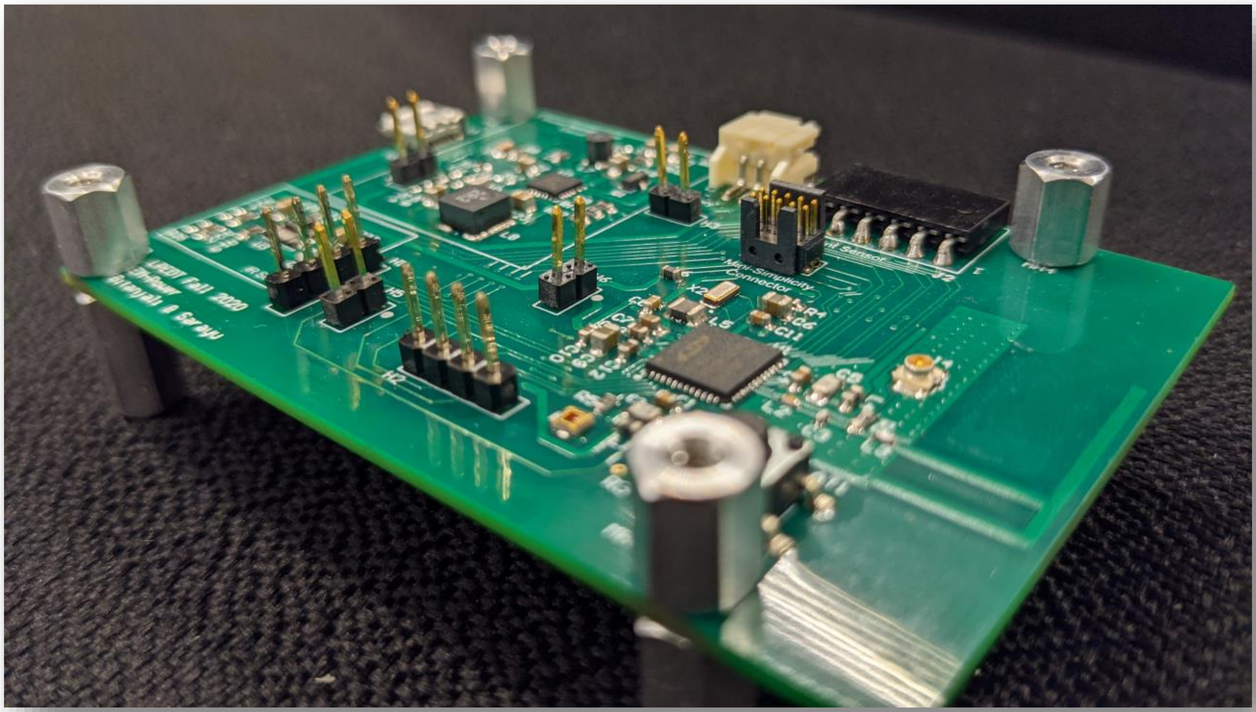
Biometric-based Security System

Team EM-Power | Gitanjali Suresh and Sarayu Managoli

12/12/2020



For the course Low Power Embedded Design Techniques

Under the Guidance of
Prof. Randall Spalding

University of Colorado Boulder
Fall 2020

Table of contents

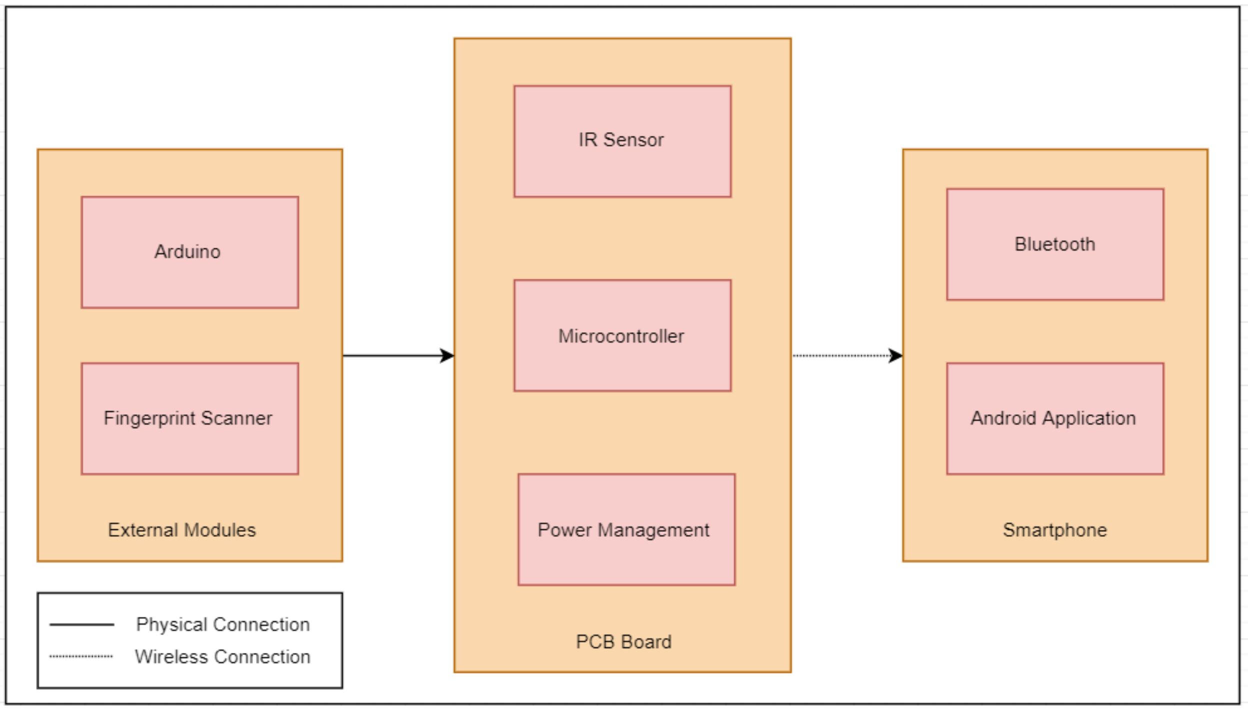
Project Overview

This project deals with a security system that is designed to sense and authenticate the users in order to permit them into a common space such as a lab, room, etc. The system is designed to be placed at the entrance of a room, entering which would require the users to utilize the fingerprint scanner. We display the information of the person who is authorized to enter the room on the application paired using Bluetooth. If there is a mismatch in the database, an alert LED is turned ON. Technology becomes more relevant if systems are made more reliable in terms of energy efficiency and low power consumption. The system includes the use of an IR sensor which indicates that a user is approaching and only then do we activate the fingerprint scanner. This design approach is intended to implement load power management as a part of this project, thus satisfying the low power requirements.

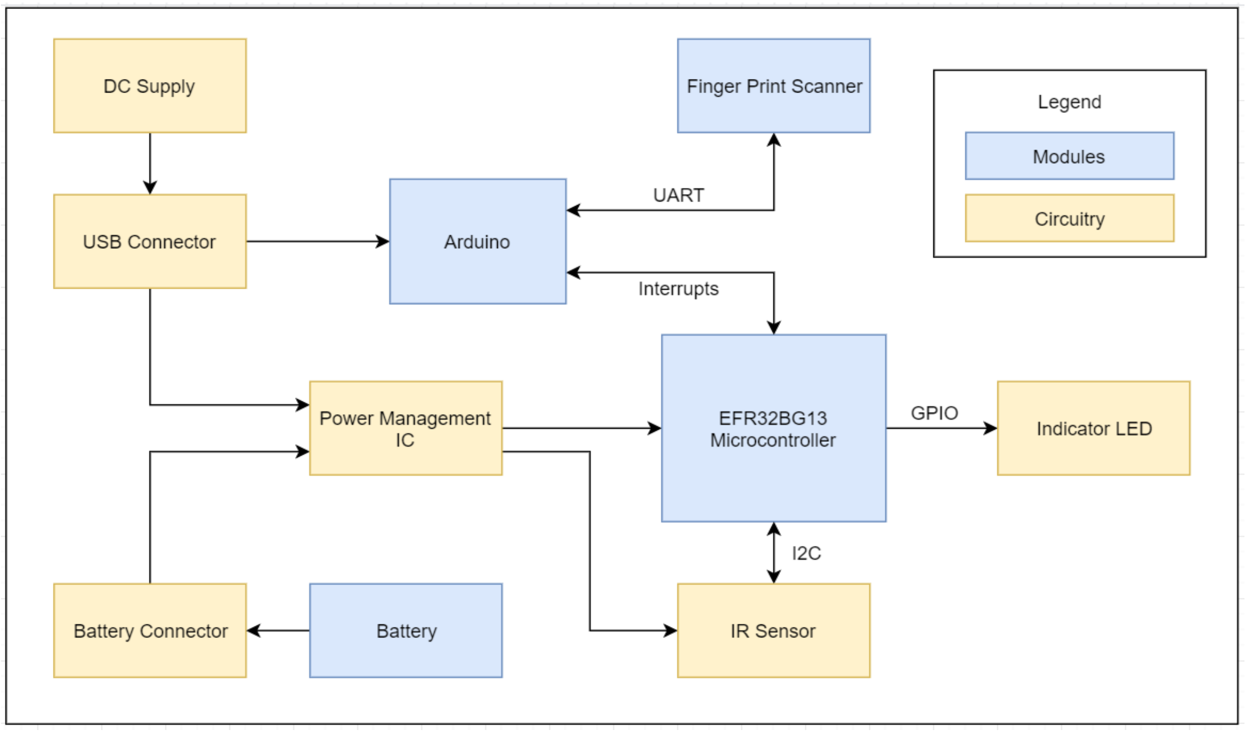
What issue does this project solve?

During the current times, this project holds great significance since all public areas have a restriction in place in terms of the number of people who can be present in a confined space at a time. Even at institutions like CU, only a certain number of students are allowed inside the lab at once. Placing this system outside the lab would help the instructor in staying informed about the students who are present and working in the lab without having to enter the space themselves. As a future scope for this project, using NFC, we could authenticate the instructor’s phone to access the information regarding the attendance. This would also help in keeping track of the students/admins who have entered the premises.

Functional Block Diagram



Board Block Diagram



Key Components

1. System On Chip - EFR32BG13
   Reference manual can be found here
   Link to purchase can be found here

   For this project, we are using this processor by Silicon Labs. This will play the role of the 32-bit ARM Cortex-M4 controller for all the other ICs and sensors. Multiple communication protocols are supported by the processor which helps us in interfacing various sensors as a part of the implementation. This SOC has been selected since it has the capabilities to support flexible energy modes and hence, is better suited for Load Power Management. The wireless Gecko includes a high-performance radio transceiver.

2. Fingerprint Sensor
   Datasheet can be found here
   Link to purchase can be found here

   This is a UART based sensor. Hence, the interface is simple. The sensor operates at a range of 3.3~6V which suits our implementation. The time taken to image a fingerprint is 1 second, which makes it responsive enough for the application. The full dimensions of the sensor are 14 x 12.5 mm. All the accessible fingerprints have to be enrolled in the system for better efficiency. This sensor does not have low power modes, hence we are using a IR sensor to manage power supplied to this sensor.

   This sensor has been chosen since it is light, compact, and can capture up to 200fingerprints which are sufficient for our implementation. The interface is UART which is simple and easier to implement compared to the other communication protocols. However, since it does not have inherent energy modes, we plan to use the IR sensor to wake the fingerprint sensor once motion is detected.

3. IR Sensor
   Datasheet can be found here
   Link to purchase can be found here
   Reference Module can be found here

   IR sensor is essential to our project. Since this is used to sense the presence of a user, we have decided to add this sensor to in turn enable or disable the fingerprint sensor. This helps in the implementation of Load Power Management. The sensor is highly reliable and has a high amount of sensitivity. Since it is widely used for battery-powered products, it is suitable for our project as well.

4. Android Application

To log and access data that is obtained from the sensor, we plan on developing an application compatible with Android devices. Through this, we will be able to keep a tab on the users who have utilized the fingerprint sensor to assert their presence in the said space.

5. Battery - Lithium Ion battery
   Datasheet can be found here
   Link to purchase can be found here

   This battery operates at an approximate voltage of 3.7V. This supports our implementation since one of our sensors operates at 3.3V and the other at 5V. The voltage is low enough not to tax the regulating circuits. The typical capacity of the battery is 650mAh. We calculated the required battery capacity to be approximately 544mAh. Hence to choose a safer capacity, we chose the said battery. This weighs approximately 13 grams which is suitable for the implementation of this project. The maximum battery output voltage is 4.2V and minimum goes down to 3.0V

6. Power Management Circuit
   a. Power Management and Battery Charger IC
      Datasheet can be found here
      Link to purchase can be found here

      The LTC3566 family are highly integrated power management and battery charger ICs for Li-Ion/Polymer battery applications. They include a high efficiency current limited switching PowerPath manager with automatic load prioritization, a battery charger, an ideal diode, and a high efficiency synchronous buck-boost switching regulator. Designed specifically for USB applications, the LTC3566 family's switching power manager automatically limits input current to a maximum of either 100mA or 500mA for USB applications or 1A for adapter-powered applications. As mentioned in the above section, the design requires us to use a LDO to step-down 3.7V to 3.3V. This IC consists of an always-on LDO which supplies a voltage of 3.3V. The input voltage is in the range of 2.7V to 5.5V which is suitable for our selected battery as well. The buck-boost regulator can be programmed for output voltages greater than 2.75V and less than 5.5V. This seems to be a more suitable PMIC compared to our earlier understanding of using a charge management IC and an independent buck-boost converter.

Power Management Unit

The PMU consists of all the elements that are needed to generate and supply the required voltage to the system including the processor and the sensors. The fingerprint sensor requires a voltage of 5V and the processor and the IR sensor require a voltage of 3.3V. Hence, in the unit, we include a buck-boost converter.

As explained above, to bring the voltage up to an operational voltage of 5V, we would be requiring a boost converter and to bring it down to 3.3V, we would be needing a LDO regulator.
Current utilized by each of the sensors

Fingerprint sensor	Normal Mode	<50mA
IR sensor	Normal Mode	100µA
EFR32BG13	EM0	4.992mA
	EM1	2.496mA
	EM2	3.3µA
	EM3	2.8µA
	EM4	1.1µA

C-rate of the Battery

A C-rate is a measure of the rate at which a battery is discharged relative to its maximum capacity. A 1C rate means that the discharge current will discharge the entire battery in 1 hour. The C rate indicates the current rather than the current density.

As seen in the data sheet, the C rate of the battery used for this project is 0.2C. The battery capacity measures 650 mAh. This means to say that the battery produces 650 mA of current per hour. Considering the C rate of the battery, it can produce (⅕) of 650 mAh for 5 hours. Hence, the battery selected produces 130 mA of current for 5 hours.

The battery also has a discharge voltage of 3V. Standard current during both charge and discharge cycles are 0.2C.

Charge and discharge cycles

The battery efficiency comes down as we keep charging and discharging. According to the manufacturer, the battery specified above should be used or about 500 charge cycles. The maximum voltage for charging is 4.2V, and the cutoff voltage in discharge is 3V. If batteries are charged or discharged at voltages outside these ranges, battery performance and safety are compromised. During charge cycles, the operating temperature range is +10°C to +45°C and during discharge cycle, the operating temperature range is -20°C to +60°C. However the storage temperature range is -20°C to +25°C for upto 3 months. The maximum charging current is 0.5C and the maximum discharge current is 2.0C. When used correctly, Lithium-ion Polymer Battery Pack (Rechargeable Single cell Battery) provides a safe and dependable source of power. Pack is made at temperatures between +10°C and +25°C, never exceeding +30°C In this way the maximum shelf-life (i.e. max. retention of cell performances after storage periods) of Lithium-ion Polymer Battery Pack is achieved.

Updates:

1. Alternate charging source other than energy harvester and backup solution to charge the energy storage element.
   The PMIC selected provides us with an option to use a DC source such as a USB port or wall adapter to charge the battery. The USB source is programmable to either 100mA or 500mA. Because power is conserved, the LTC3566 family allows the load current on V_{OUT} to exceed the current drawn by the USB port, making maximum use of the allowable USB power for battery charging. As presently, there is no defined energy harvester solution provided, hence USB is the only source of charging provided to the battery.

2. What is the maximum charging current allowed by the PMU circuitry?
   For the power management IC selected, the maximum charge current specified is equal to 1.5 A. The synchronous buck-boost DC/DC can provide up to 1A. Maximum constant current mode charge current comes to approximately 1A. If the combined load at V_{OUT} is large enough to cause the switching power supply to reach the programmed input current limit, the battery charger will reduce its charge current by the amount necessary to enable the external load to be satisfied.

3. What is the maximum charging current allowed by the energy storage unit?
   The maximum charging current allowed by the energy storage unit is rated at 0.5 C.
   Hence, for this battery of 650 mAh capacity, 0.5 * 650 = 325 mA.

4. What will the maximum current of the jump start power source be set to?

The LTC3566 family's switching power manager automatically limits input current to a maximum of either 100mA or 500mA for USB applications or 1A for adapter-powered applications. Since we plan to power it using a USB, it would be optimum to set the maximum current of the jump start power supply as well to less than 500mA.

In order to answer the last update feedback question asking how we plan to limit this current to be below 325mA for a USB-powered application, we plan to make use of the ILIM0 and ILIM1 control pins of the PMIC. If both these pins are kept low, we can set the USB current limit to 100mA.

5. Where will jump start power and ground signals connect to?
   The jump start power signal would be connected to the V_{BUS} pin of the PMIC. The range of this pin is 4.35V to 5.5V. The jump start ground signal would be connected to one of the GND pins of the PMIC.

6. What are the connection points to enable external power to the digital/MCU portion of the board?
   The way the PMIC is designed, depending on available V_{BUS} power, a Li-Ion battery on BAT will either deliver power to V_{OUT} through the ideal diode or be charged from V_{OUT} via the battery charger. Through V_{OUT}, power is provided to LDO3V3. If V_{IN1}(obtained from V_{OUT}) is significantly greater than the programmed V_{OUT1}, then the converter will operate in buck mode. If V_{IN1} is significantly less than the programmed V_{OUT1}, then the converter will operate in boost mode. This way, we can provide an external power source in the form of a USB or Wall adapter.

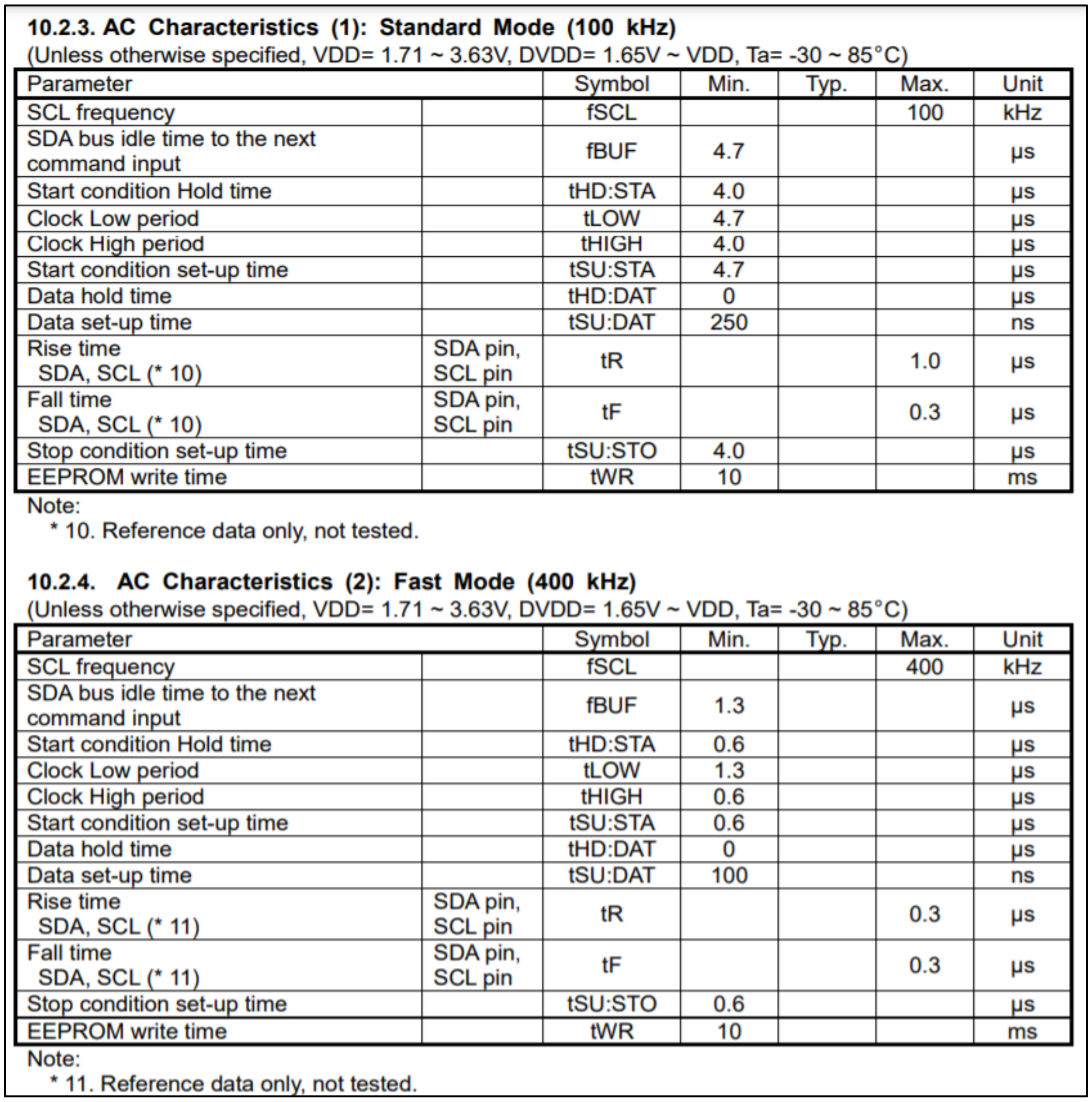
7. How much current will the energy storage element and the PMU be able to provide?
   The battery, in one single charge, provides upto 325 mA of current. The PMIC on the other hand provides 30mA through the LDO regulated output and upto 2.5A through the buck-boost regulated output.

Communication Protocols used by the Sensors

Fingerprint sensor:
The Fingerprint Scanner works as a slave device with the microcontroller (master device) controlling it by sending commands serially through a UART interface. The command interface: 19,200 bps, 1 start bit, 1 stop bit, no parity (8N1).

In an ideal case (when there is no delay in detecting the start bit and when no setup or hold times are required for sampling), the error margin at the start bit position is 50% (in other words, the sampling position may shift up to 50% before or after the center).

IR sensor:
The IR sensor uses I2C protocol to communicate with the processor.



10.2.3. AC Characteristics (1): Standard Mode (100 kHz)
(Unless otherwise specified, VDD= 1.71 ~ 3.63V, DVDD= 1.65V ~ VDD, Ta= -30 ~ 85°C)

Parameter		Symbol	Min.	Typ.	Max.	Unit
SCL frequency		fSCL			100	kHz
SDA bus idle time to the next command input		fBUF	4.7			μs
Start condition Hold time		tHD:STA	4.0			μs
Clock Low period		tLOW	4.7			μs
Clock High period		tHIGH	4.0			μs
Start condition set-up time		tSU:STA	4.7			μs
Data hold time		tHD:DAT	0			μs
Data set-up time		tSU:DAT	250			ns
Rise time SDA, SCL (* 10)	SDA pin, SCL pin	tR			1.0	μs
Fall time SDA, SCL (* 10)	SDA pin, SCL pin	tF			0.3	μs
Stop condition set-up time		tSU:STO	4.0			μs
EEPROM write time		tWR	10			ms

Note:
* 10. Reference data only, not tested.

10.2.4. AC Characteristics (2): Fast Mode (400 kHz)
(Unless otherwise specified, VDD= 1.71 ~ 3.63V, DVDD= 1.65V ~ VDD, Ta= -30 ~ 85°C)

Parameter		Symbol	Min.	Typ.	Max.	Unit
SCL frequency		fSCL			400	kHz
SDA bus idle time to the next command input		fBUF	1.3			μs
Start condition Hold time		tHD:STA	0.6			μs
Clock Low period		tLOW	1.3			μs
Clock High period		tHIGH	0.6			μs
Start condition set-up time		tSU:STA	0.6			μs
Data hold time		tHD:DAT	0			μs
Data set-up time		tSU:DAT	100			ns
Rise time SDA, SCL (* 11)	SDA pin, SCL pin	tR			0.3	μs
Fall time SDA, SCL (* 11)	SDA pin, SCL pin	tF			0.3	μs
Stop condition set-up time		tSU:STO	0.6			μs
EEPROM write time		tWR	10			ms

Note:
* 11. Reference data only, not tested.

AK9753 has two Modes, Normal Mode and Switch Mode. Normal Mode is the mode which controls AK9753 by using I2C interface. The digital output of the four IR sensors and the internal temperature sensor can be used through the I2C interface in Normal Mode. INT output also can be used.

Key Requirements of the Project

Some of the key requirements of the project are mentioned below.

Battery Life Expectation

The battery we plan to use has a capacity of 650mAh and a lifespan of around 500 charge cycles. Storage temperatures above room temperature will increase the rate of self-discharge, reducing the available capacity of the cell. As these cycles repeat, the efficiency of the battery reduces. But keeping up with the manufacturer instructions, we plan to use it for more than 500 charge cycles.

To make the project more energy-efficient, we are planning on scheduling the processor such that it runs on low energy mode (EM2). Every 2 seconds the processor is woken up in EM1/EM0 mode to sense the presence of a user. If the user is present, the IR sensor (powered through the circuit) indicates accordingly, and the fingerprint sensor is activated by providing it with enough power to accept user data. This way, we plan to accomplish low power consumption. We assume that the user who intends to use the fingerprint sensor would be present for more than 3-5 seconds and hence, we would not be missing anyone who actually wants to use the system. The IR sensor is not expected to run on its own, it is expected to be activated by the processor when it wakes from a defined duration of sleep. The processor is woken up through an internal timer interrupt.

The whole system is planned to be battery operated using a separate circuitry for battery management. We plan on charging the battery using a 5V USB.

The processor consumes 4.992mA in EM0 mode, 2.496mA in EM1 mode, 3.3µA in EM2 mode, 2.8µA in EM3, and 1.1µA in EM4. This is operating current when the processor operates at 3.3V and no DC to DC converter is in use.

Specifications

Dimensions: 90mm X 50mm.
Battery: 3.7V chargeable
Processor: 32-bit, 512KB Flash, 64GB RAM, Operating voltage: 1.8V-3.8V, Output Power: 19 dBm, 12-bit ADC resolution

Fingerprint Sensor: Operating Supply Voltage: 3.3V to 7.5V, Operating Supply Current: 50mA, Operating temperature range: -20°C to 60°C, Dimensions: 50mm x 23.2mm x 25mm

IR Sensor: Operating Supply Voltage: 1.71V to 3.63V, Operating Supply Current: 100µA in continuous mode and 1µA in power down mode, operating temperature range: -30°C to 85°C, Dimensions: 3.8mm to 4.6mm

Expected System Operating Temperature: -15°C to 70°C
Wireless Range: 50 feet
Relative humidity: 40% to 85% RH
Warranty: 3 - 4 years

Some of the parameters mentioned above are approximately noted and are subject to change.

Use Case Model

Energy consumption for each of the sensors and transceivers used has been calculated/mentioned below with approximate periods of operation.

The device is designed to operate at the lowest energy modes at most times. Every 2 seconds, the IR sensor is turned ON to operate at the required energy level. If it happens to sense the presence of a user, we intend to power up the fingerprint scanner. This later logs the user information and glows the LEDs accordingly. During the sensing phase, the device operates in EM0 mode. Bluetooth authentication is initiated based on user requests.

Calculations:

Blue Gecko Input Voltage = 3.3V
Blue Gecko Output Voltage = 1.8V
Blue Gecko Efficiency = 1.8/3.3 = 54.54%

IR Sensor Voltage = 1.7 to 3.3V
IR Sensor Current = 100µA

Fingerprint sensor Current = 50mA

EFR32BG13 ON Current, considering EM0 mode of operation = 87µA/MHz @ 38.4MHz
= 3340.8µA

EFR32BG13 OFF Current, considering EM2 mode of operation = 2.0µA

EFR32BG13 Receive Current at 1 Mbps at 2.4 GHz operation = 9.5mA
EFR32BG13 Transmit Current at 0 dBm output power 2.4 GHz operation = 8.5mA
EFR32BG13 Average Current during Bluetooth operation = (9.5 + 8.5) / 2 = 9mA

Assumptions:
IR Sensor will be turned on or completely powered for a period of 2s.

The Fingerprint sensor will be turned on or completely powered for a period of 2s if a human presence is detected by the IR sensor.
Assuming all this operation happens once every 10s (not completely periodical), in a period of 10s, the processor will be actively powered for 4s and is expected to be in low power and/or sleep state for 6s.

Number of activations in 1 hour = 180

On duty cycle = (180 * 4 seconds) / (60 minutes * 60 seconds) = **20%** (This is when IR and Fingerprint sensors are both considered). Individually, the ON duty cycle is 10%.

Off duty cycle = 1 - 0.2 = **80%**

Total worst-case current consumption when IR is ON = IR Sensor Current + EFR32BG13 Current during Bluetooth Operations = 100µA + 9mA = **9.1mA**

Total worst-case current consumption when Fingerprint sensor is ON = Fingerprint Sensor Current + EFR32BG13 Current during Bluetooth Operations = 50mA + 9mA = **59mA**

Weighted average power out of battery = [IR(On duty cycle * Current consumption * Blue Gecko Output Voltage) + Fingerprint(On duty cycle * Current consumption * Blue Gecko Output Voltage)] / Blue Gecko efficiency = [(0.1 * 59mA * 1.8V)+(0.1 * 9.1mA * 1.8V)] / 0.545
= 22.49174mW = **22491.7µW**

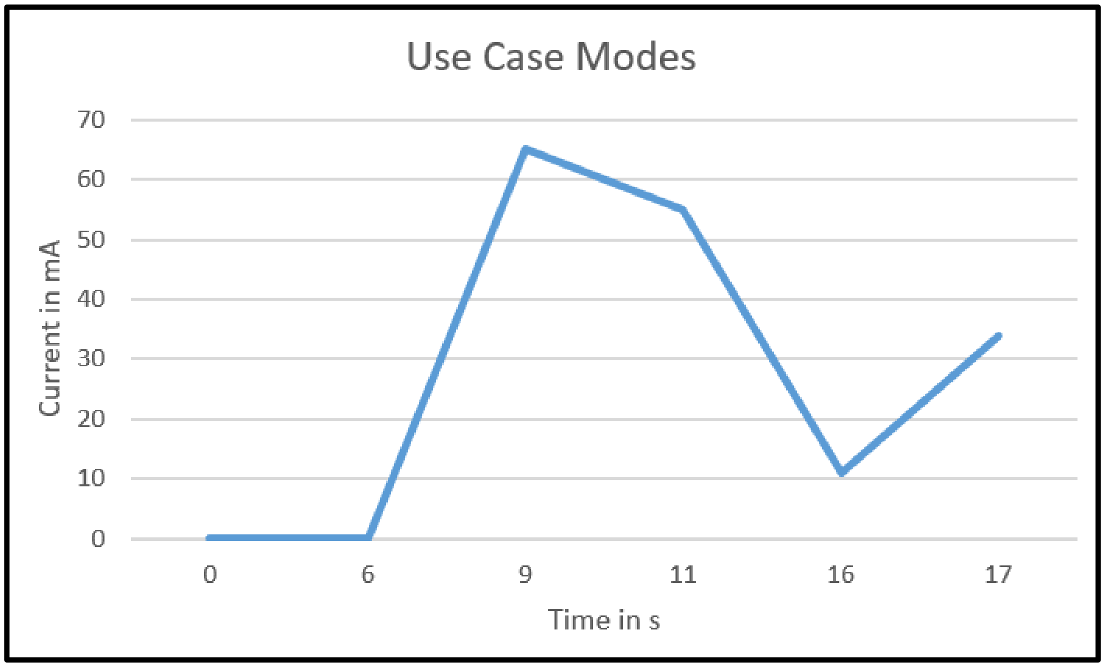
Weighted average power out of battery = (Off duty cycle * Current consumption * Blue Gecko Output Voltage) / Blue Gecko efficiency = (0.8 * 2.0µA * 1.8V) / 0.545
= **5.28µW**

Total average power out of battery = 22491.7µW + 5.28µW = **27771.7µW**

Average energy consumption assuming 8 hours of operation = 8hr/use * 60min/hr * 60sec/min * 27771.7µA/sec = **451.48J**

Total average current consumption = 0.1 * 59000µA + 0.1 * 9100µA + 0.8 * 2.0µA
= **6811.6µA**

Battery use capacity = 6811.6µA * 8 = 54492.8µAH = 54.492mAH
Nominal battery capacity = 10 * 54.492mAH =544.92mAH


Use Case Modes
Current in mA
70
60
50
40
30
20
10
0
0
6
9
11
16
17
Time in s

Schematic Updates

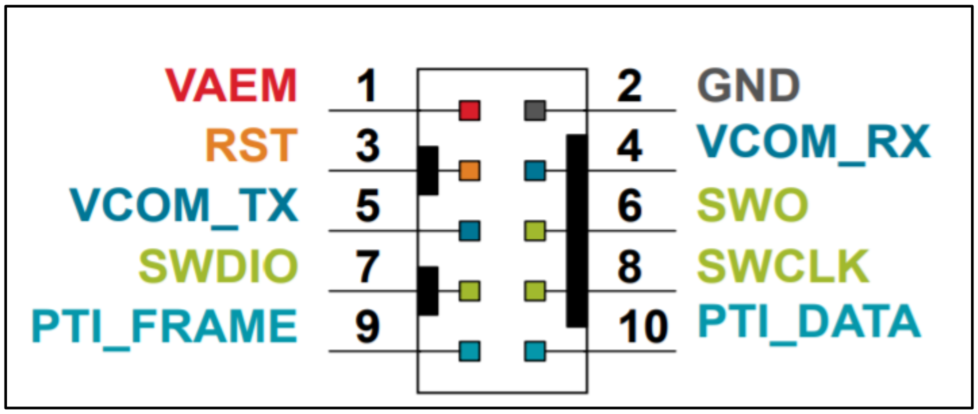
Microcontroller Circuitry

Circuitry for Programming and Debugging

The EFR32BG13 Wireless Gecko devices include hardware debug support through a 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface, as well as an Embedded Trace Module (ETM) for data/instruction tracing. This interface is used to program and debug the Blue Gecko.

A Mini Simplicity connector is used to provide the necessary debug features for a space constrained environment. The Mini Simplicity connector is a 10-pin connector which supports the following set of advanced debug features:-

1. Serial Wire Debug (SWD) with SWO
2. Packet Trace Interface (PTI)
3. Virtual COM port (VCOM)
4. AEM monitored voltage rail

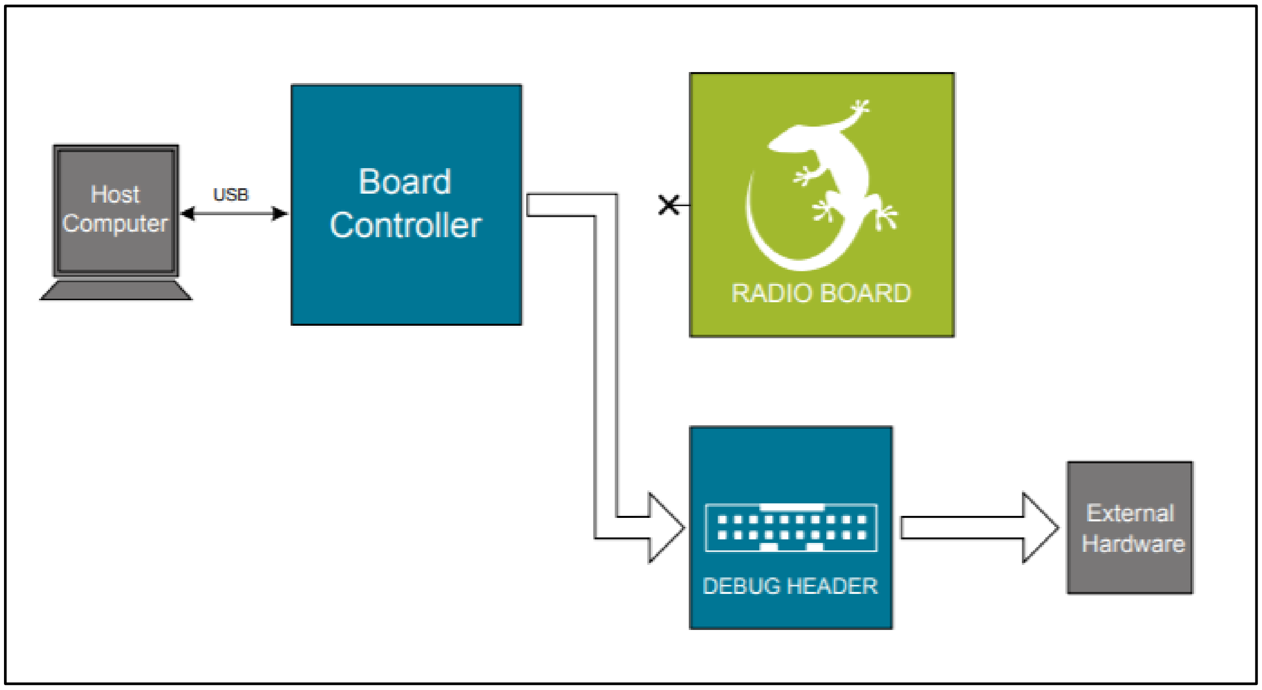


The pin descriptions of the connector are as follows:

Pin Number	Function	Description
1	VAEM	Target voltage on the debugged application. Supplied and monitored by the AEM when power selection switch is in the "AEM" position.
2	GND	Ground
3	RST	Reset
4	VCOM_RX	Virtual COM Rx
5	VCOM_TX	Virtual COM Tx
6	SWO	Serial Wire Output
7	SWDIO	Serial Wire Data
8	SWCLK	Serial Wire Clock
9	PTI_FRAME	Packet Trace Frame Signal
10	PTI_DATA	Packet Trace Data Signal

The MCU is programmed with the help of this Mini Simplicity Connector, which is connected to the Debug Adapter provided with the Blue Gecko Wireless Starter Kit, which combines the functionality of the 20-pin debug connector and the 20-pin simplicity connector. Therefore, in order to program and debug the developed embedded system 3 external things are essential namely -

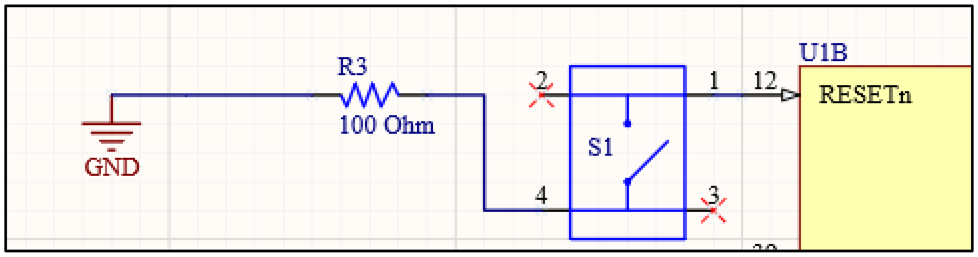
1. Debug Adapter
2. Wireless Starter Kit of Blue Gecko
3. Host System



As can be seen from the above figure, the external hardware is the custom developed embedded system using EFR32BG13. And the Board Controller is the Wireless Starter Kit with the interface between the External Hardware and the Debug Header acting as the Debug Adapter.

Reset Circuitry

The reset circuitry is designed by referring to the EFR32BG13 Reference Manual. The internal reset circuitry of EFR32BG13 is designed in such a way that forcing the RESETn pin low generates a reset of the EFR32xG13 Wireless Gecko. The RESETn pin includes an on-chip pull-up resistor, and can therefore be left unconnected if no external reset source is needed.



Clock Generation Circuitry

The EFR32BG13 supports two crystal oscillators and fully integrates five RC oscillators. It supports a high frequency crystal oscillator with crystal frequencies in the range 38 to 40 MHz. By referring to the schematics of the Blue Gecko Starter Kit, a 38.4 MHz crystal was chosen for the high frequency crystal oscillator (HFXO) circuitry. The specifications of the crystal are as follows:

Nominal Frequency = 38.4 MHz
Load Capacitance = 10.0 pF
Frequency Tolerance = 10 ppm
Series Resistance = 50 Ω
Operating Temperature = -40℃ to 105℃

Based on the above mentioned specifications, a crystal with the following digikey part number was selected = 1908-1209-1-ND

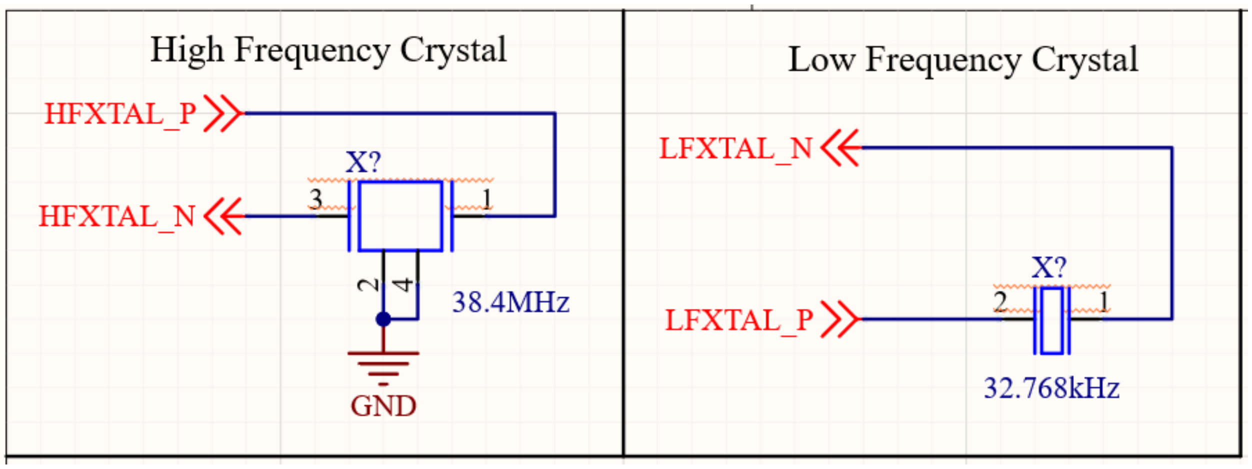
Similarly, a low frequency crystal of 32.768 kHz was chosen for operating EFR32BG13 in low energy modes. This crystal is used for low frequency oscillators (LFXO). The specifications of the crystal are as follows:

Nominal Frequency = 32.768 MHz
Load Capacitance = 12.5 pF
Frequency Tolerance = 20 ppm
Series Resistance = 80 kΩ
Operating Temperature = -40℃ to 85℃

Based on the above mentioned specifications, a crystal with the following digikey part number was selected = 535-13356-1-ND

Both the crystal circuits do not require external crystal load capacitors as they contain on-chip tunable load capacitors.

The circuit design is as shown below which was referred from the schematics of the Wireless Starter Kit for Blue Gecko:



Signal Test Points

The following signals in the boards have test points:

1. Reset Signal (RESET)
2. Power Supply Pins (Both 3.3V and 5V)
3. Ground Pins (GND)
4. IR Sensor Test Points (SCL, SDA, INT)
5. PMU CHRGEN pin
6. Debugger Pins (SWO, SWDIO, SWDCLK)

For our implementations, these test points were extremely useful. We were able to tap these lines and obtain the signals for better analysis of the I2C communication protocol. However, since we struggled to isolate the problem regarding our UART transmission and reception, we believe that inclusion of the test points LEUART_RX and LEUART_TX would have been highly appreciated. We did have these test points as a part of our initial plan, however failed to include them in our final PCB.

Flash Circuitry Information

Programming of the MCU flash requires a current of 3.34mA. As can be seen from the characteristics of the PMIC, the PMIC would be able to provide the sufficient current required to both program the flash and keep the circuit running.

Unlike the Wireless Starter Kit which had an additional serial flash circuit, the internal flash is sufficient for our application. Hence, there is no additional circuitry needed to program the Flash.

I/O Update

The I/O pins of the microcontroller are used by the following:

1. Fingerprint Sensor
2. IR Sensor
3. Debug Connector
4. Power - On LED

- As can be seen from the block diagram the fingerprint sensor uses a UART interface to communicate with the microcontroller. To ensure that there is low power consumption, we exploit the MCU's Low Energy UART peripheral. So for our design we are using the LE_UART0 for the interface. The respective port pins for the transmit and receive from the MCU are PD10 and PD11.

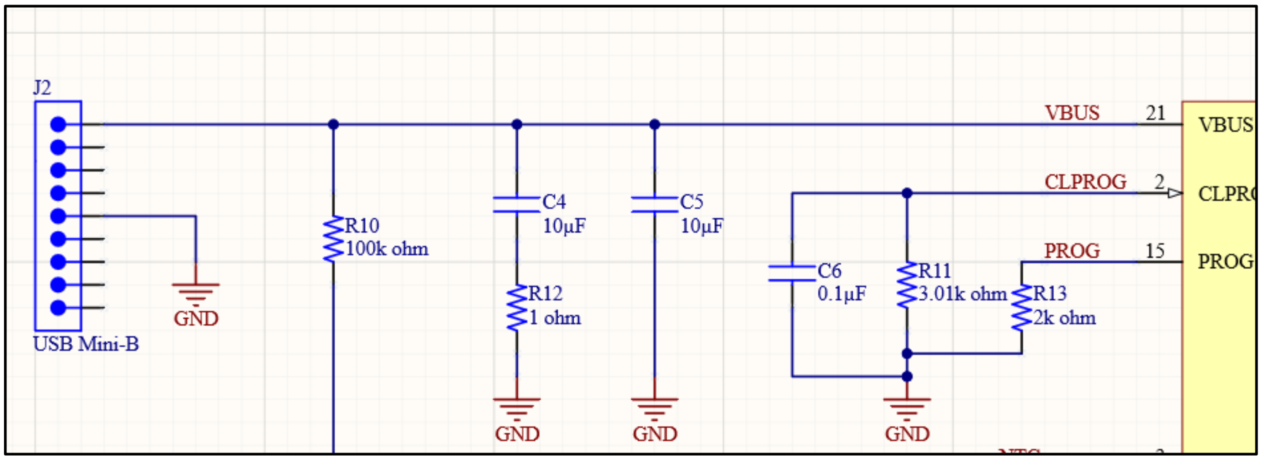
- In case of the IR sensor, it is an I2C interface. In addition to the basic two I2C pins - SCL and SDA, the IR sensor also has an active low interrupt pin. In our design we are planning to use the I2C1 which corresponds to Port C. The specific port pins are :-
    - SDA - PC11
    - SCL - PC10
    - INT - PC9

- For the debug connector we are using the 10-pin mini simplicity connector as explained above under the section, Circuitry for Programming and Debugging. The functionality of the mini-simplicity connector are mapped to the following I/O pins :-
    - PTI_FRAME - PB13
    - PTI_DATA - PB12
    - VCOM_RX - PA1
    - VCOM_TX - PA0
    - SWCLK - PF0
    - SWDIO - PF1
    - SWO - PF2

- To indicate the successful powering up of the MCU, an external LED is interfaced to the port F on pin 4, i.e., PF4.

Is ESD protection required for these I/O ports?

Since USB would not be used by end users, we decided that it would not need the addition of an ESD diode. However a protective circuit has been included in the schematic as shown below.



The datasheet of the PMIC evaluation kit states that some USB cables could cause the voltage VBUS to overshoot when it is plugged in. In order to dampen this effect, it is recommended to include the network of C4, C5 and R14 resistors.
The datasheet also recommends that a type 3 compensation be added for the buck-boost regulator. Currently, type 1 compensation is populated. However, DNP (do not populate) paddings have been provided to accommodate a type 3 compensation if needed in the future.
Post the feedback received on update 6, upon checking, the evaluation board does suggest a 10uF capacitor for the USB. Therefore, that has been retained as is.

Bulk or large decoupling capacitor

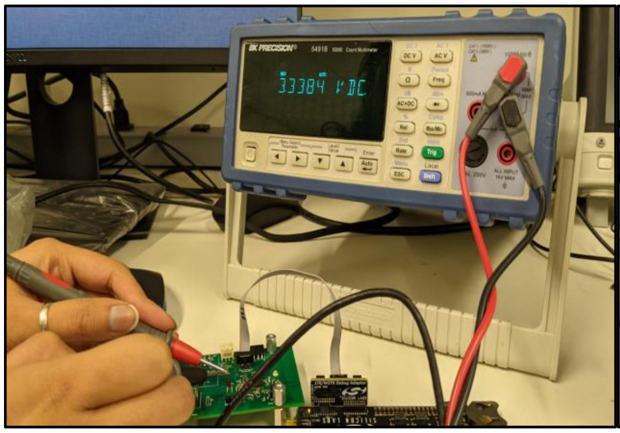
We did not include a bulk capacitor to our PMIC. The datasheet does not specify the requirement of one.

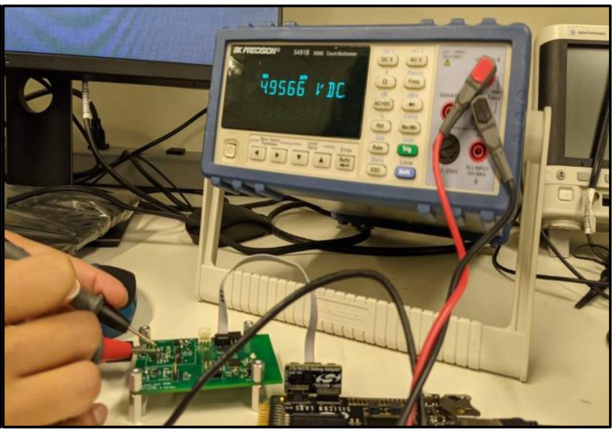
Energy Storage Element

The energy storage element used here is a battery. It is a Lithium Ion battery as specified in the Key components section. This battery operates at an approximate voltage of 3.7V.

Component	Energy Mode	Current Consumed
EFR32BG13	EM0	4.992mA
	EM1	2.496mA
	EM2	3.3µA

PMU Results and Summary





As seen above, the PMU is able to successfully provide the two power rails that are required by our system. The 3.3V is generated using the always ON LDO.

5V is generated using the calculation as shown below:

Calculation of feedback resistance:

According to the data sheet, $V_{OUT1} = V_{FB1}(R_1/R_{FB} + 1)$
$V_{OUT1} = 5V$ and $V_{FB1} = 0.8V$ (given)
Hence, $R_1/R_{FB} = 5.25$
Assume, $R_{FB} = 1$ kohm.
Calculated $R_1 = 5.25$kohm

Calculation of feedback capacitor:

$f_{LP} = 1/(2\pi\sqrt{LC_{out}}) = 1/(2\pi\sqrt{(2.2\mu \text{ X } 22\mu)}) = 22.877\text{kHz}$

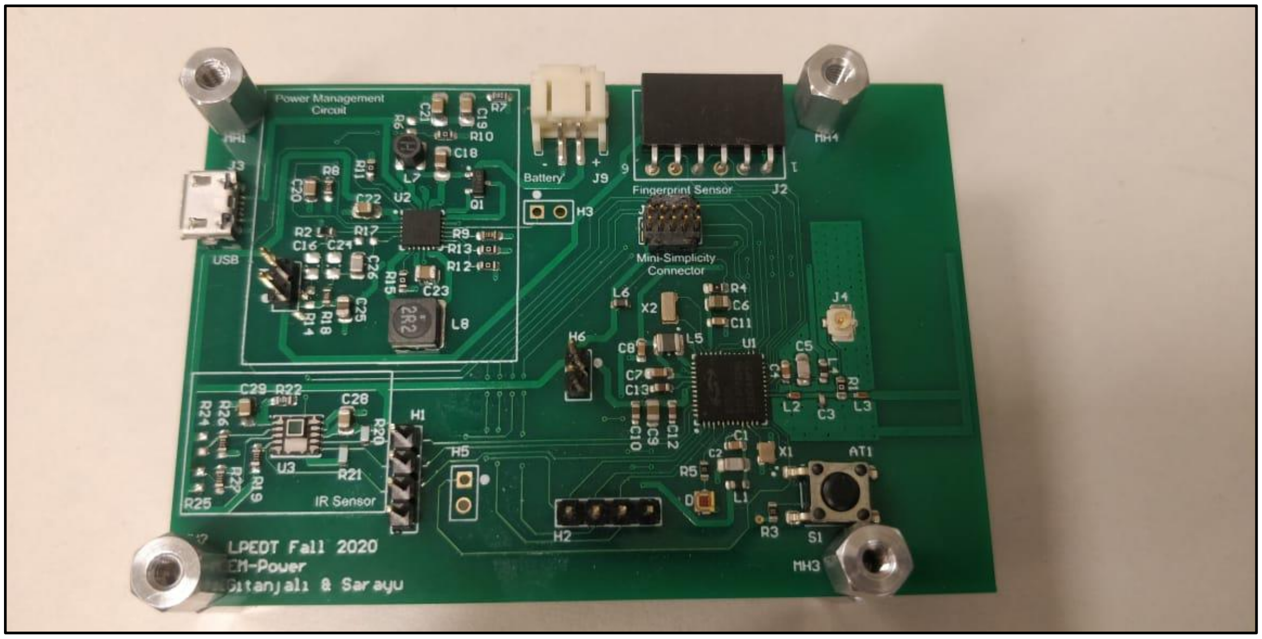
$f_{UG} <= f_{LP}$

$f_{UG} = 1/(2\pi R_1 CP_1)$

$2287.69 = 1/(2\pi \text{ X } 5.23k * CP_1)$

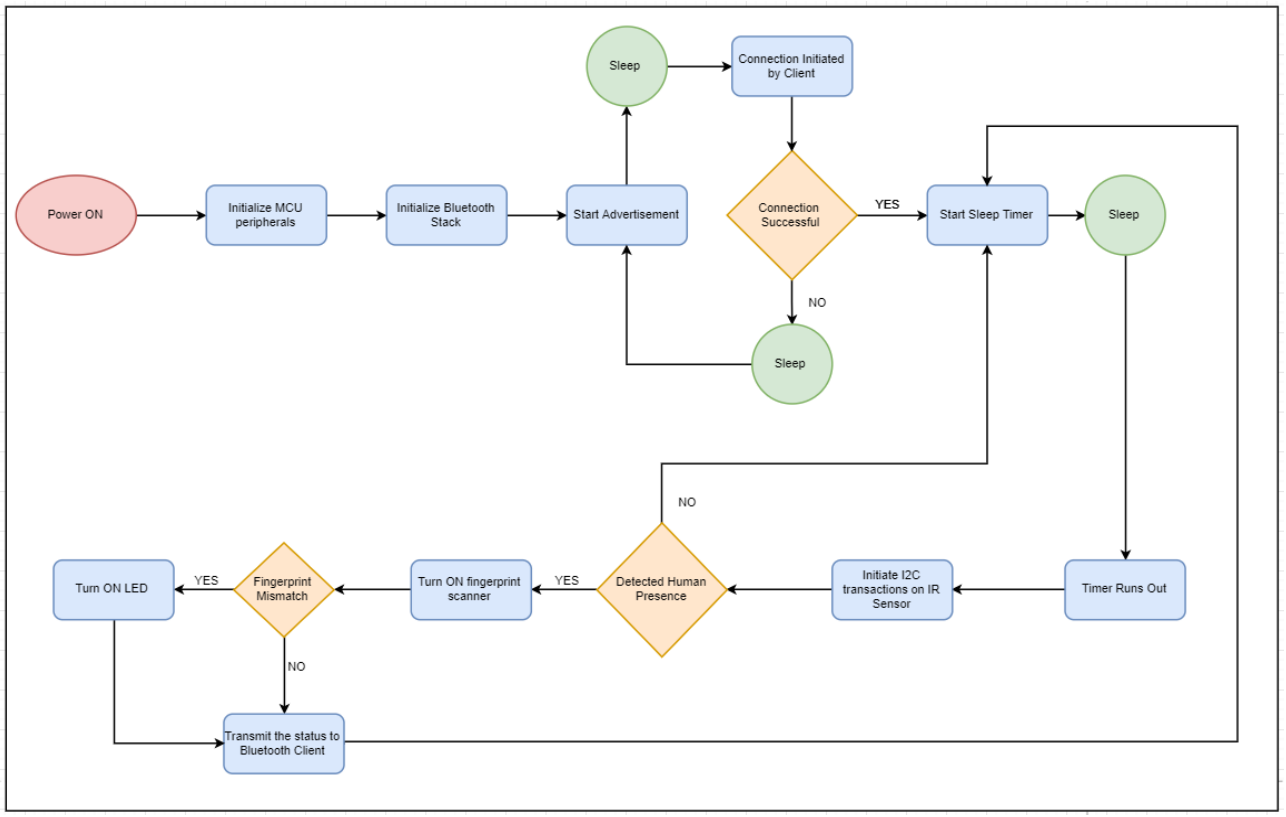
$CP_1 = 0.1\mu F$

Assembled board

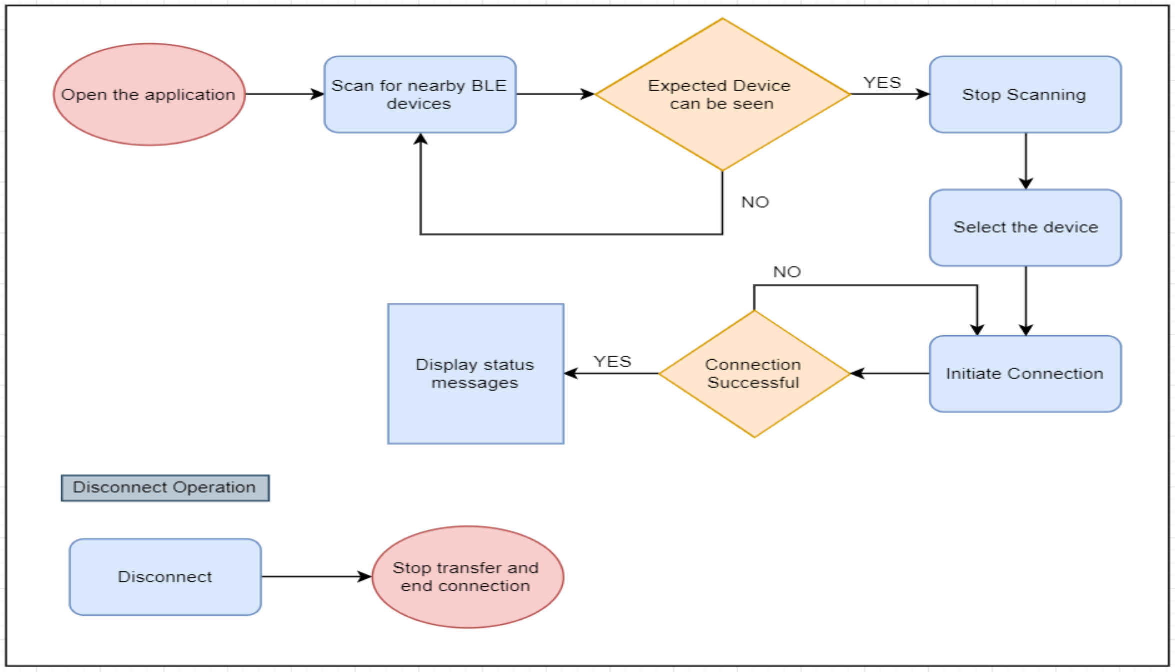


Software flowchart

This is the software flowchart on Microcontroller side:


Power ON
Initialize MCU peripherals
Initialize Bluetooth Stack
Start Advertisement
Sleep
Connection Initiated by Client
Connection Successful
YES
NO
Sleep
Start Sleep Timer
Sleep
Timer Runs Out
Initiate I2C transactions on IR Sensor
Detected Human Presence
NO
YES
Turn ON fingerprint scanner
Fingerprint Mismatch
YES
Turn ON LED
NO
Transmit the status to Bluetooth Client

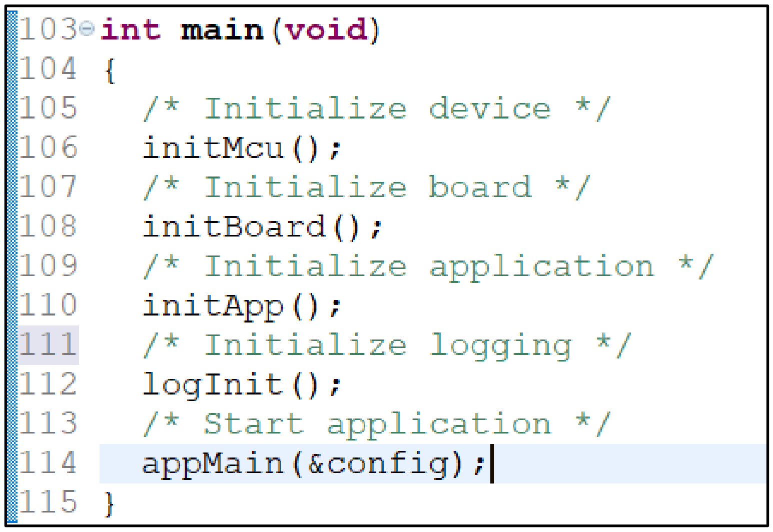
This is the software flowchart from the application perspective -



List of commands

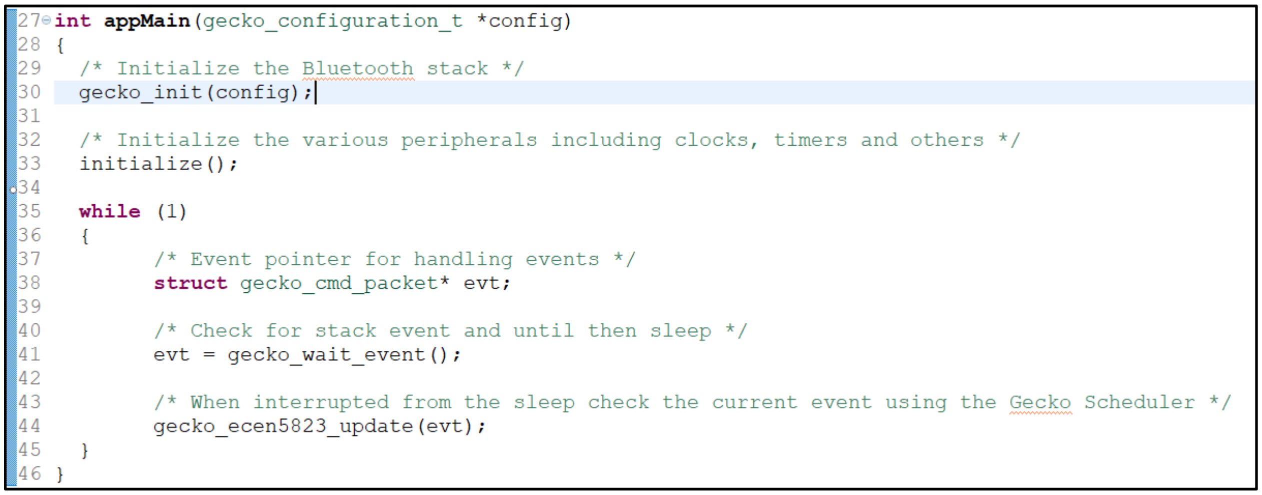
The list of commands can be divided into two categories taking into account the consideration that the commands are sent from both the Microcontroller and the custom-developed Android application.

When the program is initialized, the following commands are used at the microcontroller side -



```c
int main(void)
{
  /* Initialize device */
  initMcu();
  /* Initialize board */
  initBoard();
  /* Initialize application */
  initApp();
  /* Initialize logging */
  logInit();
  /* Start application */
  appMain(&config);
}
```

1. A set of initialization commands are run as shown in the image above, to initialize the MCU and the board as a whole.



```c
27 int appMain(gecko_configuration_t *config)
28 {
29   /* Initialize the Bluetooth stack */
30   gecko_init(config);
31
32   /* Initialize the various peripherals including clocks, timers and others */
33   initialize();
34
35   while (1)
36   {
37         /* Event pointer for handling events */
38         struct gecko_cmd_packet* evt;
39
40         /* Check for stack event and until then sleep */
41         evt = gecko_wait_event();
42
43         /* When interrupted from the sleep check the current event using the Gecko Scheduler */
44         gecko_ecen5823_update(evt);
45   }
46 }
```

2. The appMain() function acts as the main function which initializes the necessary peripherals needed for our application. So in our case, these peripherals include clocks, a low energy Timer (LETIMER), I2C0 and the GPIOs.
3. Followed by this, the Bluetooth stack is initialized using the gecko_init(config) as shown in line 30.

4. Once the gecko is initialized, the Bluetooth starts advertising and waits for a connection from the other client device, which is explained below.

It can be clearly seen from the software flowcharts above that when the program is initialized, it expects a certain set of commands to be sent from the Android application side. The following are the set of commands associated with the application side -

1. Scan for the available Bluetooth devices using the SCAN option on the Application.
2. When the expected device is found in the list, then stop scanning using the STOP SCAN option.
3. The connection is initiated with the selected device using the CONNECT option. This command is sent to the advertising Bluetooth device, which in our case is the custom board.
4. Lastly, another command that can be sent from the application is DISCONNECT which is used to end the transaction and thereby the wireless communication between the smartphone and the custom board.

On the microcontroller side, after the establishment of a valid connection, the following commands are sent to various peripherals:

1. A set of I2C commands are sent to configure and initialize the IR sensor connected to the I2C0 peripheral of the MCU.
    a. Turn on the power supply to IR sensor pins
    b. Reset the IR sensor.
    c. Configure the IR sensor for single mode acquisition.
    d. Configure the IR sensor to send interrupts when the data is ready.
    e. When the Data Ready interrupt is received, read data from the IR and check for the presence of a human.
    f. Turn off the supply to the IR sensor pins.

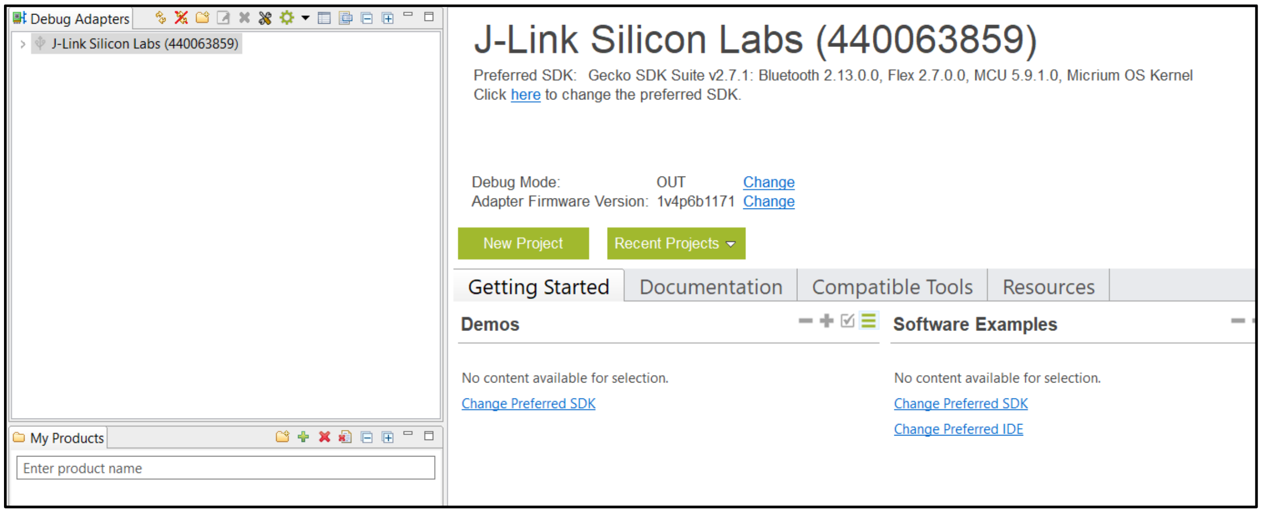
2. When the IR sensor checks for human presence, and if someone is detected then the fingerprint scanner is turned ON by the MCU, by providing an interrupt to the Arduino.
3. Once the fingerprint scanner is turned ON and it detects either an authenticated fingerprint or an unauthenticated one, the information is sent as a response via the Arduino to the MCU.
4. Following this, the microcontroller sends the status of a fingerprint scanner to the Android application via Bluetooth.

This completes the list of commands sent by both the application and the MCU.

Programming the MCU

The programming of the MCU is done using the Simplicity debug adapter board. This adapter board is hoisted on the starter kit for EFR32BG13. The custom board needs to consist of a Mini Simplicity connector to connect the adapter board to the custom board.

1. In order to program using the Simplicity IDE 4, go to Launcher and select J-Link Silicon Labs. This opens up a window as shown:



J-Link Silicon Labs (440063859)

Preferred SDK: Gecko SDK Suite v2.7.1: Bluetooth 2.13.0.0, Flex 2.7.0.0, MCU 5.9.1.0, Micrium OS Kernel
Click here to change the preferred SDK.

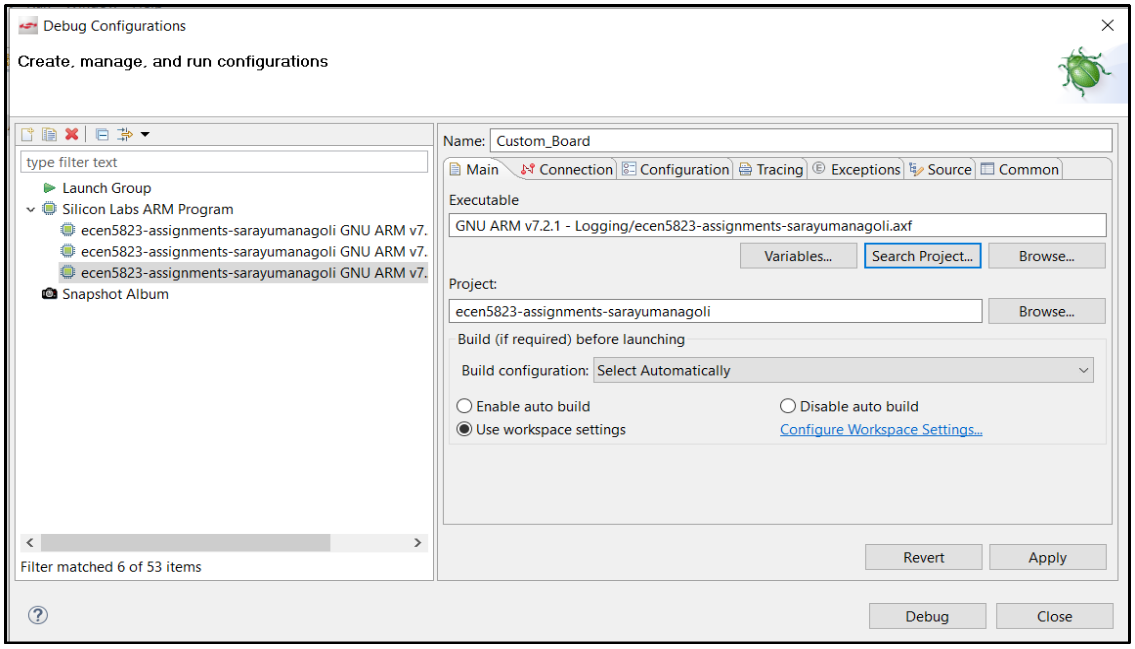
Debug Mode: OUT Change
Adapter Firmware Version: 1v4p6b1171 Change

No content available for selection.

No content available for selection.

2. Select the Debug Mode to OUT. This enables you to interface with a custom board.

3. Now, go to the project you desire to execute on the board, right click on it and select Debug As and click on Debug configuration.


Debug Configurations
Create, manage, and run configurations
type filter text
Launch Group
Silicon Labs ARM Program
ecen5823-assignments-sarayumanagoli GNU ARM v7.
ecen5823-assignments-sarayumanagoli GNU ARM v7.
ecen5823-assignments-sarayumanagoli GNU ARM v7.
Snapshot Album
Filter matched 6 of 53 items
Main
Connection
Configuration
Tracing
Exceptions
Source
Common
Variables...
Search Project...
Browse...
Browse...
Configure Workspace Settings...
Revert
Apply
Debug
Close

Name: Custom_Board

Executable: GNU ARM v7.2.1 - Logging/ecen5823-assignments-sarayumanagoli.axf

Project: ecen5823-assignments-sarayumanagoli

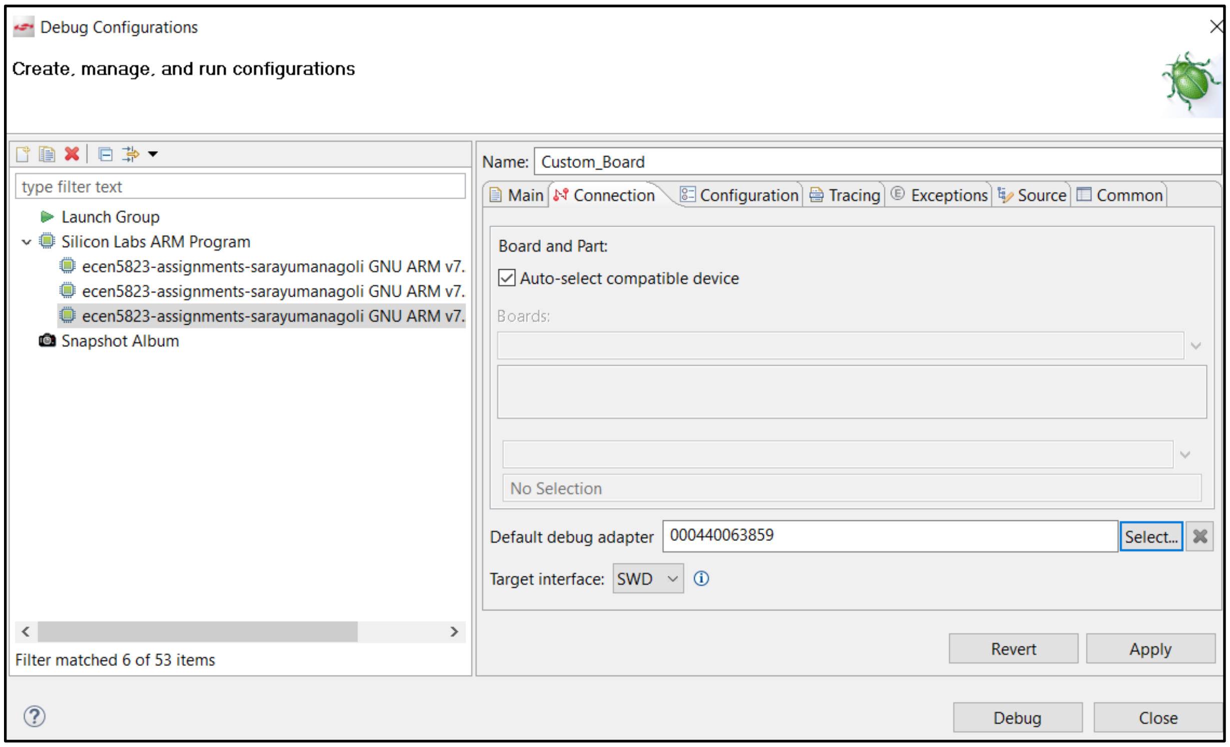
Build (if required) before launching

Build configuration: Select Automatically

( ) Enable auto build ( ) Disable auto build
(●) Use workspace settings

4. Create a new Silicon Labs ARM Program and rename it "Custom_Board". Select the executable that the configuration needs to look for. Select Apply.

5. Now open the Connections tab, and select the following option.



Debug Configurations

Create, manage, and run configurations

type filter text

- Launch Group
- Silicon Labs ARM Program
  - ecen5823-assignments-sarayumanagoli GNU ARM v7.
  - ecen5823-assignments-sarayumanagoli GNU ARM v7.
  - ecen5823-assignments-sarayumanagoli GNU ARM v7.
- Snapshot Album

Filter matched 6 of 53 items

Name: Custom_Board

Main | Connection | Configuration | Tracing | Exceptions | Source | Common

Board and Part:

☑ Auto-select compatible device

Boards:

No Selection

Default debug adapter 000440063859 Select...

Target interface: SWD

Revert Apply

Debug Close

6. Click on Apply.
7. Now if we select Debug, the program will execute through the custom board.

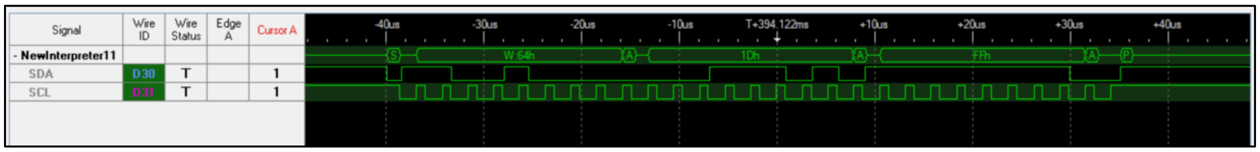
Will an external energy source be required to program the MCU?

Since WSTK is used in order to program the board using the debug connector, the debug connector does supply power from the development board. So technically, an external energy source is being used in order to program the code. Also, while programming, the MCU was tested to successfully program by using just the battery.

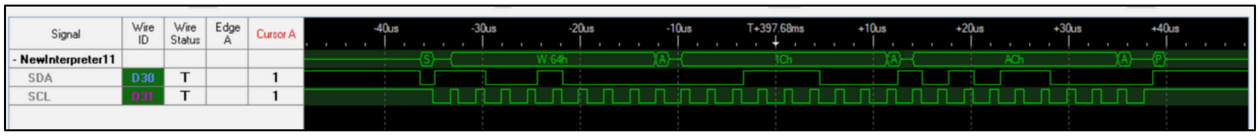
Signal quality analysis of I2C signals

The analysis of the I2C signals can be performed by looking at the data transactions. The entire I2C operations happen in accordance with a certain order. The same order can be seen from the following screenshots.

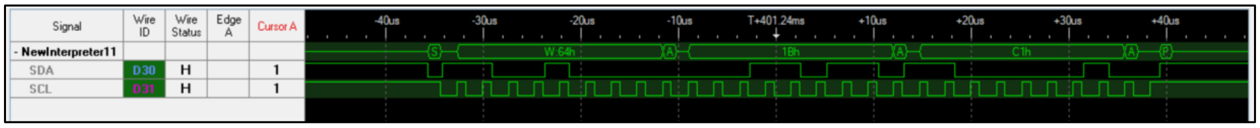
1. The first command is to reset the IR sensor. This is done by writing 0xFF to the control 2 register of the IR sensor that is addressed as 0x1D.



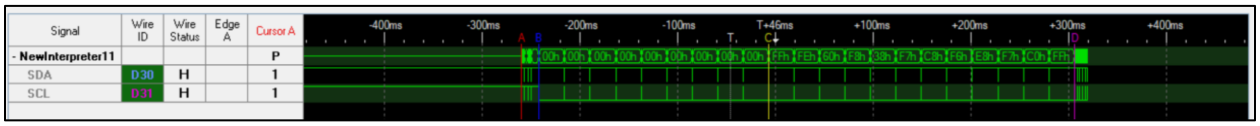
2. The second command is to configure the mode of acquisition as a continuous single shot mode by writing 0xAC to the EPROM control register addressed as 0x1C.



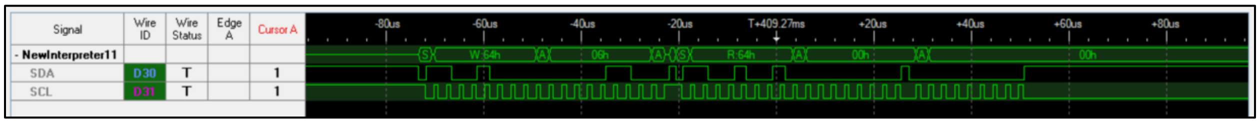
3. The third command is to configure the IR sensor to fire an interrupt from its INT pin when the data is ready to be read. This is done by writing the command 0xC1 to the Interrupt Control register addressed by 0x1B.



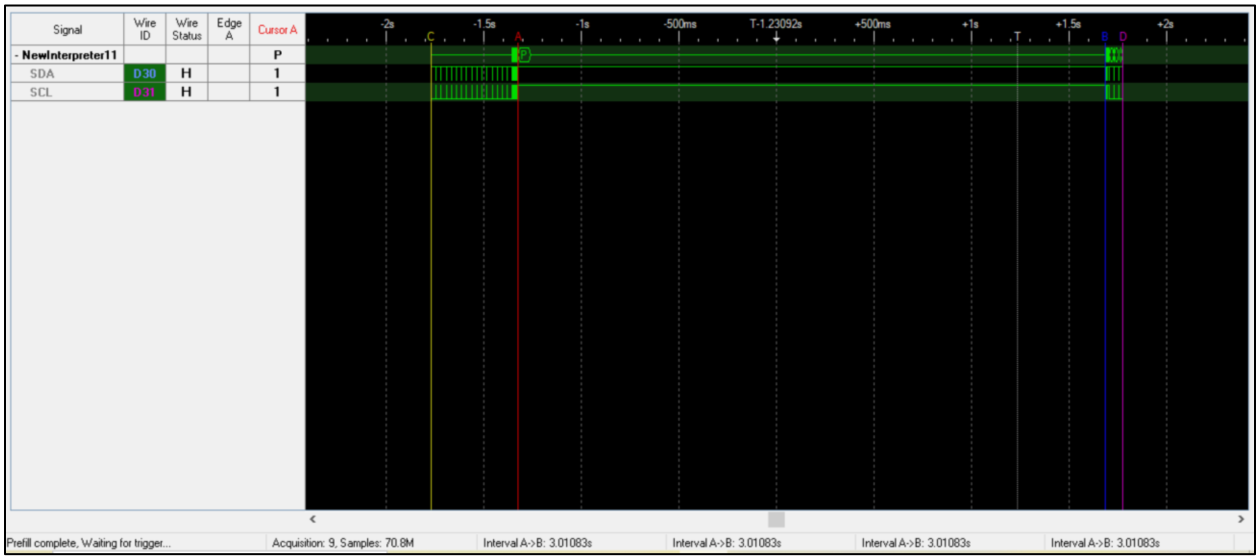
4. Once a data ready interrupt is received by the MCU, it performs the next I2C transaction to perform a read. The IR sensor is designed in such a way that, initially the registers are locked. In order to unlock them, all the data registers must be read once and then the next time a read is performed again the data can be read. This is clearly seen in the below screenshot where the first read of the data registers gives a 0x0 and then the consecutive reads give valid data.



5. For clarification to show that the first command that was given was a read, a zoomed in picture of the above snapshot has been attached below. Here, it can be seen that a read command is issued with a request for read from the address 0x06 which is the location of the IR register 1 (low byte).



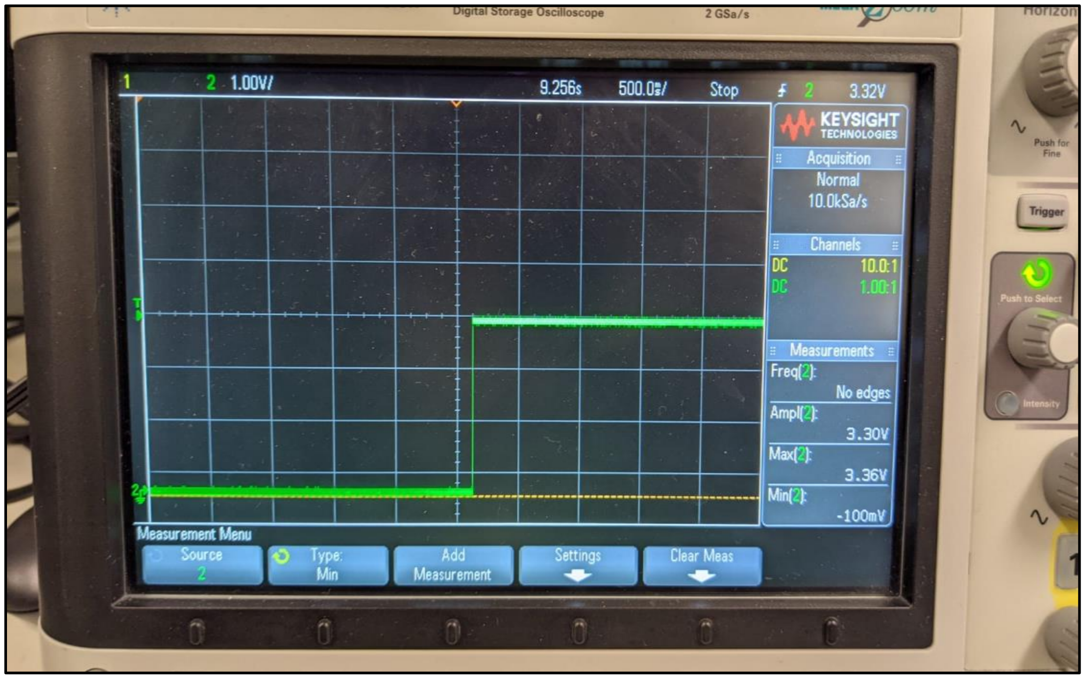
6. This last screenshot is for the sole purpose of timing where it can be seen that for a period of almost 3 second there is no I2C transaction being performed. This timing matches with the sleep timer with which the MCU was programmed. Hene, this verifies that the I2C peripheral is powered off for the 3 second duration to save power, disallowing any I2C transactions.

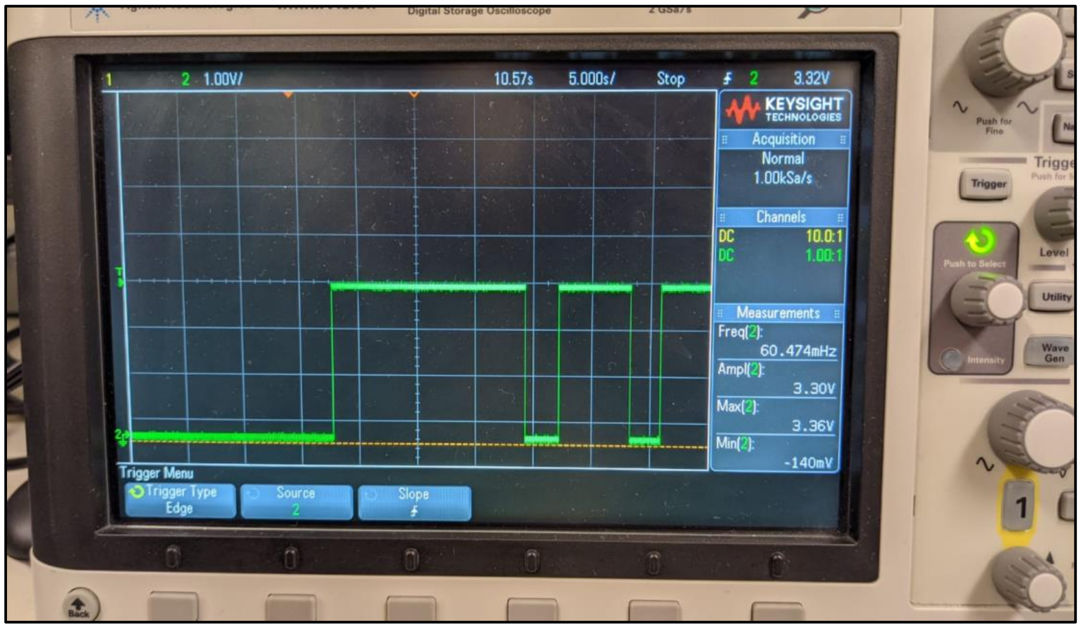


Signal quality analysis of UART signals

As mentioned in the project presentation, the Blue Gecko was able to detect the fingerprint sensor and we were able to transmit the command signals to the module. However, the responses from the module were not entirely read by the microcontroller.

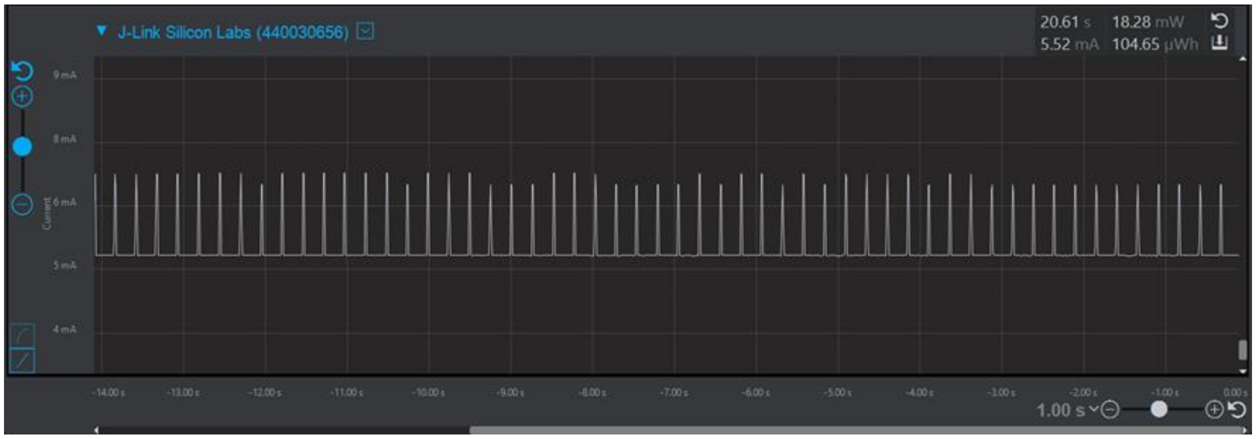
Below are the images of the UART signals. The signals were thoroughly distinguishable and as seen, the TTL levels were clearly signified by the oscilloscope. The minimum voltage recorded shows -100mV. This is due to the expected noise on the signal.



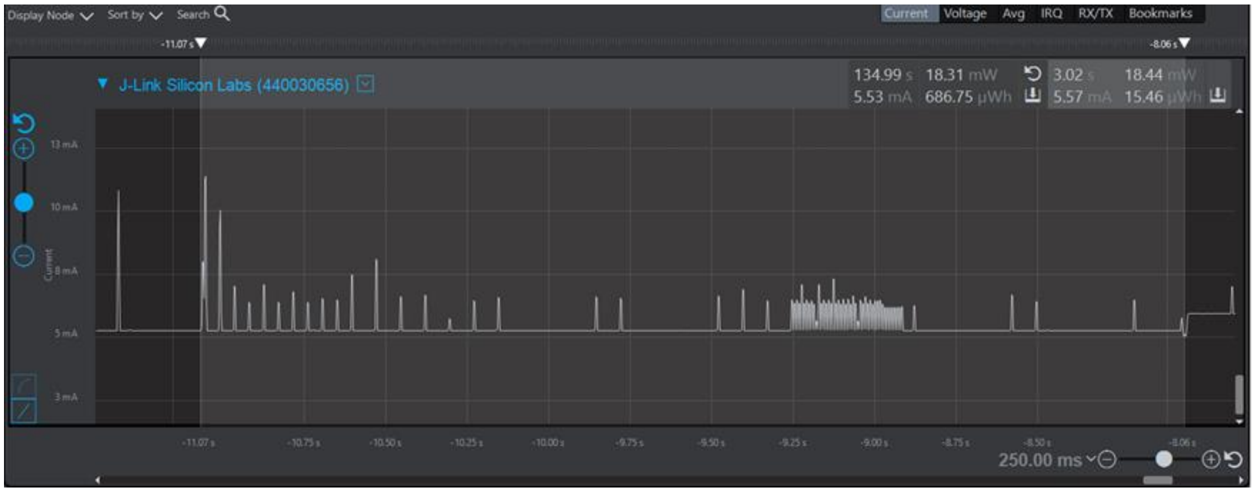


Current profile for various operation on the board

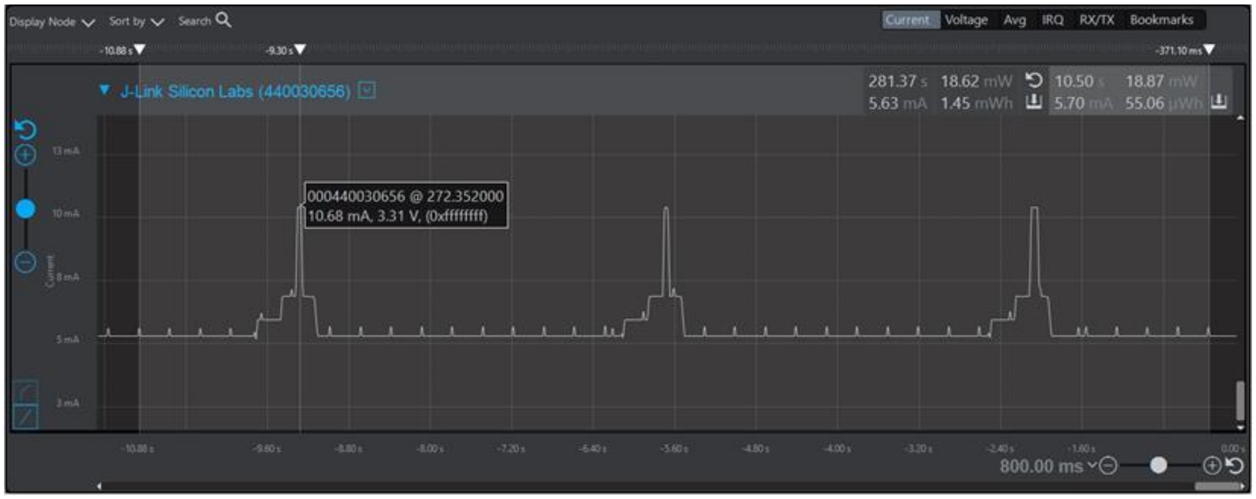
1. Timeline during Bluetooth advertisement



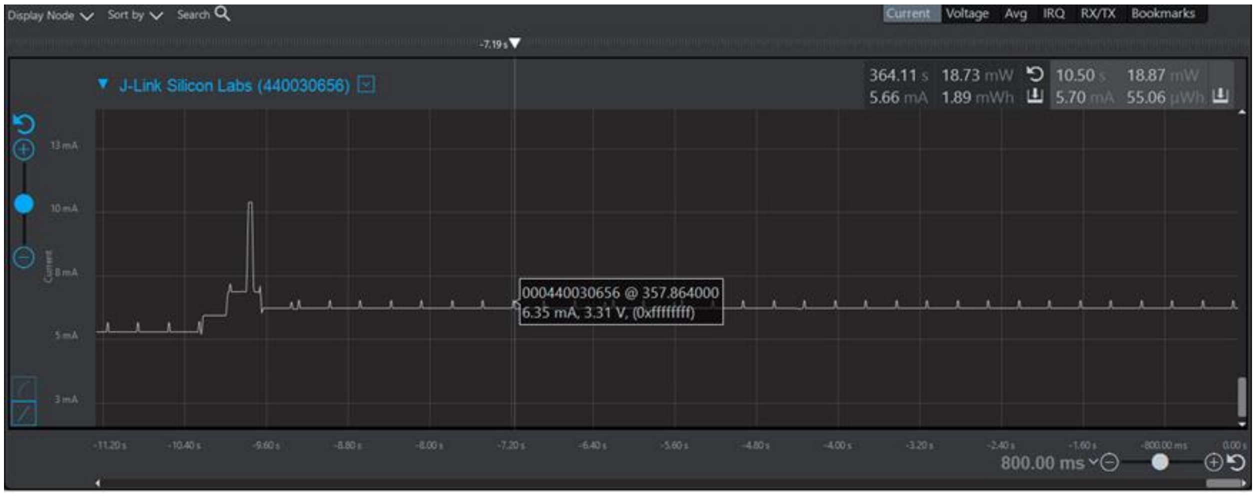
2. Timeline during Bluetooth connection initiation



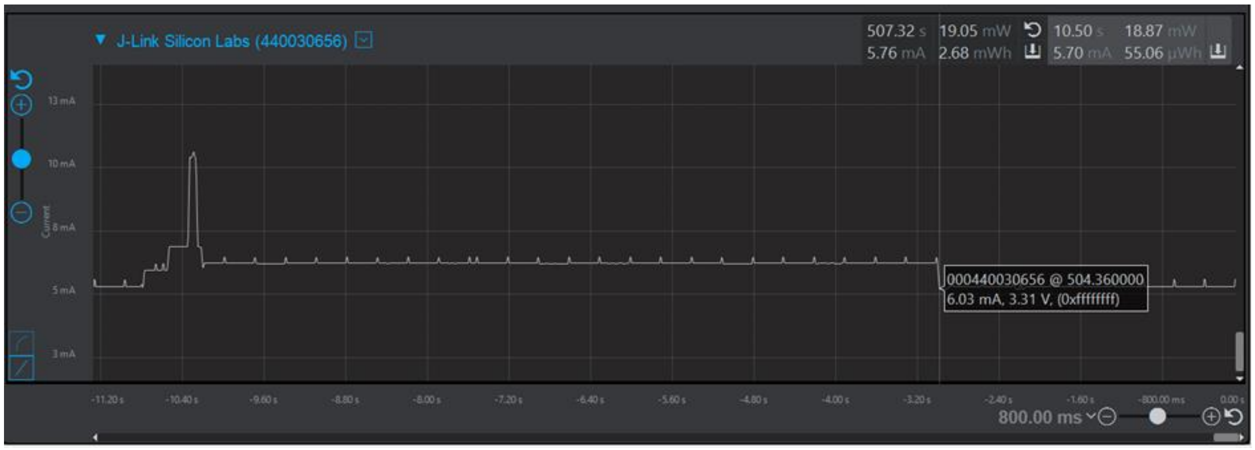
3. I2C operation in EM2 (IR sensor operation)



4. When a human presence is detected, the fingerprint sensor is turned on leading to an increase in the current consumption.


Display Node
Sort by
Search
Current
Voltage
Avg
IRQ
RX/TX
Bookmarks
J-Link Silicon Labs (440030656)
364.11 s 18.73 mW 10.50 s 18.87 mW
5.66 mA 1.89 mWh 5.70 mA 55.06 µWh
13 mA
10 mA
8 mA
5 mA
3 mA
Current
000440030656 @ 357.864000
6.35 mA, 3.31 V, (0xffffffff)
800.00 ms

5. The low current consumption here indicates that the fingerprint operation is complete, and the control is switched back to the main function.


J-Link Silicon Labs (440030656)
507.32 s 19.05 mW 10.50 s 18.87 mW
5.76 mA 2.68 mWh 5.70 mA 55.06 µWh
13 mA
10 mA
8 mA
5 mA
3 mA
Current
000440030656 @ 504.360000
6.03 mA, 3.31 V, (0xffffffff)
800.00 ms

Issues faced during project implementation

Some issues that we faced during project implementation are as follows:

1. We had trouble interfacing the UART sensor to the Blue Gecko. We believe that the issue was initially with the firmware. The UART functions based on various commands and the responses obtained from the fingerprint sensor. We attempted an interrupt-based UART and were unable to sense any response from the sensor although all our commands seemed to transmit. Having done that, we included a simpler polling-based implementation which would halt the system until an acknowledgement would be received. This halted the system after only 3 out of 8 bytes of acknowledgement were received. These implementations were based on LEUART. Upon coming across a source which mentioned LEUART might not support a 19200 bps baud rate, we attempted to integrate the sensor using USART. This was of no avail as well.
2. Interfacing the I2C sensor also took a significant amount of time since the sensor used for this system was quite different compared to the ones we had used before. This module consists of 4 integrated IR sensors and 1 temperature sensor. The module required us to completely read the 2 data registers for each of the sensors once before we could start reading the actual data. The module later "unprotects the data" and we would be able to read valid data. Figuring this out took a significant amount of time.
3. The fact that a wrong microcontroller was ordered via DigiKey came to light after we had already assembled our first board. In order to replace it with the right controller (EFR32BG13) we had to de-solder the existing one using a hot air gun and hand solder EFR32BG13. Thanks to SA Shamanth for the timely help!
4. Upon interfacing the fingerprint sensor using Arduino UNO, we tried to power the fingerprint sensor using our 5V power rail generated using the PMIC. We realized, doing this was pulling down the voltage to 3V. This discovery was made on the day of the demo and we could not venture further into this problem.
5. The Bluetooth app we designed was unable to pair to phones using Android 10 and 11.

Summary

Towards the end of the course, we were able to design a power management and processor circuit. Battery too could independently power the system. We were able to integrate an inverted-F antenna and design the circuit for Bluetooth communication. The system also consists of an integrated IR module on the board. The product successfully achieved BLE pairing with the phone. We were able to achieve low power mode by designing a state machine for BLE. We also designed an Android application to help communicate with the product.

We are able to take away a lot of new concepts and learnings post the completion of this course. Learning to work on Altium was a huge plus in terms of learning. We were not familiar with the tool prior to this course and now, we can confidently continue the learning. We are thankful to Prof. Randall Spalding and the SAs Kailey Shara and Shamanth Kudaligi Pranesharao for their support during the project reviews and implementation.

Lessons learned by Gitanjali Suresh

1. Always verify your PCB footprints.
2. Verify the parts before placing an order.
3. Always follow and update the Gantt chart.
4. Consider as many aspects of the project/product as possible while proposing the idea. The product needs to address and solve a problem.
5. Read the datasheet thoroughly for the hardware components and the firmware implementation.

Lessons learned by Sarayu Managoli

6. Verify calculations made for certain components before selection.
7. Check solder and connections even if they seem connected.
8. Have through-hole connectors for better and firm grip.
9. Plan the firmware implementation and start soon!
10. Start discussions regarding issues faced and seek help. Having a certain number of opinions helps in circling in the root cause of a problem.

Activities Accomplished - Past Week

Activity	Description	Status
Bulk Capacitance Selection	This has been done for the updated Lithium Ion battery selected.	Completed
Schematics update	Power management module and microcontroller schematic update	Completed
Footprint update	Footprints have been created for all the components in the schematic	Completed
Power calculation	Recalculated battery capacity	Completed
Component selection	IR, battery and other power and charge management components are being finalised	Completed
Component selection	Finalise components	Completed
Schematic review	Schematic review #2 has been completed. Final changes have been included in the update and a quick schematic review is planned before completion of the layout	Completed
Sensors and battery component	These components have been requested.	Completed
Layout update	Layout review #3 and #4 completion	Completed
Sparkfun and Mouser Component order	Order needs to be placed for the selected components	Completed
Digikey order	Order needs to be placed for the selected components	Completed
Firmware development	Firmware needs to be updated and tested for the sensors	Completed
Android App Setup	The initial design of the Android app	Completed
Board Assembly	Upon the reception of the PCB, we plan to immediately begin the assembly of the board	Completed
Testing and validation	Validation of the firmware and testing of the signals on the board	Completed

Verification and Validation Plan

The spreadsheet consisting the verification and validation plan has been added as a part of the ZIP file submitted for the assignment.

Project Schedule

The schedule is in the form of a Gantt chart and it can be found as a part of the ZIP file.