Low Power Embedded Design Techniques

University of Colorado
Boulder

Course Project Report

Under the guidance of Prof. Timothy Scherr & Prof. Randall Spalding

Team Bhishma:
Sarthak Jain
Hardik Senjaliya
Vatsal Sheth
Date: 12/18/2019

1. Product Overview

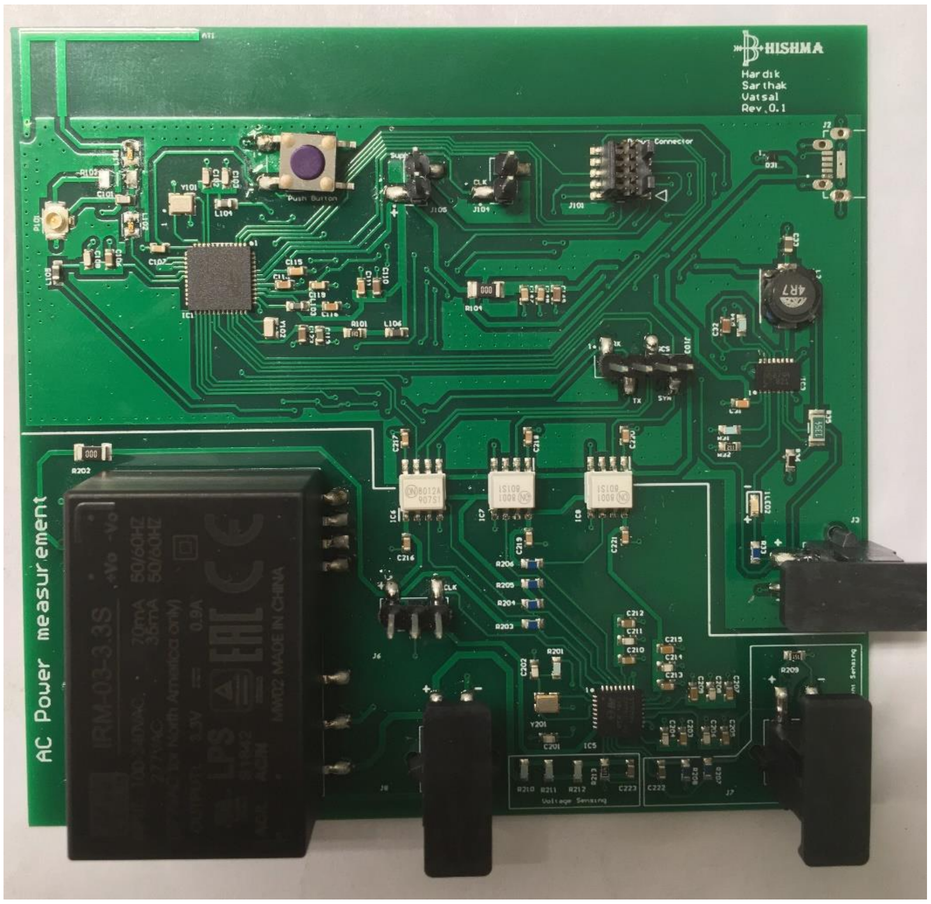


Fig.1 An image of the completed PCB for development

We propose a low-powered, IoT device for monitoring the AC voltage, current and power directly from supply points. The product taps into the AC line and neutral supplies and down-converts obtained current and voltage into a manageable DC supply.
The power-monitoring IC (henceforth referred to as STPM34) will be interfaced with main controller (Blue Gecko) using UART. All components described above will be present on the low-powered nodes. We have designed an entire mesh network, with the capability of connecting to a large number of low-power nodes, a certain number of which will have their own single friend(master) node board. The friend board is simply a Blue Gecko development radio board by Silicon Labs, with an interface to an ethernet co-processor (WIZNET W5100). The low-power nodes will update the friend node periodically, and ethernet will be used to upload real-time data periodically and immediately in case of peak load on cloud (featured as a server running on a Raspberry Pi with a TCP/IP connection). The server application will be developed for monitoring and shutting down power if required. An optional feature for OTA firmware updates will be implemented based on available time during the project. The primary power source for the low power node will be the battery, which will provide power to the Blue Gecko chip via a power management IC

(LTC3550). The friend node will need to be permanently online, and will hence be powered by USB perpetually.

Originally, we had planned for three PCBs, with each connected to the internet in a solitary environment. However, this does not support a low-power design methodology. Our initial research had led us to believe that microcontrollers supporting wi-fi could be designed with low-power in mind, but none of these controllers supported an energy profiling tool, and sufficiently low-power energy modes. We propose modifications to the original project proposal, which have been compiled with the original idea to present an overview of the complete product.

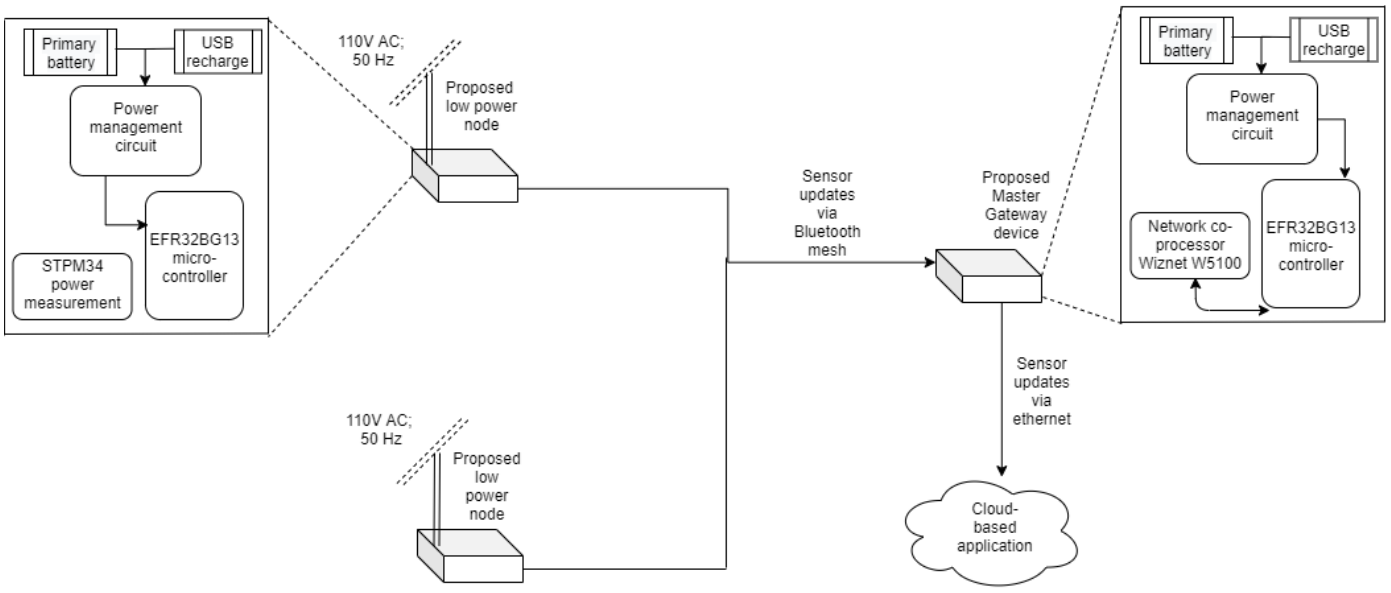


Fig.2: A block diagram of all functional components of the product & application

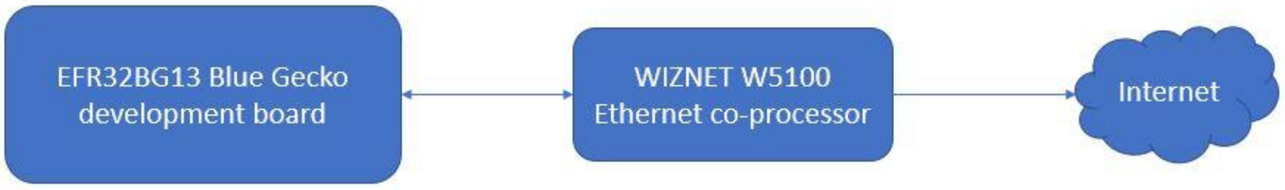

EFR32BG13 Blue Gecko development board
WIZNET W5100 Ethernet co-processor
Internet

Fig.3: A block diagram of the master node, on development board

2. Need for solution

Load Sharing in the Power grid is an important component. In a 3-phase system, load balance is important to optimize power delivery efficiency. The excess load can not only imbalance but in worst can also trip the grid. So, our product targets this market of power measurement and a real-time update on the cloud-based systems. It also allows manual decision to cut the transformer from the grid in caseload increases. As of now, the existing hardware in the product would need to be updated for installation in the industrial domain. The current hardware support monitoring of a 110V AC, 60 Hz supply line. For a higher voltage requirement, the hardware would have to support down-conversion of the updated requirements. Besides the industrial market, our product also targets the domestic and residential markets, like smart homes where the product can be used in its current state to monitor current consumption or even power consumption for wall outlet-powered devices, and use the same to automate the charging times required by devices.

3. Hardware Description

a) Block Diagram

Main objective of the Hardware in sensing part is to measure Power. There are two important quantities Voltage and current that we need to measure in order to calculate power. Once we have those, STPM34 a mixed signal IC calculates power and updates that at 8 KHz sampling rate in its internal registers. These values are then read upon request by our MCU (SiLabs Blue Gecko) via UART interface. MCU and associated circuit are powered by a 3.7 V Li-ion battery. LTC 3550 IC is used which has a buck regulator as well as the capabilities to charge a Li-ion battery. This power management section provides 3.3 V regulated output from Li-ion battery and also charges that battery via USB at 5 V, 600 mA. This device has an on-board Inverted F antenna for Bluetooth which is connected to network via Bluetooth Mesh.

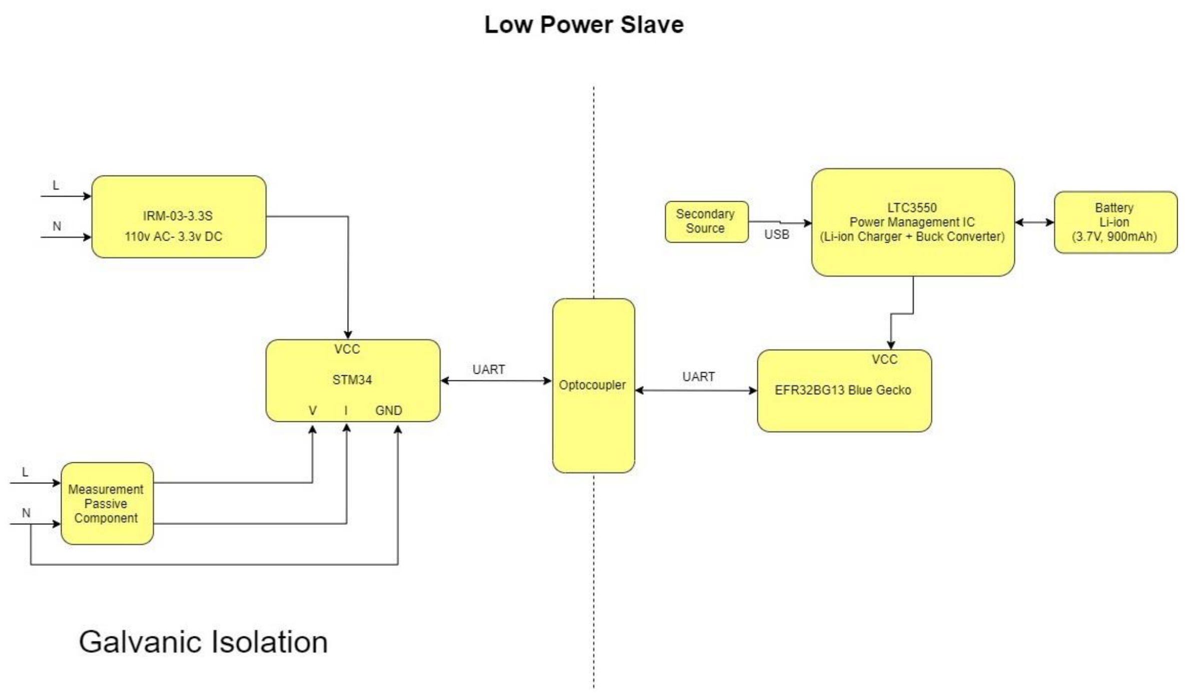


Fig.4: A block diagram of all functional components of low-power node

b) Key components

i. **EFR32BG13 Blue Gecko –** A central component of our device, it gathers data via UART from the STPM, collates it into a packet and transmits the same via the mesh network to its friend node. It is the central component which enables a low-powered state, is fed by the power management unit, and on the other end of the network, it is also a Blue Gecko which transmits the packets to the Raspberry Pi acting as server, using an ethernet co-processor. The Blue Gecko is also the chip powering the antenna, using a combination of input voltages and the internal regulators for internal generation. The main reason for selecting this component was our familiarity with the part, its ability to remain connected to a mesh network with low energy consumption, as well as the ease of programming it with a readily available development board.

ii. **STPM34 Power Measurement** – We initially were in a lot of confusion as to how to measure an actual supply. Ideas were discussed of using a transformer to down-convert, and using a regulator chip for conversion of the voltage. We eventually chose this component due to the provision it has for a current transformer, making life a lot easier as we just had to read a few registers in the STPM and connect an AC supply. These readings would be transmitted by UART to the Blue Gecko, with an array of optocouplers in between to provide isolation over the PCB, keeping the DC-powered section separate from the AC-powered section. The chip performs dual features by using a shunt resistor and voltage divider resistors for current and power measurement.

For voltage sensing, Voltage can be dropped across a large voltage divider network and measures with ADC. In our case, this is 1:1700 voltage divider which will drop 110V AC to 65mV AC.

V_{ADC_In} = 110 * (1 / 1700) = 65 mV

This will be fed to 24-bit Sigma Delta ADC with reference voltage of 1.2V and Gain is configured to 2. So Maximum Input possible in ADC without saturation is $V_{ADC\_Max.}$

$V_{ADC\_Max} = V_{ref} / (4 * A_v)$ = +- 150 mV

This configuration will accept input from voltage divider without saturation.

For current sensing we have used a current transformer coil. Current carrying wire passing through this loop will induce voltage in coil which is in ratio 1:2500. Max Input current from a residential socket is 30A. So, Max voltage at input of second ADC is +- 12 mV. We configured the Gain for Second ADC to 16. This significantly improves our SNR.

For Power measurement of AC input phase relation is important, and it is simply NOT Voltage times Current. This problem can either be fixed from Software with a Complex algorithm or we can use a mixed signal IC which has hardwired DSP Engine which does this job for us. STPM34 that we have used does exactly that. Two ADC mentioned above are with this chip itself. This chip can be interfaced with our MCU via SPI or UART. We required isolated ground to separate our two circuit and so comes the need to use Optocoupler to connect interface signal. We choose UART over SPI as it requires only two lines instead of four in case of SPI. We also had to take out two control pins SYN, SCS from this chip. Thus, we ended up having one Bi-directional Logic level optocoupler for UART and two unidirectional Logic Level optocouplers for SCS, SYN pins. Directly using Logic level optocoupler was a trade-off between price and Led driver circuit that we would have needed otherwise for optocoupler. We need power measurement only when AC supply is there. So, we used 100 V AC to 3.3 V DC switching converter directly on board which power ups the STPM34 and associated circuits. Because of this we can have our supply isolated as well.

The worst-case time for STPM34 depends on the UART bus itself, and the baud rate we decide to use. By default, the power measurement IC has a baud rate of 9600 and is supported by the LEUART mode of EFR32. As per datasheet of metering IC, the max data packet in one transaction is 5 bytes, giving a transmission time of roughly 5 ms.

iii. **LTC3550 Power Management** – The main reason for selection of this component was the dual purpose it serves, that of regulating input voltage as well as charging the battery itself. The chip requires a battery input (from either the battery of externally via USB), regulates the same using its internal DC-DC converter, acting as a buck and supplies the required output to the radio chip. An extra feature of using this IC is it handles the switching between USB supply and battery discharging due to its internal circuitry, namely a MOSFET.

c) Other components

i. **WIZNET W5100** – Network Co-processor with Ethernet PHY + MAC + TCP/IP Stack + SPI Interface selected because of low cost, low power SPI interface, and coexistence of ethernet PHY layer with TCP/IP stack. Only microcontrollers were found as an alternative, which would have added complexity.

ii. **IRM-03-3.3S** – AC-DC power supply converter: IRM-03-3.3 MeanWell's IC is an easy way to provide regulated DC power supply for powering the power measurement IC, also allowing us to maintain a distinct area on the board for isolated and non-isolated components.

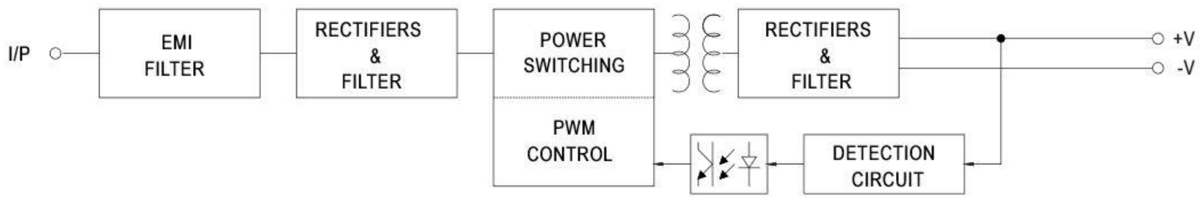


Fig.5: Internal circuitry and isolation for IRM-03-3.3S

iii. **Reset Circuit Description –** We will be using a Push button directly connected to the RESET pin to reset the Blue Gecko board. Reset is an active low signal on the device, pulled up to AVDD. The device has an internal power-on reset circuit. So only a push-button to drive the reset pin low will work. The same button will also be tied to the RESET pin of Wiznet 5100. As there is no possible use case where we will have to reset only one of these devices and hence, we can use a common reset signal. We will not be having anything to reset the STPM34 because to reset it we need an expensive push button which prevents a user from the danger of working with direct AC power. Hence, it will be reset by a power cycle.

iv. **Clock circuit generation description –**

1. High-frequency crystal 38.4MHz for BG: This crystal is the source for high-frequency RF clock. This clock will be turned off during EM2, EM3, and EM4 low power energy modes.
2. Low-frequency Crystal 32.768KHz for BG: This crystal is the source for low-frequency clock which will be in operation all energy modes. Effectively, this is the only clock running during low energy mode. This will drive our LEUART(low energy UART) which keeps our communication working during low energy mode.
3. 16MHz crystal oscillator for STPM.

4. Software Description

In our application, communication between Low Power Nodes and Master Gateway Node will be achieved by Bluetooth mesh network communication. Each slave device will be attached to a wall plug to measure the power. The power measurement IC will measure the consumed power and the measured value will be stored on the device. The power measurement will be carried out every 10 seconds (configurable using APIs). The stored data will be sent to the cloud for monitoring and ethernet will be used to send the data over the cloud. A TCP/IP socket-based server will be used to achieve the same. Updates take place constantly as the Gateway receives data. The Gateway is a device having access to both, Bluetooth and Ethernet. It always listens for the connection from the low power nodes which wake up every 10 seconds to read the data from the power measurement circuit.

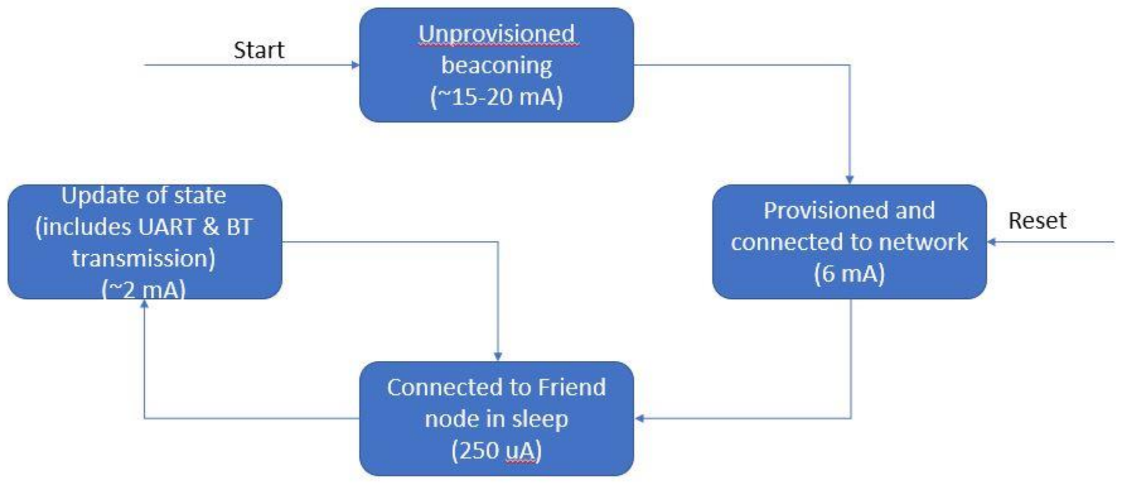

Start
Unprovisioned beaconing (~15-20 mA)
Provisioned and connected to network (6 mA)
Reset
Connected to Friend node in sleep (250 uA)
Update of state (includes UART & BT transmission) (~2 mA)

Fig.6: A flow diagram of software modes and stages

5. Features

- Energy profiling on the cloud to monitor power usage at the grid-level.
- Internet connectivity via ethernet module, enabling remote logging of data parameters.
- Low power modes enabling minimum average power consumption.
- Python application on the web to monitor and issue control commands, as part of future scope.
- Ultralife 900 mAh battery with 10 months battery recharge time.
- Utilize AC power for power measurement sections.
- Adjustable data-recording interval.
- Control over which parameter to obtain using abstracted APIs.

a) **List of APIs** – We have a small set of APIs for interacting with our product to vary features which we thought would be most useful for a customer. The variable features are the interval over which to take readings (currently configured at 10 seconds), and the parameters which the customer needs (voltage, current or power). The first feature can be accessed by changing the second parameter of the lpn_init function call. APIs named GetCurrent, GetVoltage and GetPower exist for the parameters.

b) **Programmability of Soc –** The product includes a MINI connector as our programming and debug interface. This is a proprietary connector from Silicon Labs where programming and debug is done through SWD(Serial Wire Debug). This connector has a reset pin that will drive reset on the Blue Gecko. This interface also provides the feature of Virtual COM port and AEM(Advance Energy Mode). Virtual COM port is useful to add print statements in firmware for debug purposes. AEM is useful to measure the actual current consumption and to verify our assumptions. No external energy source will be required to program the MCU.

c) **Current Profile over Expected Use case** –
We have decided on three energy modes: Energy level 3(deep sleep), Energy level 2(hibernate) and energy level 1(active), as detailed below:

- Energy level 3(Deep Sleep mode) - The low power device will be in this mode for the majority of the time. The processor has a low current consumption of around 250µA, during which the power measurement IC will be enabled low. We did consider using CRYOTIMER to create a timer interrupt every 10 seconds and take sensor readings, enabling the power measurement IC high, but instead used a software timer by Bluetooth mesh.

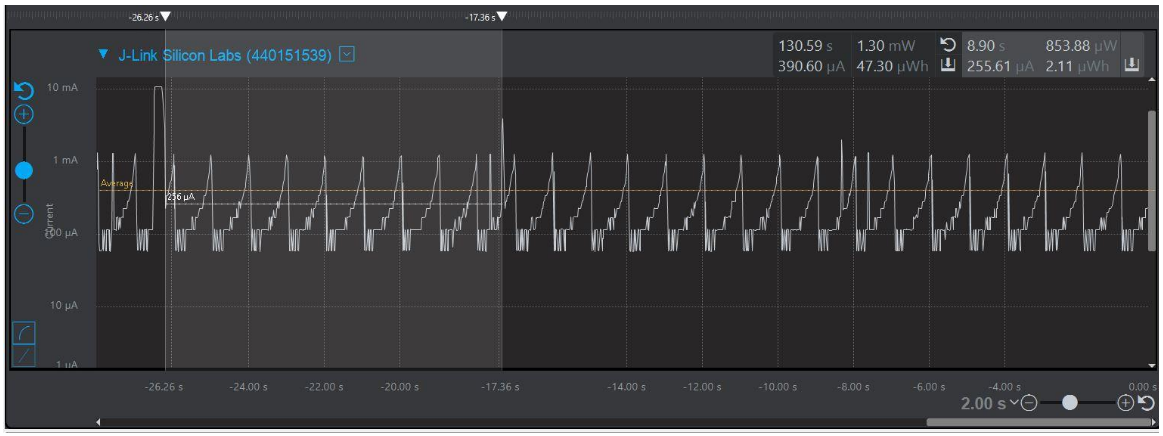


Fig.7: Current profile of device when in Energy Level 3

- Energy level 2(Hibernate Mode) - The low power device will be in this mode every 10 seconds, as it wakes up, takes a reading and sends the same to the Friend node. The current consumption for the processor in this mode is around 3-4 mA. The power measurement IC itself would consume around 4.3 mA for taking a reading, and communication between the two over UART would cost 3.5mA. We intend to keep the device in this mode for less than 2 ms.
- Energy level 1(Active mode) - The device will be in this mode only while attempting to connect to the mesh network and establish friendship with the friend node. This stage will appear once every reset. Considering a maximum transmission power, the average current in this mode will be around 24mA for a time period of a few seconds. The processor itself has a current consumption of around 4.7mA, with an instantaneous spike of over 20mA at the time of connection establishment.

The use case peak current can hence be seen as 80 mA, which comes to a total recharge time requirement of 20 months, seeing as the battery has a 10%discharge rate per months.
The energy modes and calculations listed above are all for the low-powered slave device. In the case of the master device, there will be some slight changes, as the master board does not feature a power measurement IC.

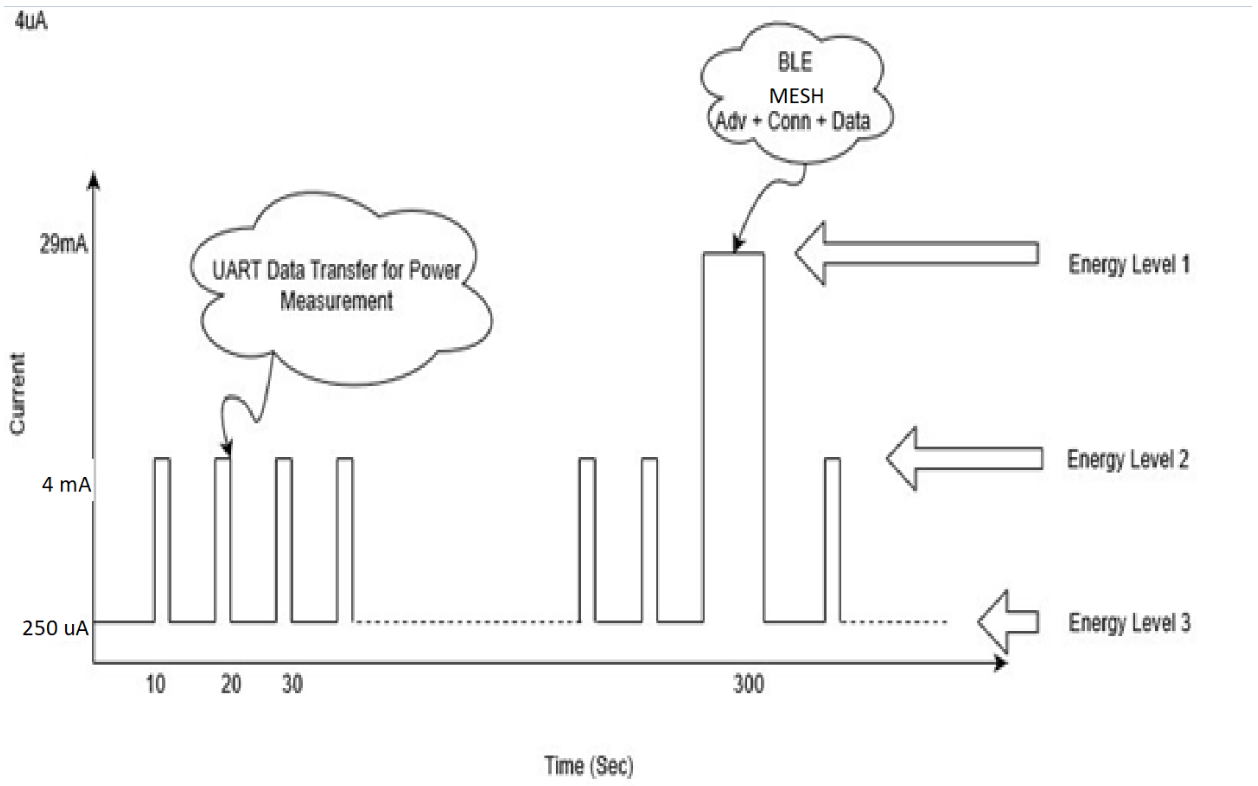


Fig.8: Use Case Diagram of energy modes

6. Specifications

- Dimensions: 88mm x 90mm
- Battery: 1 Li-ion rechargeable battery, 3.7V, 900mAh
- Wireless range: 300m
- DC Voltage Range: 2.7 - 5 V input
- AC Voltage parameter: 110 V AC

a) **Energy Storage Element Selected** – We have selected a rechargeable Li-ion battery with a capacity of 900mAh and a nominal output voltage of 3.7V. The average DC current as calculated above is 0.0627mA in 1 hour. Based on that value and battery capacity we have device life of 599 days ~ 20 months. As of now, we are not using energy harvesting for recharging the battery, but a USB rechargeable IC to recharge the battery.

Current in EM1: 4.7mA(±10%) + 20mA(instantaneous) + 24mA(average transmission current)
Current in EM2: 4.3mA [power measurement IC](±10%) + 3.5mA(Communication over UART)(±10%) + 3.3µA(processor's current consumption)
Current in EM3: 3µA + 1µA

Average current over a period of 1 hour =

[{ (24+4.7) * 600ms * (12 times in 1 hour) } +
{ (4.3+3.5+0.0033) * 1.67ms * (360 times in 1 hour) } +
{ (0.004) * 3600 seconds }] / 3600
= (206.640 + 4.691 + 14.4) / 3600
= 0.0627 mAh

i. **Peak discharge rate** of battery = 1C = 900mA in 1 hour
ii. The **lowest nominal voltage** out of battery is 3V. Accordingly, we have designed the battery cut-off voltage of the circuit also as 3V
iii. **C-rate of specified battery** is (discharge current/battery capacity)
= (0.0627mA/900mAh)
= 0.00006967
= 0.0696 mC

b) **Power Management and Bulk Capacitance Selected** – We have used AC supply to power part of our circuit, namely the power measurement circuit. An AC-DC converter brings supply 110V AC down to either 3.3V or 5V, as per user choice and requirement. This regulated DC voltage is used to supply the power measurement circuit. The UART interface of the power measurement circuit, connected to the Blue Gecko's GPIO, provides isolation by using an optocoupler in between.
We are also using a battery recharging IC, LTC3550, to recharge the battery via USB. We cannot use the AC supply to do so since the whole point of isolation would be lost in that case.

All ICs in our design operate at **3.3V Vcc**. So, we have an input voltage range of 3.7V - 3.3V. Output current requirement as mentioned above with safety margin is taken at 0.2A. With this input and output specification, we generated a power management circuit using the TI Webench tool. From the suggested designs, we choose the one with part number **LTC3550**. This is a buck DC-DC converter. We require a buck converter so that we can maintain 3.3V output, as long as the battery is supplying above 3.3V. Once the battery voltage drops below this level, we expect our recharging circuit to kick in. This helps in extracting maximum capacity from the battery. This device also provides a programmable output voltage using external resistor divider. This gives us the room to maneuver and also fix the part number. Also, this design is relatively simple in terms of components as only a bypass capacitor for input, output and external resistor bridge to program the output voltage is required.

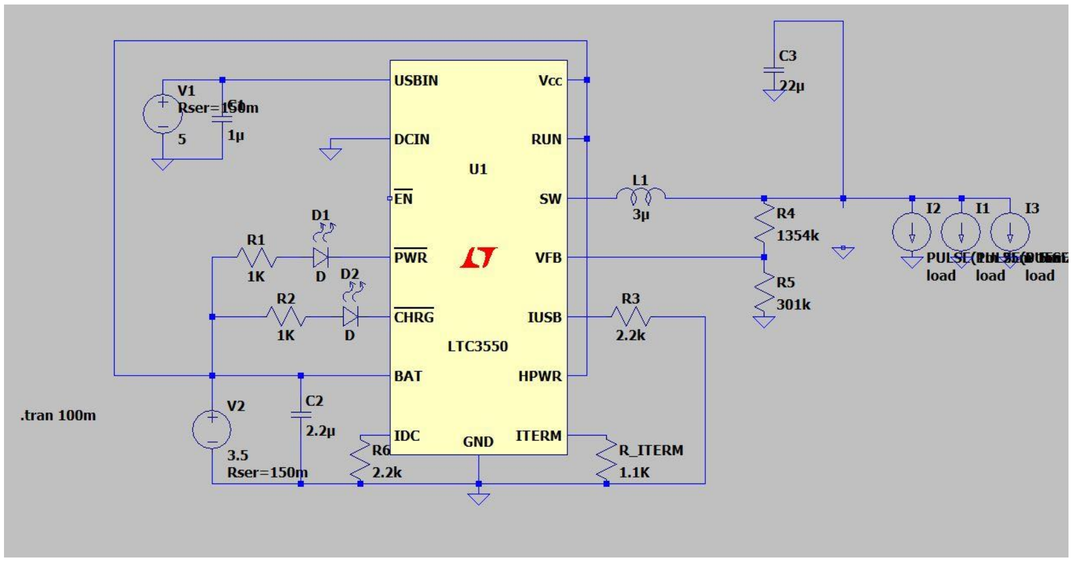


Fig.9 A Circuit diagram of our power management unit

The diagram above has been referred from the device datasheet and is a generic use-case application model. The actual resistance and inductance values have been calculated and represented in the bulk capacitor update section.

i. **Bulk Capacitance Selection and Calculation**

We have considered a worst-case simulation of battery voltage dipping to 3.5V, with charging circuit disabled and peak current requirement of 300mA, which is extremely unlikely. We arrived at the 300mA requirement by considering all the worst-case current consumption of each energy mode, combined, which is very unlikely, as the math actually reflects a worst-case current consumption of less than 250mA.

We have chosen a bulk capacitance value of 22uF, with an input capacitance of 1uF.

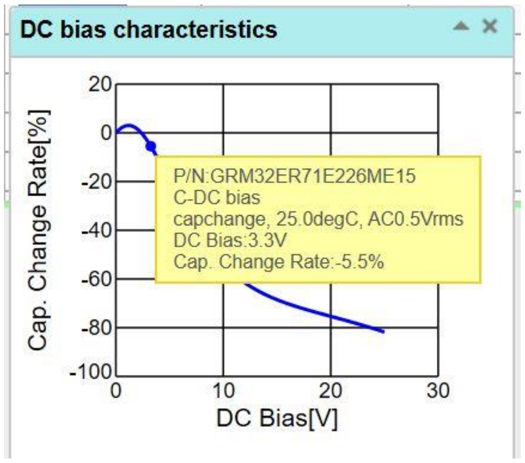
According to the datasheet, the relation between the input voltage and bulk capacitance is given by:

$$C_{IN} \text{ required } I_{RMS} \cong I_{OMAX} \frac{\sqrt{V_{OUT}(V_{CC} - V_{OUT})}}{V_{CC}} \quad (2)$$

According to the datasheet, the relation between the output voltage and bulk capacitance is given by:

$$\Delta V_{OUT} \cong \Delta I_L \left(ESR + \frac{1}{8fC_{OUT}} \right)$$

We have considered a voltage swing of 0.2V. Our required output voltage is 3.3V, and the maximum cutoff voltage for our ICs is given by Wiznet W5100, like 3V. For a safety margin, we have considered 3.1V.
Switching frequency = 1.5MHz
The value of delta load current came as 0.6mA for a pulse current of 300mA.
Factoring all these values in, we get an output capacitance of 20uF and an input capacitance of 1uF. For these parts, ESR = .00456728 ohm.
We have selected a chip monolithic ceramic capacitor, GRM32ER71E226ME15 from Murata. Its specifications are 22µF ±20%, 25V, Ceramic Capacitor, X7R, 1210 (3225 Metric).



P/N:GRM32ER71E226ME15
C-DC bias
capchange, 25.0degC, AC0.5Vrms
DC Bias:3.3V
Cap. Change Rate:-5.5%

Fig.10: DC bias characteristics of our selected capacitor

According to the image above, it has a capacitance of 20.79uF at DC bias voltage of 3.3V, which is exactly the capacitance required by our buck circuit to maintain voltage at 3.3V, with a very slight dip of 12mV for worst-case current requirement of 80mA, on the slave device. On the master device, the maximum current is 300mA being drawn by the ethernet processor. In this case, the circuit allows a dip of 400mV, which still leads to an output voltage of 3.1mV, which is sufficient to drive all our ICs. We have selected a capacitance with such a large value since we wanted to keep parts common across the master and slave boards.

7. Testing and Verification

a) Test points

For Supply –

- Buck Output 3.3 V
- Battery Input
- USB
- GND

For Blue Gecko –

- Rx, Tx, and GND for UART
- HF XTAL and LF XTAL for clock output
- CLK, CS, MOSI, MISO, GND for SPI

For STPM34 –

- VCC
- GND
- Clock output
- Rx and Tx for UART

We had sufficient test points, but we could definitely have used more LEDs on board or indications.

b) Verification of Current Use Case

Maximum peak current during bluetooth advertising = 28 mA; Duration of peak = 10ms
Maximum peak current during connection pings = 15 mA; Duration of peak = 2.5 ms
Maximum peak current during transmission = 24 mA; Duration of peak = 5 ms

For a single connection event, we observed a wide range of advertisements. On average, we could consider 15 advertisements for connection to establish.
To maintain a connection for 5 minutes, we considered a connection interval of 4 seconds, and slave latency of 32 seconds, implying a connection ping every 32 seconds. In 5 minutes, that comes to 10 pings, each with the timing of 2.5 ms, equaling 25 ms.
For a period of 5 minutes, transmission occurs once, implying 5 ms and 24 mA.
This comes to an average of:

Average current = Total current / Total time taken
= [(28*10*15) + (15*10*2.5) + (5*24)]/(150 + 25 + 5)

= 26mA,

Which is less than our original assumption of 30mA.

c) Verification of Power supply voltages

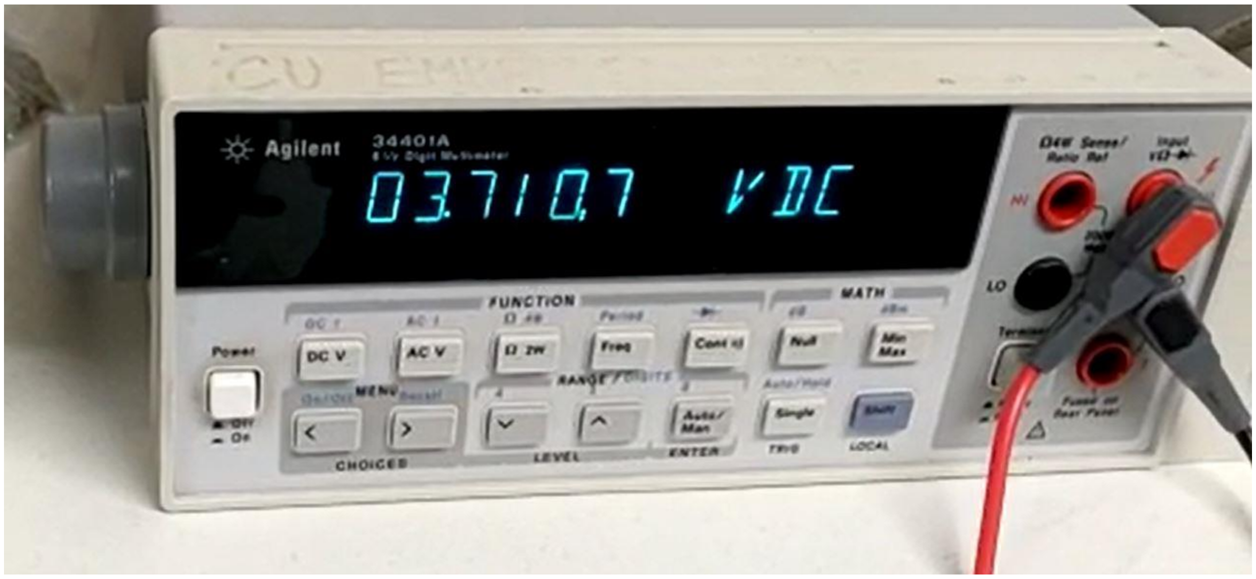


Fig.11: Multimeter snapshot of battery being charged beyond 3.7V

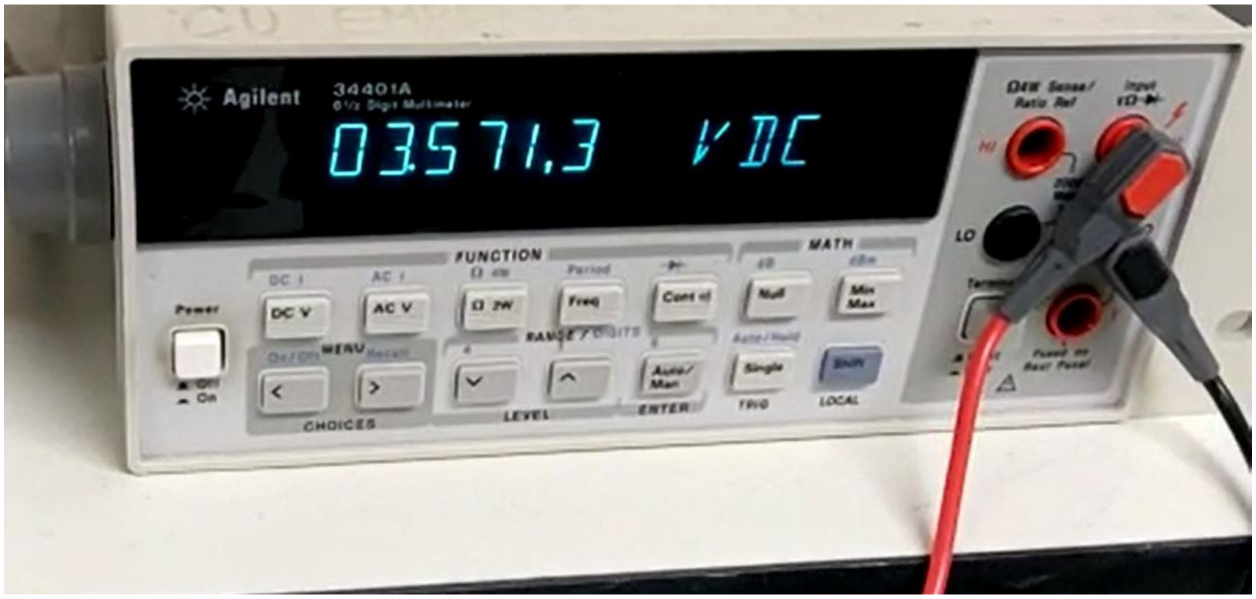


Fig.12: Multimeter snapshot of battery voltage supply on PCB

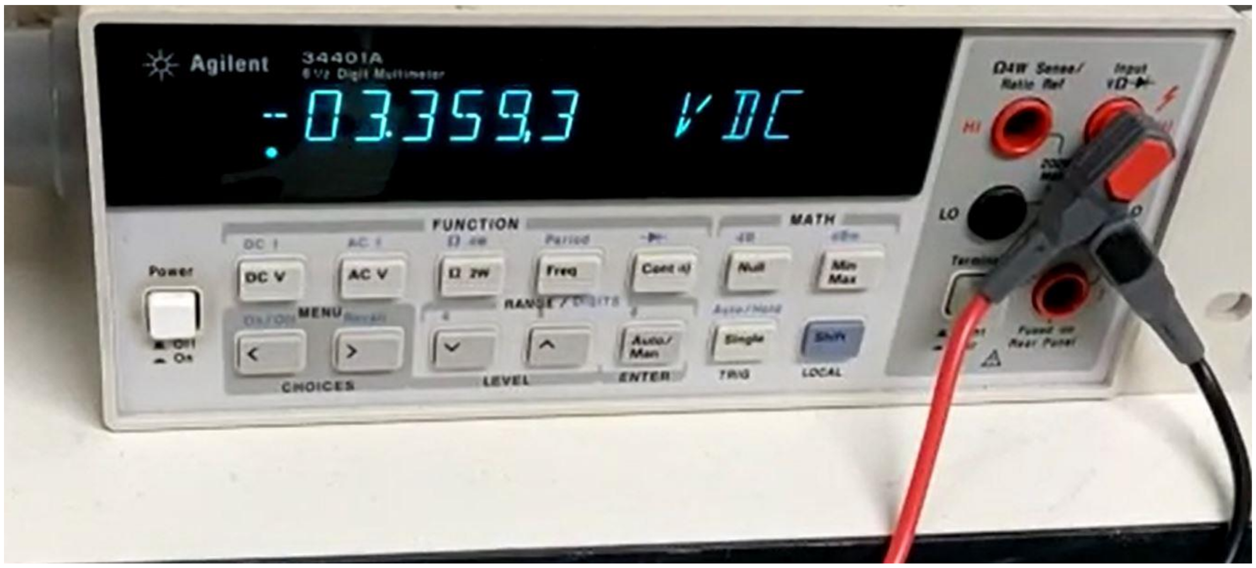
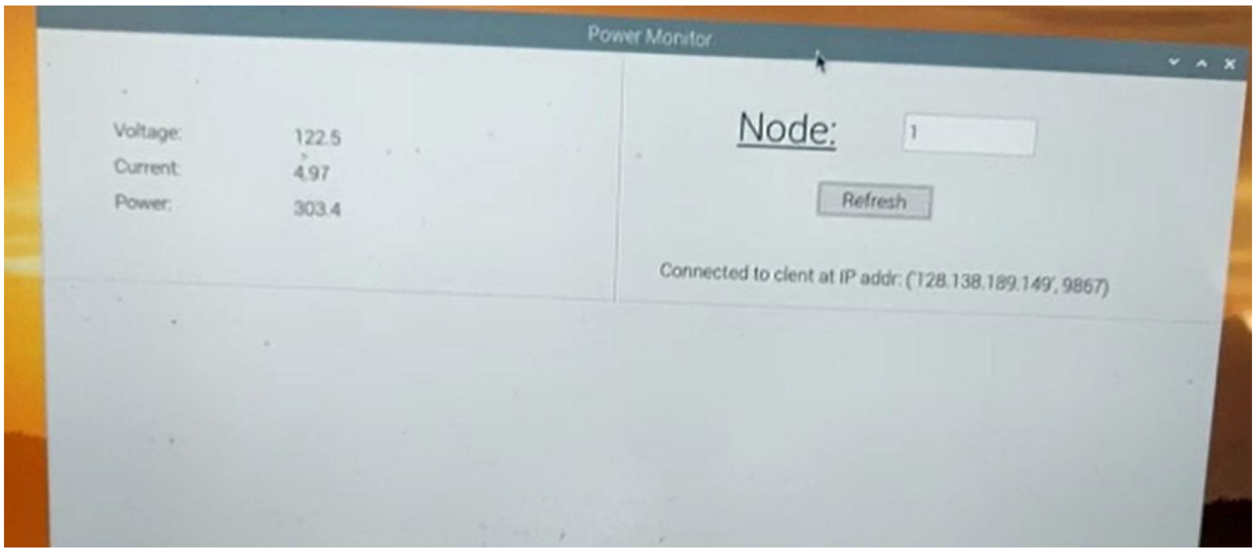


Fig.13: Multimeter snapshot of IC supply at 3.3V

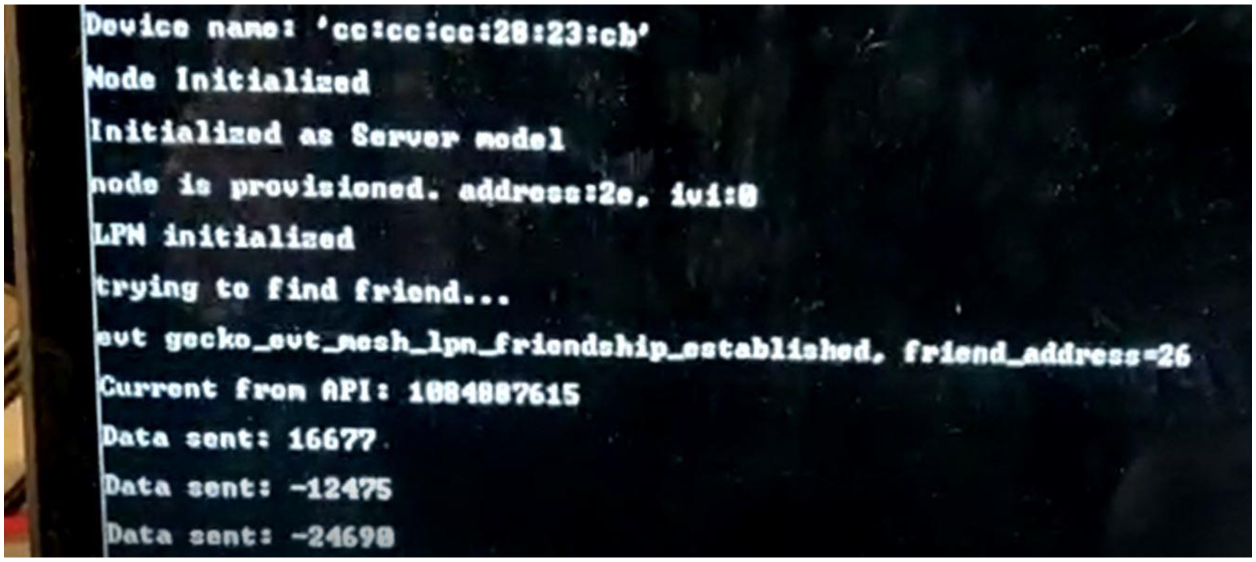


Voltage: 122.5
Current: 4.97
Power: 303.4

Node: 1

Connected to clent at IP addr: ('128.138.189.149', 9867)

Fig.14: GUI snapshot of connection IP x:x:x:x with voltage, current and power readings



```
Device name: 'cc:cc:cc:28:23:cb'
Node Initialized
Initialized as Server model
node is provisioned. address:2e, ivi:0
LPN initialized
trying to find friend...
evt gecko_evt_mesh_lpn_friendship_established, friend_address=26
Current from API: 1084887615
Data sent: 16677
Data sent: -12475
Data sent: -24698
```

Fig.15: Friendship established between friend node and LPN node

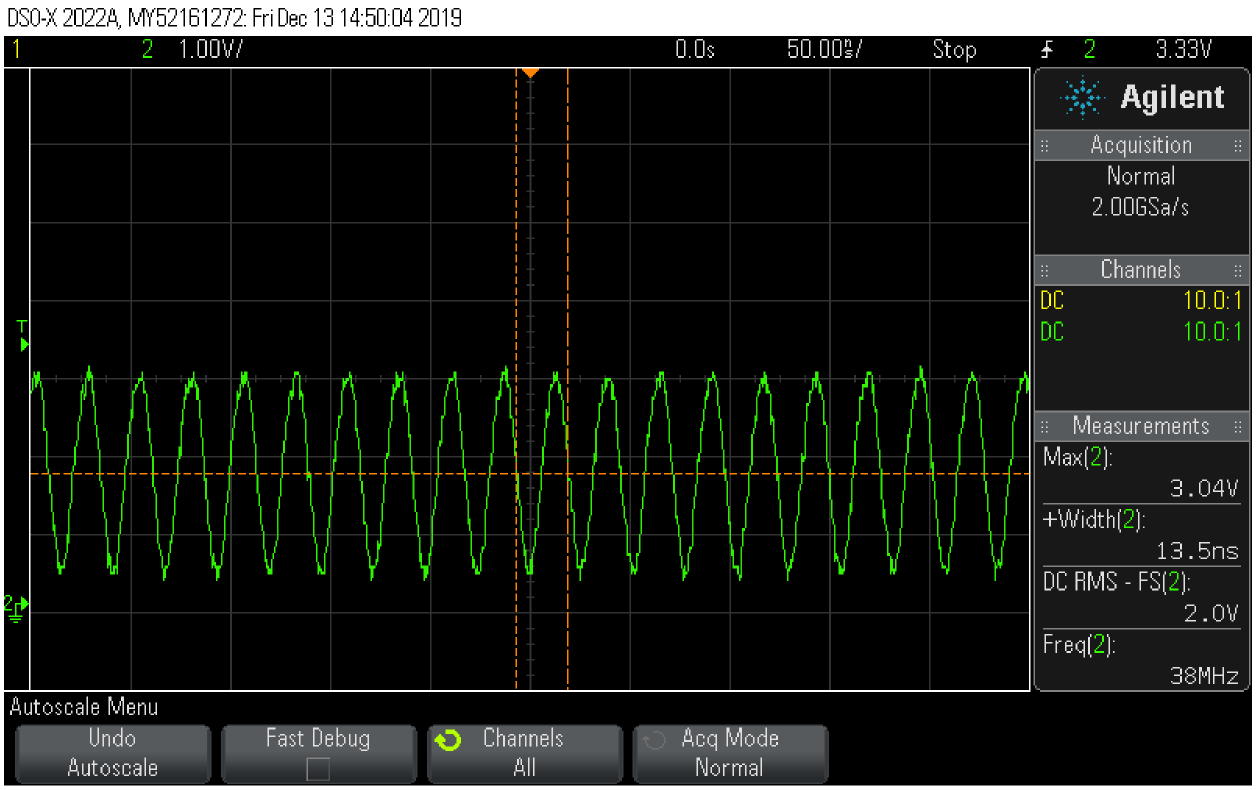


Fig.16: Scope shot of clock output from HFXO; 38MHz

Sr.No	To be verified	Definition of Passing	Date test performe d	Tested By	Measured Result	Passed
1	Verify Power Supply voltage at AC-DC Converter Output	3.3V measured voltage at output pin	11/25	Vatsal	3.3V measured voltage at output pin	Y
2	Verify Power Supply voltage at Power Management IC Output	3.3V measured voltage at output pin	11/25	Vatsal	3.3V measured voltage at output pin	Y
3	Verify Signal Quality for UART Comm. Bus	Tap the bus on logic analyzer and verify it using protocol analyzer feature	12/1	Vatsal	Screenshot attached in below section	Y
4	Verify battery Charging	Discharge battery using resistive load and then provide USB input to charge. Charging LED should light up when the battery is being charged	11/26	Sarthak	Charging LED lights up	Y

5	Verify System Power good booting	Replace the battery with a Lab DC source. Drop the voltage so that MCU goes in POR state. Check the clock output test point. Increase the DC supply and check whether clock starts at expected input voltage	11/25	Sarthak	Clocks start at 2V	Y
6	Verify Power measurement Sensor	NA (It can be checked only when the entire firmware is running)	12/5	Vatsal	Screenshot attached in below section	Y
7	Verify BLE can transmit	Test against Silicon Labs Mobile application	11/23	Hardik	BLE transmitting	Y
8	Verify BLE can receive	Test Against SiLabs Mobile Application by coding our device to echo back whatever it receives	11/23	Hardik	BLE receiving	Y

9	Verify USB can charge the battery	Check 5V output at USB connector and light up of charging LED	12/3	Hardik	LED lights up, battery charging visible on multimeter	Y
10	Verify that MCU and power measurement IC is functional	Check the clock out test point for both the IC to verify that they are working	12/1	Sarthak	Clock outs can be seen on oscilloscope	Y
11	Verification of firmware routines.	Use Virtual Comm Port connection on debug port. Use printf to send register read and other required data on terminal.	12/5	Hardik	Print statements visible on terminal	Y

8. Signal Analysis

Below are the few snapshots from Logic Analyzer, showing Communication packets transaction between STPM34 and MCU via UART. We have also set up UART interpreter on Logic Analyzer which makes it easy to directly see the data bytes going on.

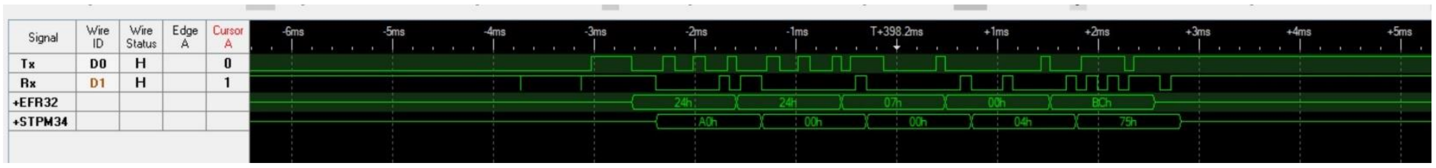


Fig.16: 5-Byte packet structure

STPM34 has a 5 bytes packet structure where last byte is for CRC. In this first packet I am writing to a register in STPM34 to disable CRC. We don’t need CRC as track length is short and we have sufficient pull-up resistor and don’t expect much noise. CRC Value for first packet is manually calculated and hardcoded in initialization routine.

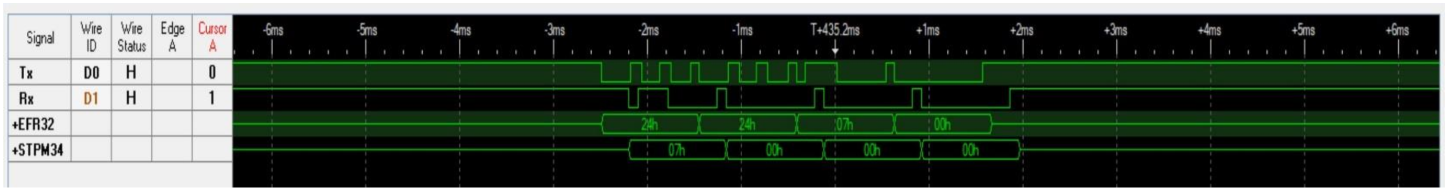
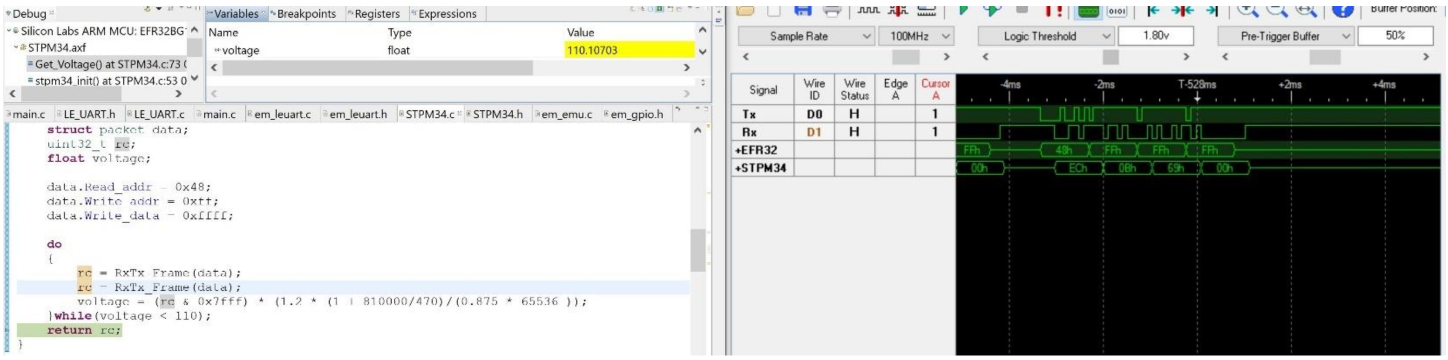


Fig.17: 4-Byte packet structure with CRC disabled

As you can see in this snapshot, CRC is disabled and now the packet size is reduced to 4 bytes.



Debug
Silicon Labs ARM MCU: EFR32BG
STPM34.axf
Get_Voltage() at STPM34.c:73
stpm34_init() at STPM34.c:53
Variables
Breakpoints
Registers
Expressions
main.c LE_UART.h LE_UART.c main.c em_leuart.c em_leuart.h STPM34.c STPM34.h em_emu.c em_gpio.h
Sample Rate 100MHz
Logic Threshold 1.80v
Pre-Trigger Buffer 50%
T-528ms
FFh 48h FFh FFh FFh
00h ECh 0Bh 69h 00h

Name	Type	Value
voltage	float	110.10703

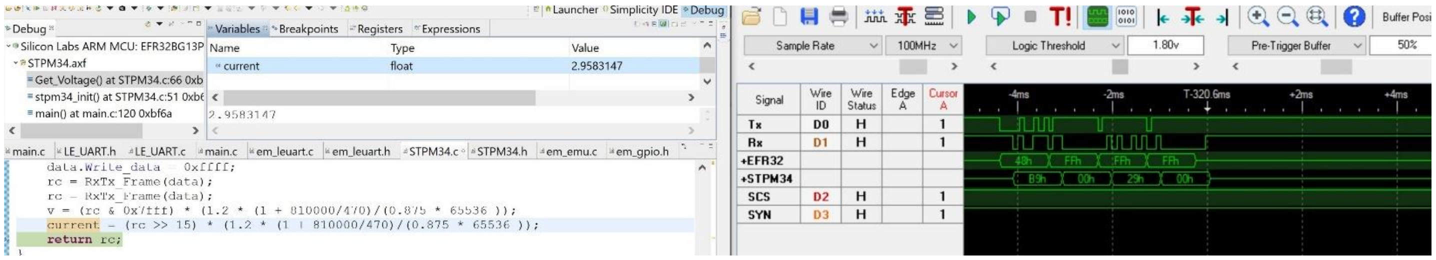
```
    struct packet data;
    uint32_t rc;
    float voltage;

    data.Read_addr = 0x48;
    data.Write_addr = 0xff;
    data.Write_data = 0xffff;

    do
    {
        rc = RxTx_Frame(data);
        rc = RxTx_Frame(data);
        voltage = (rc & 0x7fff) * (1.2 * (1 + 810000/470)/(0.875 * 65536 ));
    }while(voltage < 110);
    return rc;
}
```

Fig.18: Voltage packet

This snapshot shot shows the read for RMS Voltage being sent and its reply by STPM34. In the watch window of IDE we can see the computed Voltage value in variable.



Name	Type	Value
current	float	2.9583147

```
2.9583147
```

```
    data.Write_data = 0xffff;
    rc = RxTx_Frame(data);
    rc = RxTx_Frame(data);
    v = (rc & 0x7fff) * (1.2 * (1 + 810000/470)/(0.875 * 65536 ));
    current = (rc >> 15) * (1.2 * (1 + 810000/470)/(0.875 * 65536 ));
    return rc;
}
```

Signal	Wire ID	Wire Status	Edge A	Cursor A
Tx	D0	H		1
Rx	D1	H		1
+EFR32				
+STPM34				
SCS	D2	H		1
SYN	D3	H		1

Fig.19: Current packet

This snapshot shot shows the read for RMS Current being sent and its reply by STPM34. In the watch window of IDE we can see the computed Voltage value in variable.

9. Difficulties Encountered

a) **Hardware –** Initially, the microcontroller refused to be programmed. We would continuously face an error: "Target voltage too low: 0.62V", which we found weird since when we took a multimeter to the MCU's pins, we could see the appropriate voltage levels, all correct. Eventually, Prof. Spalding pointed out to us one day in the lab that our debug connector had some dry solder at the joints. Upon cleaning it up, we could program the MCU perfectly.

b) **Hardware –** We found it quite difficult to solder the PMIC, since it only has pads on two sides, and hence it was difficult to verify during placement if the IC had been placed on correctly. (It was easier verifying the same with ICs having pads on all 4 sides.) Even after reflowing our first iteration, the PMIC refused to work. Turned out the solder paste had not adhered completely to the pads on one side, leaving the IC slightly raised on one side.

c) **Design –** Lack of LED placement on the board. We did not want to unnecessarily raise costs. We went on to realize that it is sometimes worthwhile to dish out during development. We had only one LED placed on-board, to indicate charging status of the battery. We could have used a lot more around the Blue Gecko.

d) **Design –** During battery selection, we were considering a 4.2V battery. This actually complimented the PMIC quite well, since whenever the battery charging voltage on the PMIC's charging pin crossed 4V, it would turn off an LED, indicating the battery was fully charged.Else while charging the battery, LED would remain on. We found this feature quite useful during component selection. This value of '4V' was an inbuilt feature, not subject to the developer's whims.
While component ordering, we eventually ended up ordering a 3.7V battery, due to reasons long forgotten. This had the effect of ALWAYS keeping the charging LED on, since voltage on the charging pin never crossed 4V, making the one LED we had kept on-board, effectively useless.

We were eventually able to find a workaround in software. The Gecko maintains a useful feature of monitoring supply voltage. We were able to use this supply voltage to estimate the battery voltage, allowing us to determine when the battery was completely charged.

e) **Design** – Similar to the point above, we also did not think of any way to tell the user when the battery level is running low, for some confounding reason, we completely glossed over this during development. Again, software came to the rescue, allowing us to monitor when the battery required charging based on supply voltage at the Gecko's input pins.

f) **Schematic & Layout Design** – The antenna matching section of the Gecko requires a 1.9nH inductor. Altium's library loader provided us with a 0603 sized footprint for the same inductor. However, the component actually has a 0201 (0603 metric) footprint. Incidentally, a 0603 sized component with 1.9nH inductance does not exist.
Eventually we had to use a 0402 sized component, placing one pad of the component on the footprint's pad, while joining the other pad on the footprint to the inductor with a lot of solder in the intervening space.

10. Summary of Functionality

We have two isolated circuits on our board. One is powered by 100 V AC which is converted to 3.3 V DC to supply this part of circuit. It measures Power on AC line and updates its internal registers. These are then read by the second circuit via UART. This interface is isolated via optocoupler. This part of circuit is powered by Li-ion Battery. Power is managed by an IC which has a buck regulator and a battery charger via USB. MCU used here is SiLabs Blue Gecko which has a BLE Mesh stack implemented on it. This node sends the reading to Friend Node in BLE Mesh which is implemented on Blue Gecko Development Kit. This is interfaced to a WIZNET 5110 module for ethernet. Thus, all over Low Power nodes are connected to one friend node via BLE Mesh. Friend Node is then connected to internet via Ethernet on a TCP Socket Server implemented on RPi. RPi also hosts a GUI developed on PyQt providing User friendly interface to access this collected data.

11. Lessons Learned

i. Sarthak

- The correct way of applying solder paste on the PCB, that is to drag the card across in one swipe, keeping it as flat as possible, and making sure no section gets overlapped. I faced the repercussion of doing it the other way when I had to redo solder pasting a couple of times.
- Crystal Oscillators do not have any correct orientation in which they must be placed, as long as they are placed in the correct orientation, facing upwards.
- Hand soldering and rework of ICs with flux using the large-scale microscope in the ITLL with the small wire.
- Even plastic connectors can be pushed into the reflow oven!
- That hardware does not need to be perfect, and it never will be. It is sufficient to leave the hardware as it "just works". Although taught during the lecture, it is something I faced practically.

ii. Hardik

- Even though we have QFN packages, it is a lot easier to solder ICs by hand with pins of all 4 sides rather than pins on just 2 sides, as it is difficult realizing the offset on the vertical sides.
- It is better to use libraries provided by manufacturer wen working with the network stack with TCP like in our case.
- Although tedious, it might be advisable to create our own footprints of components, as Altium's Library Loader messed up the footprint of one our own inductors.
- If you repetitively miss your set weekly deadlines, you should catch up by giving in a little extra one week.
- You can manage to solder a 0402-package component on a 0603 footprint!

iii. Vatsal

- The hot gun can be used to melt solder and rework it without having to completely remove the component.
- We learned of a new phenomenon, the Tombstone effect, which causes a component to stand up on one side if it hasn't perfectly caught the solder paste on both pads.
- Layout is an extremely tedious procedure, and cannot be completed in one night. It requires a lot of iterations, which we realized as we missed a couple of deadlines during layout submissions.
- UCB Wireless has multiple subnets, which makes it difficult to have a server and client using a socket subsystem on the same network, if they don't have the same subnet. This problem prevented us from giving a demo during class.
- AC can lead to problems, explosive ones!

12. WOAH! Factor

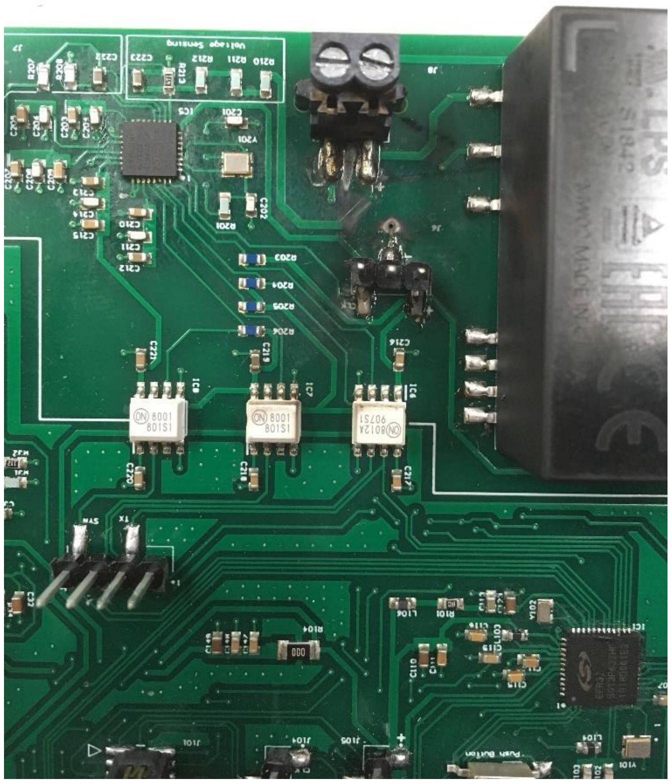


Fig.20: PCB with burnt section near top connector

While debugging Hardware for STPM34, I wanted to check the input to Voltage ADC. As soon as I connected the Oscilloscope's ground with my boards ground there was a large arching and it blew up our AC – DC converter. It also, burnt few of our power tracks along with nearest ground via. This was unintended but it gave us the opportunity to verify our isolation design. First thing we did next was to check Blue Gecko which is on isolated side of our board. We were successful in programming it and connecting Silicon Labs Mobile App to verify the Bluetooth connection. Well, this was our WOAH movement that we got to check our Power isolation!!!