put BNO-055 in GPIO interrupt-based mode wakeup. A pure EM3 modesleep and no polling would give a huge amount of saving in energy.
- PG Pin to trigger saving to memory at low battery state as well as read start address for next iteration – We would require to make use of WP to restrict access to certain portions of Flash memory.
- Adding Ambient sensor to keep track of any unwanted exposure to strong UV light or Sunlight
- Adding theft protection feature
- Adding Environment controller with monitoring which maintains the temperature and humidity inside the container
- Above can be achieved by keeping provision of updating required ranges of each parameter and selecting application through bluetooth before transportation,
- Casing for the product for it’s protection against environmental conditions.

- Develop a Mobile application for the project – easier to view error logs and user friendly.
- Product software can be updated for different applications like grocery shopping, medicines shipping, etc where contents may get spoiled with exposure to unwanted environment conditions.

Lessons learned

Rishab

1. **Time management as well as not to prolong making decisions:**
   The submissions for 2 main courses Low power and Network system coincided for major milestones and had an impact on low power timeline. Had I taken only PCB and Low power could have done a better job and completed the project without the stress.
2. **Calm mind:** It was only when I decided to keep myself calm and read the datasheet of FLASH I was able to get it to work.
3. **Innovation in testing:** Cross testing of custom board' hardware and evaluation board helped to figure out the incorrect target chip was selected for programming as both the hardware and software were correct.
4. **Positive affirmation and confidence boost:** I learnt from my project partner. There was one occasion when I had said to myself that FLASH will work today and it worked the same day. I had been wanting to get a SPI based memory to get to work since a year and got it to work without any reference code and only datasheet studies.
5. **Importance of Design review:** There can never be a design review which would go to waste. In the last design review the solder mask was removed. If we had another design review, I think we would have added a few more test points as well as made a provision to access the unused GPIO hardware pins.

Mukta

1. **Make sure that the correct chip is selected in software:** Our chip is EFR32BG13P732F512GM48 instead of EFR32BG13P632F512GM48 which caused communication issues in SPI and LEUART, maybe this is due to the errata information shared in both the chips is different and it gets loaded when chip_init() in called inside system_init() function.
2. **Selecting Sensor:** Find all data regarding software and hardware before finalizing – Struggled in GPS - Getting to know exactly how settings needs to be done earlier in stage would have lessen the hassle I had to go through at end.
3. **Check traces of signals at different points in code development:** I checked traces for messages of GPS when there was nothing left to diagnose and realized I was

actually receiving the correct data frame just that it was giving an error flag high saying data is not acquired from satellite.

4. **Utilization of extra pins:** Even though it seems unnecessary to make provisions for disconnecting/connecting different signals or pins in proto board – take out unused GPIOs on header - This would have helped to configure different pins for GPS or any other sensor in my case so as to get it tested.
5. **Learn when to stop:** It might hamper your other work - I have spent too much time stuck on development of GPS code instead if I would have stopped it early on and moved on then could have achieved better results.

References

- https://oig.hhs.gov/oei/reports/oei-04-10-00430.asp
- https://pharma-mon.com/drug-storage-monitoring/warehouse-temperature-and-humidity-monitoring-system-requirements-for-vaccine-cold-chains/
- https://extranet.who.int/pqweb/sites/default/files/documents/E003%20TPP%20for%20Humidity%20Control.pdf

Code references

- https://github.com/Rishab-Shah/ecen5823-courseproject-vido2373
- https://github.com/SimpleMethod/STM32-GNSS
- https://github.com/u-blox/ubxlib
- https://github.com/adafruit/Adafruit_BNO055
- https://github.com/BoschSensortec/BME280_driver
- https://github.com/jmichael16/LPEDT-Fall-Detection

The final project from ecen5823 - IoT class was chosen to proceed to develop the project. Oral permission from Vishnu was taken for the same and his authorship is retained in code files. The project was modified as per our needs to suit VaXinator.

Acknowledgement

We express thanks to Chris Choi (SA) for helping us get to the specific end application for our project where we wanted to explore the payload monitoring area of work. We would like to say thanks to Professor for the enormous help he provided us and solving our doubts. We express our gratitude to Lauren Darling for arranging a hands-on pre-assembly run to prepare for the assembly day. We would like to thank our SAs who helped us a lot while designing and bringing up this project. Last but not least, all of our classmates for their support.

Project Updates

Update 1 - Project Proposal	Date: 01/22/2022
Update 2 - Project Schedule	Date: 01/29/2022
Update 3 - Component Selection	Date: 02/05/2022
Update 4 - Altium Library Creation	Date: 02/12/2022
Update 5 - Schematic Creation	Date: 02/19/2022
Update 6 - Bulk Capacitance Analysis	Date: 02/26/2022
Update 7 - Basic Sensor interface Software and Component Ordering	Date: 03/12/2022
Update 8 - PCB Assembly	Date: 04/02/2022
Update 9 - PCB verification	Date: 04/09/2022
Update 10 - Application Bring-up	Date: 04/23/2022