Fall 2021 ECEN 5823
Course Project Report

Section 1 - Project Proposal

Student Names

Peter Braganza: pebr2429@colorado.edu
Swapnil Ghonge: swgh6994@colorado.edu

Project Overview

Section Author: Peter Braganza

For the ECEN 5823 Course Project (CP) we have chosen to implement option 2 – client and server pair (BLE). We will be making a BLE system to judge the rashness of a person driving a vehicle. The system will have a client-server configuration where the server will acquire sensor data which will be periodically sent to the client which will analyze the data and judge the rashness of the driver and can take necessary action such as turn the vehicle off, if required.

In essence, the project will be a rashness monitoring system for vehicles and can be used by vehicle rental companies to track the manner in which rentals are being driven. This gives the company the data required to do a risk analysis of renting a car/bike to a particular customer. The system can be adapted for different land vehicles but for the purpose of proof of concept we will be modeling the system around a bike.

High Level Requirements

Section Author: Peter Braganza

1. There will be a server-client pair both of which will be blue gecko boards
2. The Server will get values from an IMU sensor and use accelerometer values to send to the client board.
3. The Server will also have an ultrasonic sensor which will sense distance of objects
    - 3.1. Then range of distance it will sense is 2m to 1m. within this range it will note for how long objects in proximity to the sensor.
    - 3.2. It will also implement a hysteresis loop in order to reduce false triggering of the sensor.
4. The Server will store IMU data and ultrasonic data in a buffer and will send indications/notifications to the client when a connection is open and bonded.
5. The Client will receive both IMU data and ultrasonic data and use this information to keep track of the number of violations made by the driver.
    - 5.1. If the driver drives in close proximity to another vehicle i.e. if the ultrasonic sensor value is between 1 to 2m for 5 or more seconds, then the client will count that as one distance violation.
    - 5.2. Similarly, if there are sudden jerk motion made by the driver i.e. if the accelerometer values indicate sudden change in motion beyond a threshold, then the client will count this as one motion violation.
6. Low Power Management:
    - 6.1. The server will be in the lowest possible sleep state as much as possible.

6.2. Server will connect to the client periodically and otherwise will close the connection and not be advertising.
6.3. All values from server will only be sent when connection is open.
6.4. Connection will close as soon as buffer is empty.

7. Bonding will be implemented on client and server and push buttons and LCD capabilities will be used to confirm bonding.
8. LED functionality on Server:
   8.1. In order to notify the diver that they have committed a distance or motion violation. LED0 and LED1 on the server's blue gecko will be used.
   8.2. LED0 will indicate when a distance violation has occurred.
   8.3. LED1 will indicate when a motion violation has occurred.
9. LCD Display functionality on Server:
   9.1. LCD will show Connection status: Advertising/Connected/Bonded/Sleeping
   9.2. If there is an active distance violation then the LCD must display the distance in meters.
   9.3. When a sudden jerk occurs then the LCD must show Motion Violation/Count of motion violation.
   9.4. Passkey and button to confirm.
10. LCD Display functionality on Client.
    10.1. LCD will show Connection status: Discovering/Connected/Bonded
    10.2. Distance violation count.
    10.3. Motion violation count.
    10.4. Passkey and button to confirm.

Non-implemented requirements:
These are requirements which would be implemented in a real-world project but we will not be implementing in this proof of concept project.

1. The server will be powered by a coin cell battery.
2. The client would be a phone which would use an application to perform client functionality and connect to the Internet to store data in a database
3. An online database would be present which would store user data and track users' motion and distance violation along with other data (e.g. speed and orientation of vehicle).

High Level Design

Section Author: Peter Braganza

The system will consist of a Client and a BLE server, both of which will be blue gecko boards. The server will host all sensors and will contain a GATT attribute database which will contain the respective sensor data. The server will contain an IMU sensor and ultrasonic sensor which will give 3 axis accelerometer data and distance. See the below table for sensor data that will be sent from server to client. LEDs and LCD will be used to indicate motion or distance violations. If the change in accelerometers values is large it will be indicated as a motion violation. If distance measured by ultrasonic sensor is lower than a threshold then it will be counted as a distance violation.

The Client will receive sensor data when a connection is established, and devices are bonded. The client will keep track of the number of distance and motion violations and display it on the LCD display. The client will be responsible for discovering a device and enabling indications/notifications. The client will not have any low power requirement however, low power management techniques will be implemented in the server.

Measurement	Units	Data Type	Valid/Allowed Values (Range)	Update Rate
Accelerometer X-axis	mg	uint16_t	-8 to +8g	0.5 Hz (1 bytes/sec)
Accelerometer Y-axis	mg	uint16_t	-8 to +8g	0.5 Hz (1 bytes/sec)
Accelerometer Z-axis	mg	uint16_t	-8 to +8g	0.5 Hz (1 bytes/sec)
Distance	m	uint16_t	0.02m to 4m	0.5 Hz (1 bytes/sec)

Table 1: Data Types

Wireless Communication

Section Author: Peter Braganza

The Server will contain two custom services one is Accelerometer Sensor service and the other is Ultrasonic Sensor Service. In addition to that the Accelerometer will have three characteristics for the X, Y, X accelerometer characteristics. Similarly, the ultrasonic sensor will have distance characteristic. The Client will enable indications for the characteristics and the server will send values to the client when enabled.

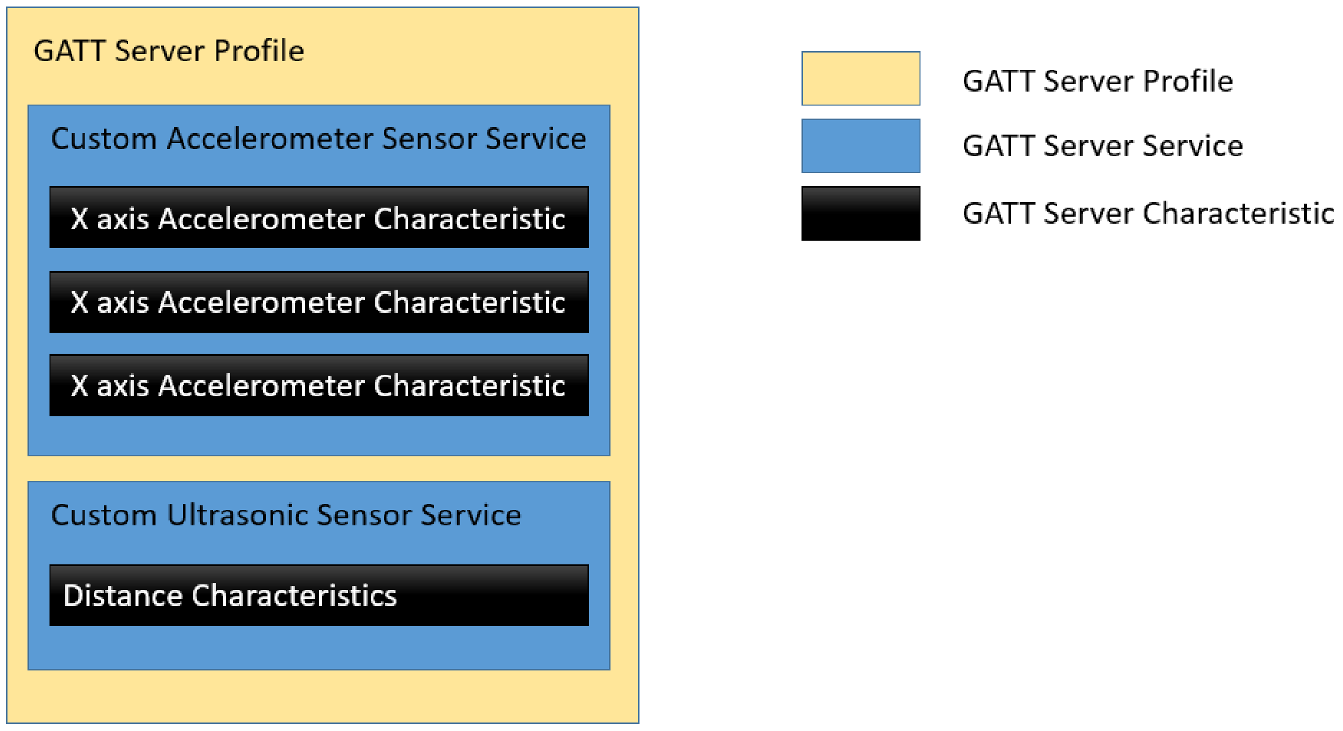

GATT Server Profile
Custom Accelerometer Sensor Service
X axis Accelerometer Characteristic
X axis Accelerometer Characteristic
X axis Accelerometer Characteristic
Custom Ultrasonic Sensor Service
Distance Characteristics
GATT Server Profile
GATT Server Service
GATT Server Characteristic

Figure 1: GATT Server

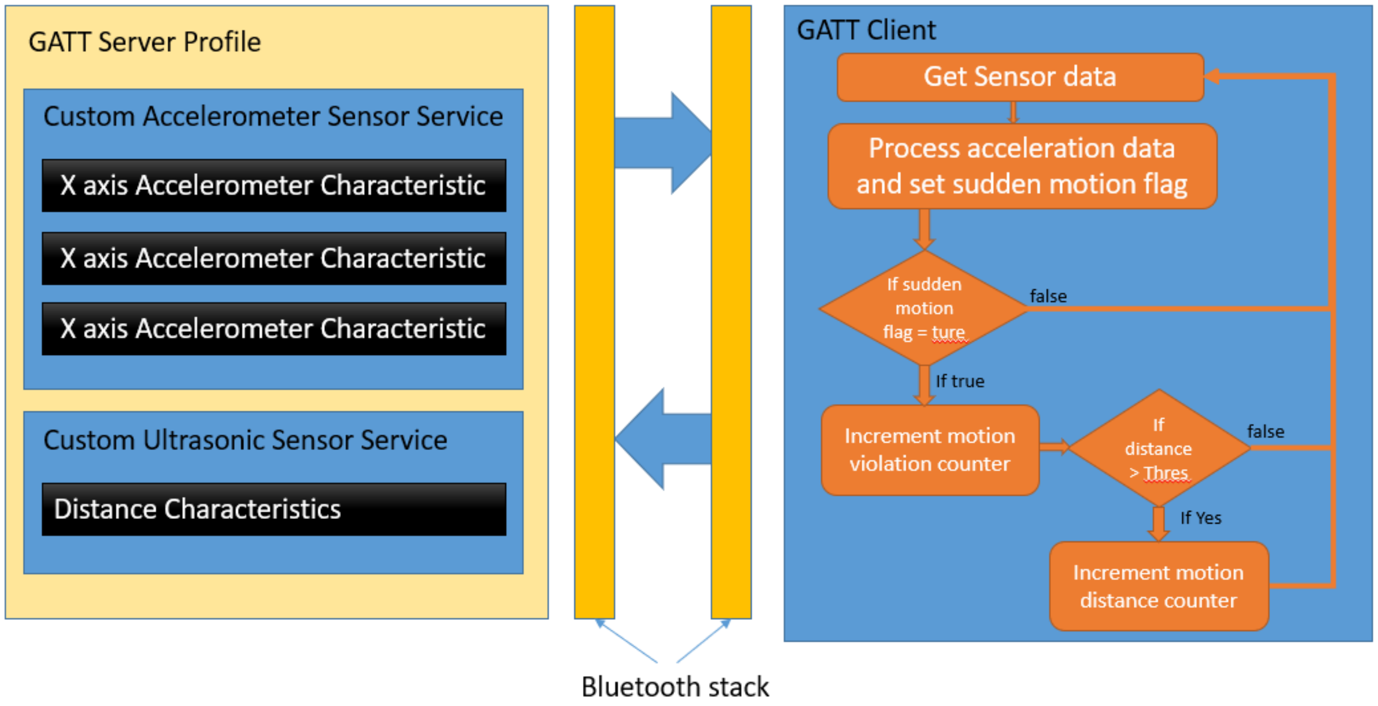

GATT Server Profile
Custom Accelerometer Sensor Service
X axis Accelerometer Characteristic
X axis Accelerometer Characteristic
X axis Accelerometer Characteristic
Custom Ultrasonic Sensor Service
Distance Characteristics
GATT Client
Get Sensor data
Process acceleration data and set sudden motion flag
If sudden motion flag = ture
false
If true
Increment motion violation counter
If distance > Thres
false
If Yes
Increment motion distance counter
Bluetooth stack

Figure 2: Client Functionality

Client will receive the custom characteristic indications/notifications when indications are enabled, and connection is open, and devices are bonded. Client will use the functionality shown above to determine if a motion or distance violation has occurred. Then a counter will be increased for the respective violations and display LCD will be updated.

Hardware functional Block Diagram

Section Author: Swapnil Ghonge

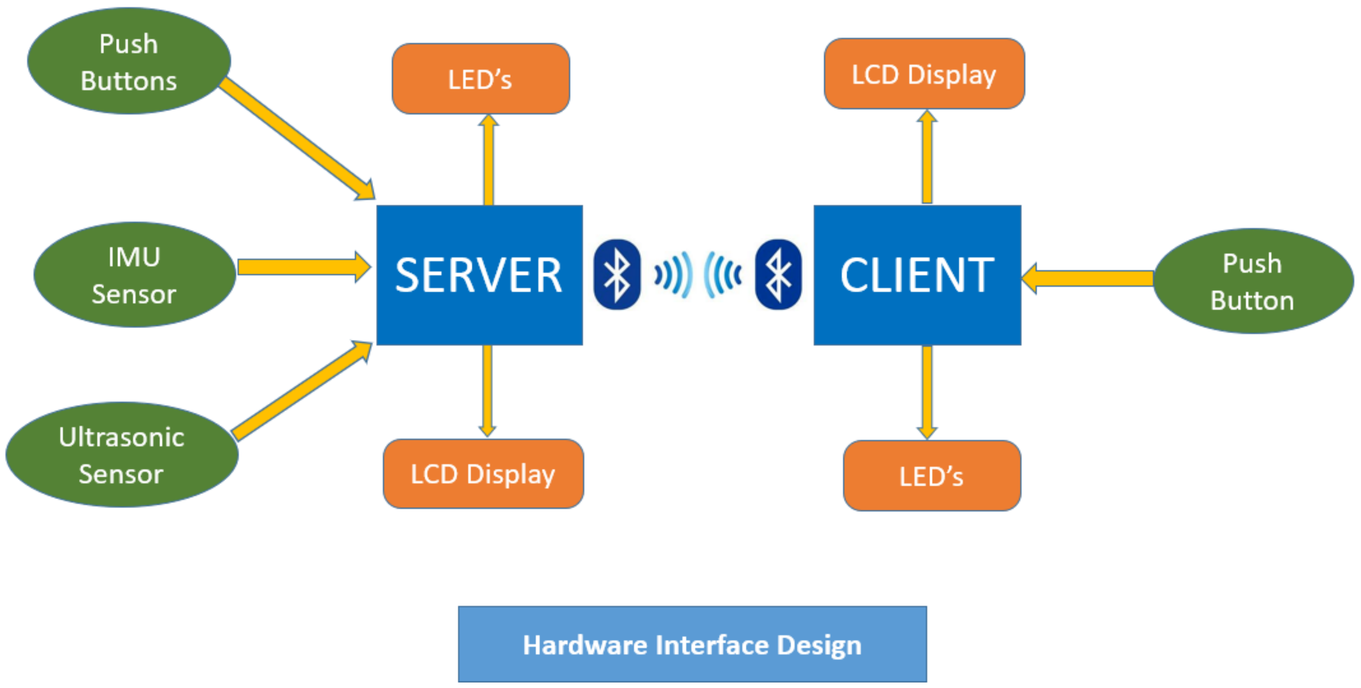


Figure 3: Functional Hardware block diagram

Figure 3 shows the functional block diagram of the system. It contains a server and client which have Bluetooth functionality. Both client and server are blue geckos. The server has an IMU sensor which gives accelerometer values for x, y and z-axis. The ultrasonic sensor will be used to acquire distance measurements in meters. Push buttons are used to confirm bonding. LEDs are used to indicate each time a violation occurs. LCD will display server data and other functionality explained in high level requirements and below is the LCD display section. The IMU sensor, Ultrasonic sensor and Push buttons will use hardware interrupts to trigger events. Additionally, Timers will be used to timer and trigger events using events.

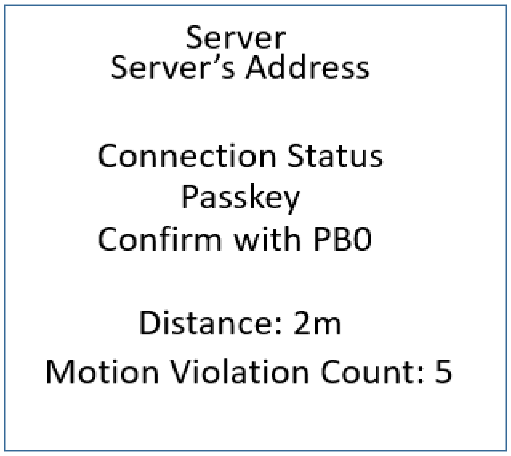
Client implements LEDs and LCD in a similar way. For details of client LCD display refer to below LCD display sections. The client does have any external hardware. It only uses a blue gecko for all functionalities.

LCD Design and UI Interface

Section Author: Swapnil Ghonge

Below figure show what data will be display on client and server respectively.
UI Description:

- The first row will be used to indicate if the board is client or server.
- The next line will contain the address of the Server/Client.
- Connection Status for Client: Discovering/Connected/Bonded
- Connection Status for Server: Advertising/Connected/Bonded/Sleeping
- Passkey and confirm with PB0 will only be used during bonding process.
- Server: Distance if violation occurs.
- Server: Motion Violation count
- Client: Distance and motion violation count.

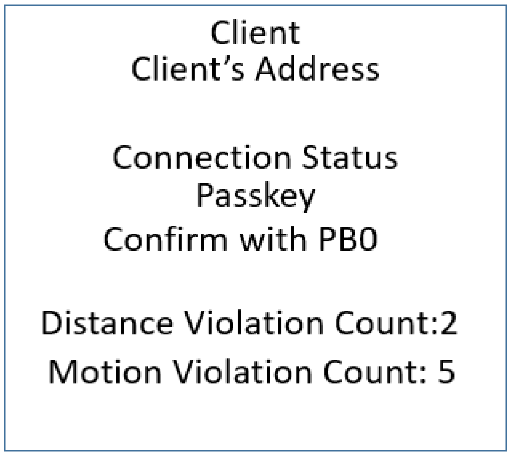


Server
Server's Address

Connection Status
Passkey
Confirm with PB0

Distance: 2m
Motion Violation Count: 5

Figure 6.1 Server LCD Display



Client
Client's Address

Connection Status
Passkey
Confirm with PB0

Distance Violation Count:2
Motion Violation Count: 5

Figure 6.2 Client LCD Display

Functional Software block diagram and Subsystem Summary

Section Author: Swapnil Ghonge and Peter Braganza

The following section explains the major software functionality of the system. The server will have multiple subsystems which will communicate to the Bluetooth stack. Each block is a functional subsystem and can be tested individually or with help of another subsystem. For Server, I^2C subsystem will be used to read and write values from IMU sensor. These sensor values are stored in a common buffer. The I^2C subsystem will use ISR routines to perform read write operations. Low Energy timer subsystem will be used for timing purposes of I^2C and periodic interrupt generation. LCD subsystem and LEDs are UI elements which will give users server information defined in LCD display section. The Bluetooth stack layer is a used to get events which will be used in a Bluetooth event handler function. Ultrasonic sensor subsystem contains functionality to measure if threshold level is crossed and then store data in buffer.

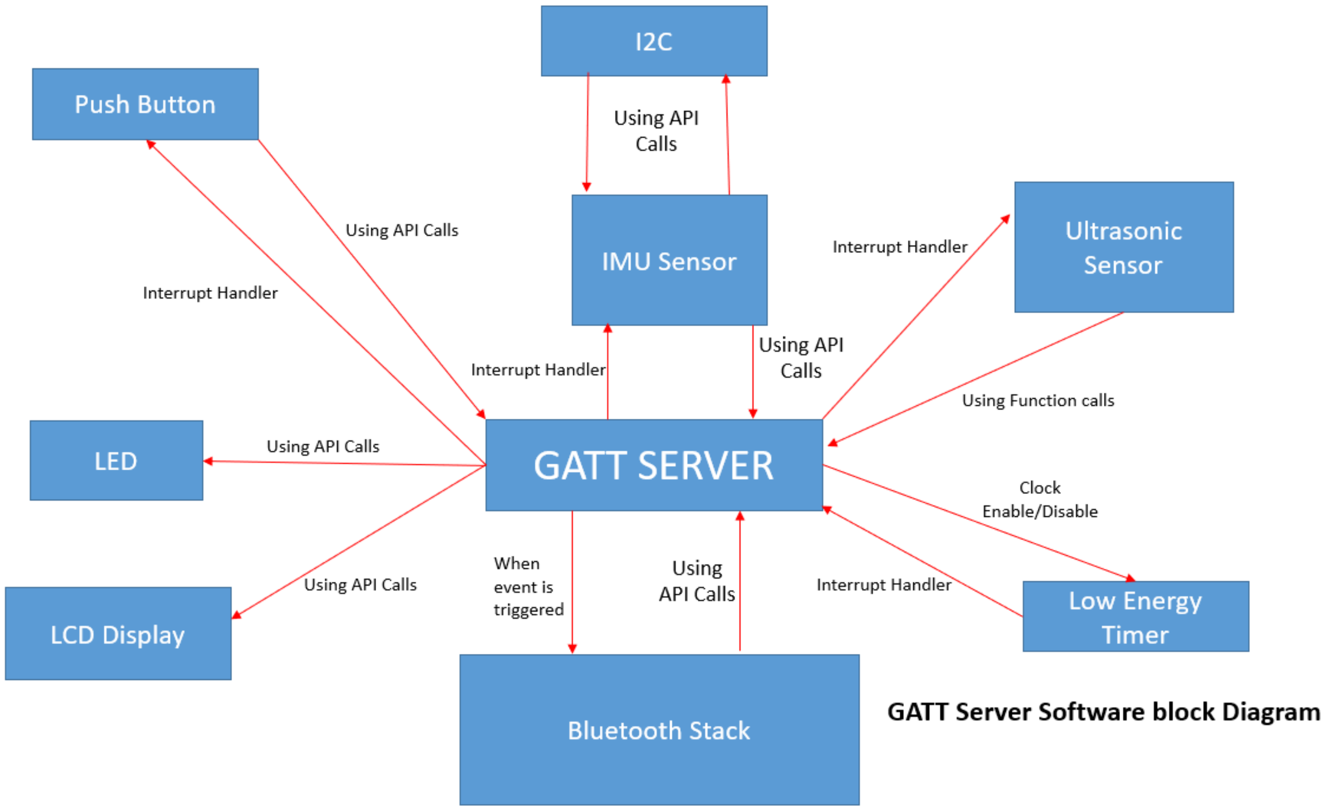


Figure 4: GATT Server Software block diagram

Client will also have a Bluetooth stack layer which will be used for event handling and to drive state machines. GPIO subsystem refers to the Push buttons which are connected to the GPIO pins. It will handle push button functionality using ISR routines. The LCD subsystem will be similar to server and is defined in detail in the LCD Display section.

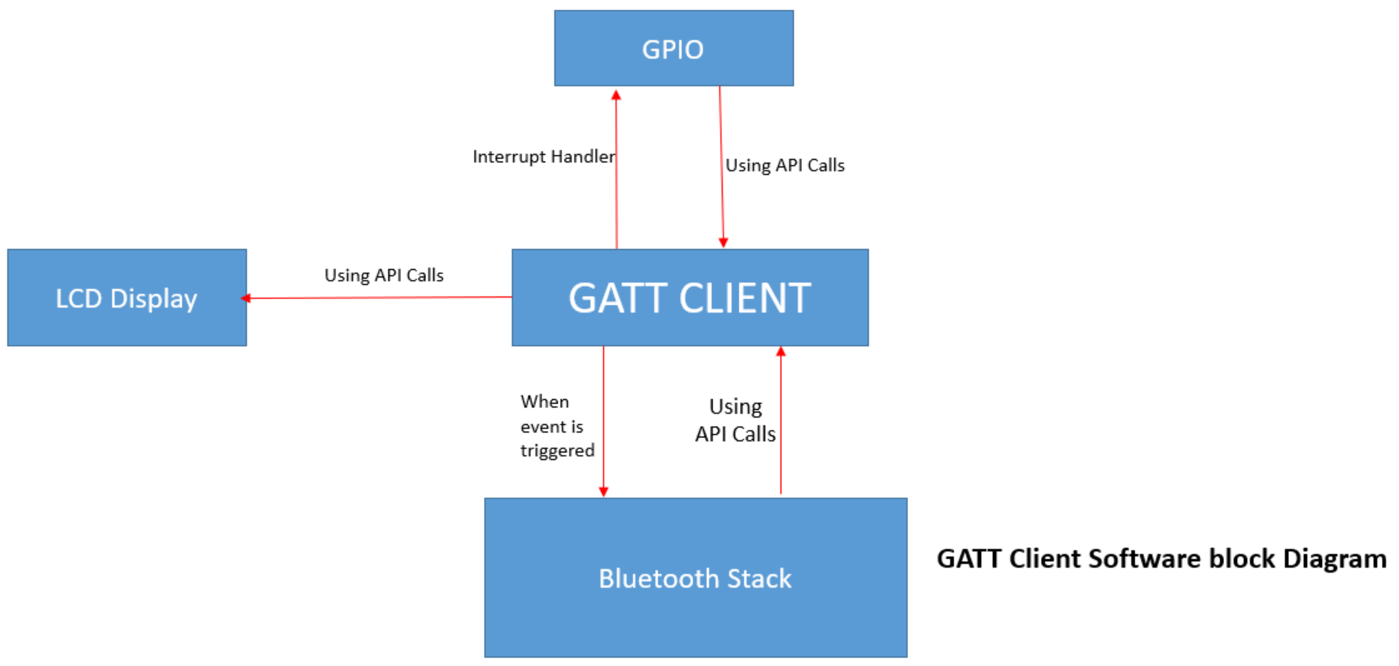


Figure 5: GATT Client Software block diagram

Division of labor

Section Author: Swapnil Ghonge

As we will be using sensor modules and silicon lab's blue gecko development board, a lot of the hardware considerations are already taken care off. Hence, we will be focusing a majority of efforts on the software implementation and low power energy management requirements of the project. Additionally, as the server contain majority functionality major focus will be towards server software code and testing various low power management techniques.

Data Flow

Section Author: Peter Braganza

Connection establishment flow between client and server is shown in figure below. The main aim is to keep the server in the lowest possible energy mode as long as possible and transmit data only when a threshold amount of data has been collected. This threshold limit is decided by a circular buffer implemented on the server. When there is no connection, the server will store sensor data in a circular buffer. Once the buffer is full the server will start advertising and connect to the client and send indications/notifications to the client. Once buffer is empty the connections is end and server will return back to lowest possible energy level.

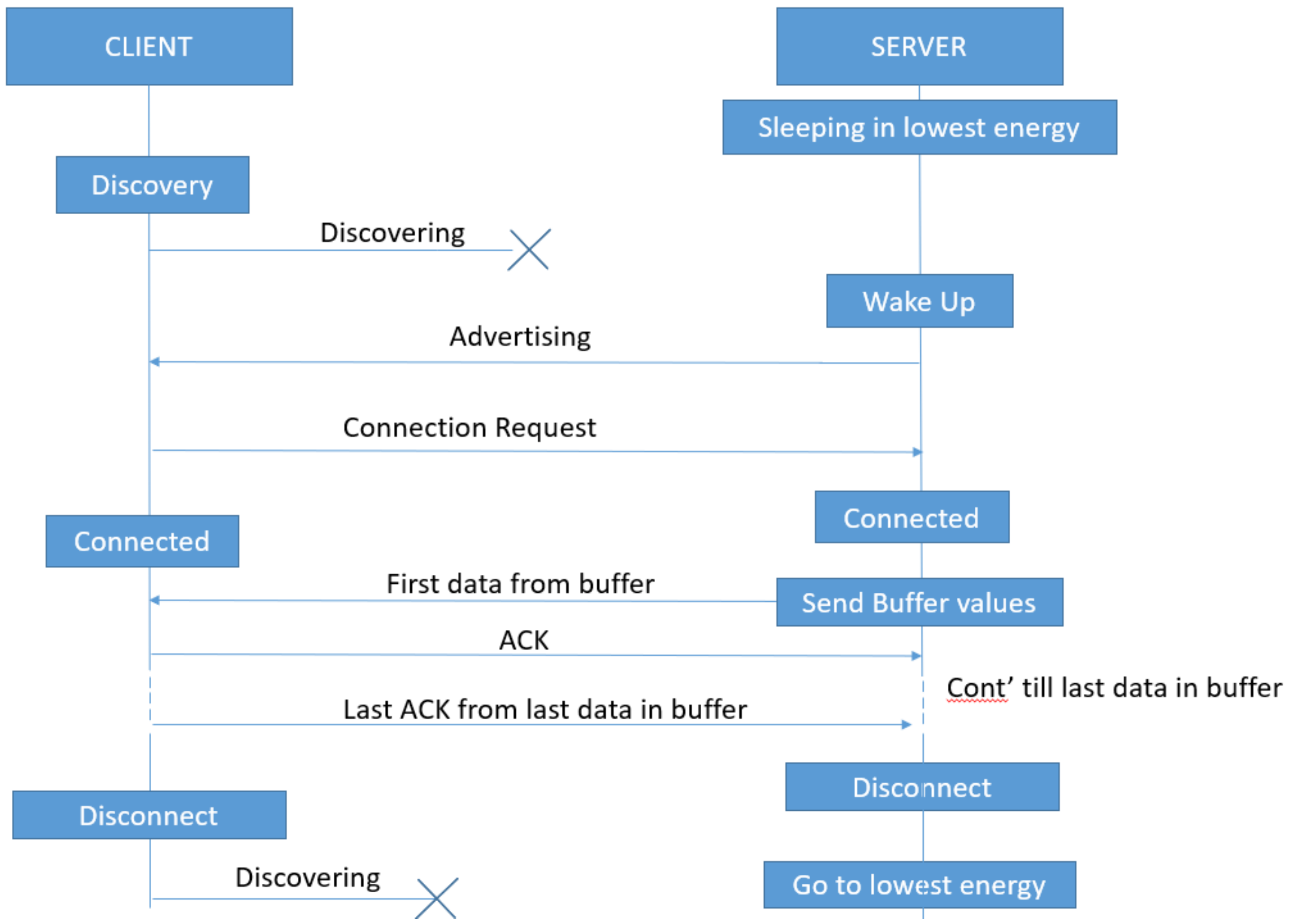


Figure 2: UML of connection sequence and data transfer

Note: This UML is applicable only once bonding has been established. If bonding has not been established, then there will an additional step after connections. There will be a bonding set of instructions will include exchange of passkey and confirming it on both devices.

Additionally, Sensor values will be received on server by using I^{2}C and interrupt handlers. LETIMERn will be used for timing purposes when load power management is used to power IMU sensor on and off and also ultrasonic.

Note : In low power management testing we will decide if load power management is required for sensors as they will be sampling data at a 0.5Hz and it may not be power efficient to keep turning the sensors on and off.

Test Plan

Section Author: Swapnil Ghonge

Each Subsystem will be tested as a standalone system or by using another subsystem along with it to check functionality. The table below shows the test plan proposed for timely testing of subsystems.

Test Plan Table				
Test Number	**Test Description**	**Date last run on**	**Test Result (Pass/Fail)**	**Notes**
1	Test to check IMU accelerometer sensor measurements are returning valid readings on serial monitor	11-15-2021		
2	Test to check ultrasonic sensor measurements are returning valid readings on serial monitor	11-15-2021		
3	Test to check if connection and bonding occurs and notifications are enabled followed by continuous transmission of accelerometer characteristic and distance characteristic values.	11-20-2021		
4	Ultrasonic sensor test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021		This will be useful for further low power management
5	Test to check is hysteresis is implemented correctly to avoid false triggering of Ultrasonic sensor	11-26-2021		
6	Accelerometer meter test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021		This will be useful for further low power management
7	LED and LCD Display test: A test to display accurate alert/warnings on LCD i.e. corresponding to distance violations and sudden motion violations.	11-29-2021		
8	Full functional test: At full functionality level that subsystems should be able to perform acceleration, distance measurement and should send out appropriate alerts to the user.	12-03-2021		
9	Low power energy management tests - To test power consumptions using the energy profiler.	12-05-2021		
10	Test to check best values for connection parameters after low power requirements and circular buffer have been implemented	12-07-2021		This will also include test for max disconnect time This is also part of low power energy management.

Proposed Schedule

Section Author: Peter Braganza

Task	Student(s) Responsible	Target Completion Date	Expected Completion Date
Getting Accelerometer Reading	Peter Braganza	-	Nov 17
Getting Ultrasonic Readings	Swapnil Ghonge	-	Nov 17
Setting up connection and bonding with client and server	Swapnil Ghonge	-	Nov 23
Sending Notifications/Indications to Client	Peter Braganza	-	Nov 27
LED and LCD Functionality	Swapnil Ghonge	-	Dec 3
Functional system testing (client and server)	Swapnil Ghonge and Peter Braganza	-	Dec 6
Low power requirements	Peter Braganza	-	Dec 8

GitHub repository URL(s) = https://github.com/CU-ECEN-5823/ecen5823-courseproject-PeterBraganza

Section 2 - Update 1

Status

Section Author: Swapnil Ghonge and Peter Braganza

For our project we have procured the following sensors and started to interface them both while learning about their functionality using data sheets and sample code. Links to the sensors and data sheets are given below. This completes the hardware component of our project as for the project we are only using module sensors and development boards.

Component name	QTY	Link	Data sheet
Ultrasonic Sensor HC-SR04	1	Ultrasonic Sensor Link	HCSR04 datasheet
Triple Axis Accelerometer Breakout – MPU-6050	1	Acceleromter Link	MPU6050 datasheet
Blue gecko board	2	Blue Gecko link	

As per our schedule we have completed interfacing the MPU6050 and close to interfacing the ultrasonic sensor. There is a slight delay in getting raw values from the ultrasonic sensor, however, that will not impede the completion of the final project as the next few tasks (setting up connections and bonding) have started being implemented. Hence, that provides us with a slight buffer to complete all other tasks on schedule. As mentioned before we started the pairing and bonding task which will use a significant portion of pervious assignment code. The main changes will be adding functionality where the client and server will pair and disconnect on a periodic basis while requiring bonding only happens on the first pairing process.

We identified that the main areas which will consume a significant portion of time are:

1. Getting Raw data from both sensors
2. Interpreting Raw data from MPU6050 to identify when a sudden jerk motion takes place
3. Implementing Low Power Requirements

Below Contains the completed task as per Week 2. Most task are in progress and are reflected in the wiki page of the below git link. It contains a more up to date progress tracking system.

Task	Student(s) Responsible	Target Completion Date	Expected Completion Date
Getting Accelerometer Reading	Peter Braganza	Nov 16	Nov 17
Getting Ultrasonic Readings	Swapnil Ghonge	-	Nov 17
Setting up connection and bonding with client and server	Swapnil Ghonge	-	Nov 23
Sending Notifications/Indications to Client	Peter Braganza	-	Nov 27

LED and LCD Functionality	Swapnil Ghonge	-	Dec 3
Functional system testing (client and server)	Swapnil Ghonge and Peter Braganza	-	Dec 6
Low power requirements	Peter Braganza	-	Dec 8

Design Changes

Section Author: Peter Braganza

When selecting the IMU sensor we decided to use the ADXL345 sensor. However, we changed that to the MPU6050 as the amount of documentation on how to use the sensor is better explained and has more example code available. Additionally, I changed the implementation of how we were originally going to get values from the sensor. We were going to read individual X, Y, Z accelerometer registers. Now, we will be reading values from the in-built 1024byte FIFO buffer. This will help with Low power requirement as the I2C bus can be disabled for a longer period. Additionally, reading from a buffer provides more accurate readings. For example, when getting values for X-axis accelerometer there are two registers ACCEL_X_H and ACCEL_X_L (1 byte size each). If we issue a command to read ACCEL_X_H after read reading ACCEL_X_L we will get the most current value of ACCEL_X_H. This is not correct as both ACCEL_X_H/L should be read at the same time to get correct readings. The same can be said for Y and Z axis. We have added another test in the test plan to made accommodations for this requirement.

Test Plan

Section Author: Peter Braganza

We have added a new test case and fully completed one test. One test is delayed but majority of software functionality has been implemented for that test. Several Test are also progress. The test completion percentage about 9% in terms of testing. However, as identified before these are some of the major time consuming task and the next few task should act as a time buffer to help increase the testing completion percentage.

Test Plan Table

Test Number	Test Description	Date last run on	Test Result (Pass/Fail)	Notes
1	Test to check IMU accelerometer sensor measurements are returning valid readings on serial monitor	11-15-2021	PASS	Completed on Nov 16
2	Test to check ultrasonic sensor measurements are returning valid readings on serial monitor	11-15-2021		In Progress
3	Test to check buffer data in MPU6050 is read correctly	11-20-2021		New Test added after design change Also In Progress

4	Test to check if connection and bonding occurs and notifications are enabled followed by continuous transmission of accelerometer characteristic and distance characteristic values.	11-20-2021		In Progress
5	Ultrasonic sensor test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021		This will be useful for further low power management
6	Test to check is hysteresis is implemented correctly to avoid false triggering of Ultrasonic sensor	11-26-2021		
7	Accelerometer meter test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021		This will be useful for further low power management
8	LED and LCD Display test: A test to display accurate alert/warnings on LCD i.e. corresponding to distance violations and sudden motion violations.	11-29-2021		
9	Full functional test: At full functionality level that subsystems should be able to perform acceleration, distance measurement and should send out appropriate alerts to the user.	12-03-2021		
10	Low power energy management tests - To test power consumptions using the energy profiler.	12-05-2021		
11	Test to check best values for connection parameters after low power requirements and circular buffer have been implemented	12-07-2021		This will also include test for max disconnect time This is also part of low power energy management.

Challenges

Section Author: Swapnil Ghonge

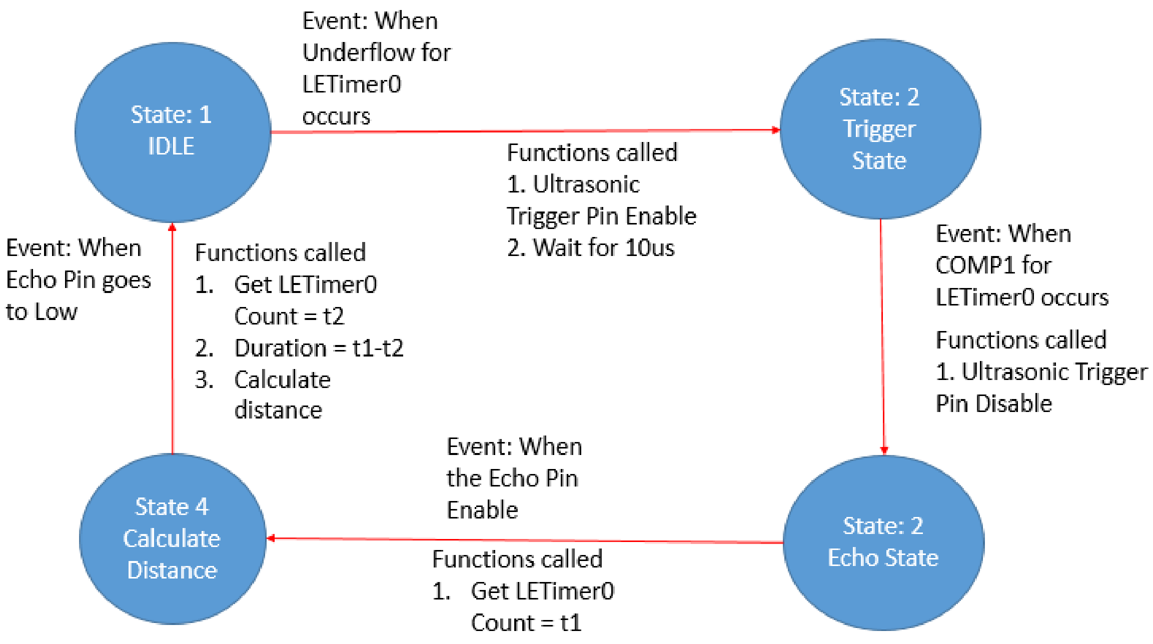
During the course of 1 week of integration of sensors with Blue Gecko Board, we came across several challenges. The first challenge we faced was understanding the sensors working and finding relevant documentation on the web especially for accelerometer ADXL345. Due to the unavailability of appropriate documentation, we shifted from ADXL345 accelerometer to MPU-6050 triple axis Gyro based accelerometer. The second challenge we faced for developing the library functions for ultrasonic sensor. The implemented state machine code for the ultrasonic sensor got stuck in the middle The issue we debugged by placing breakpoints in the IRQ handler file. We found out that the issue was that we were not getting interrupts from the GPIO echo pin which was supposed to come when it becomes off.

As previously mentioned, reading values from the MPU6050 was not a straightforward task. It requires much more initialization code than previously anticipated. We had to configure the sensor according to our use case. Additionally, the configuration of the sensor changed as we will now be using a buffer to get values.

Updates

Section Author: Swapnil Ghonge

Ultrasonic Sensor State Machine:


State: 1
IDLE
Event: When Underflow for LETimer0 occurs
State: 2
Trigger State
Functions called
1. Ultrasonic Trigger Pin Enable
2. Wait for 10us
Event: When COMP1 for LETimer0 occurs
Functions called
1. Ultrasonic Trigger Pin Disable
State: 2
Echo State
Event: When the Echo Pin Enable
Functions called
1. Get LETimer0 Count = t1
State 4
Calculate Distance
Event: When Echo Pin goes to Low
Functions called
1. Get LETimer0 Count = t2
2. Duration = t1-t2
3. Calculate distance

GitHub repository URL(s) =https://github.com/CU-ECEN-5823/ecen5823-courseproject-PeterBraganza

Section 3 - Update 2

Status

Section Author: Swapnil Ghonge and Peter Braganza

The goal for this week was to complete the all-software development on GATT Server and Client. Another of our goals was to complete the backlog we had before w.r.t. getting accurate value from ultrasonic sensor and send the indications to EFR Connect App with circular buffer. Additionally, this week our target was to calibrate the raw values that we were getting from the accelerometer and integrate.

We have completed integrating the ultrasonic sensor into the system. Which includes getting data from the ultrasonic sensor, converting it to distance and counting distance violations, updating it to the gatt server and sending to the client if indications are enabled and when bonded. The state machine for the ultrasonic sensor was also updated so that only one LETIMER needs to be used when both sensors are integrated together. State machine diagrams are shown in a later section. Additional, Majority of the client code is completed for the entire system. As of now distance violation count and pseudo data for the motion violation count (will get actual count after converting raw accelerometer data into data used to determine jerk motion) was used to check if the client side of the system is functional. Bonding between the client and server is done by attempting to read distance violation (like assignment 9).

For the MPU6050 sensor I was able to configure the sensors internal FIFO buffer and read values from it. However, the rate at which it updates is high and it would be more energy efficient to use a soft timer to trigger when to sample readings at a fixed interval. There were other issues also encountered with the MPU6050 which is in the challenges section below.

Below Contains the completed task as per Week 3.

Task	Student(s) Responsible	Target Completion Date	Expected Completion Date
Getting Accelerometer Reading	Peter Braganza	Nov 16	Nov 17
Getting Ultrasonic Readings	Swapnil Ghonge	Nov 29	Nov 17
Setting up connection and bonding with client and server	Swapnil Ghonge and Peter Braganza	Nov 30	Nov 23
Sending Notifications/Indications to Client	Peter Braganza	Dec 2	Nov 27
LED and LCD Functionality	Swapnil Ghonge	Dec 3	Dec 3
Functional system testing (client and server)	Swapnil Ghonge and Peter Braganza	-	Dec 6
Low power requirements	Peter Braganza	-	Dec 8

Design Changes

Section Author: Peter Braganza

Previously for the MPU6050 we were going to use the internal FIFO buffer of the sensor to reduce the time that the Master's (Gecko board) MCU must be awake. However, the rate at which the FIFO buffer overflows is high and does not give us a good reason to use it. The rate at which the accelerometer samples values is 1KHz and cannot be reduced, hence the FIFO buffer fills up is high. Instead using a soft timer to trigger reading accelerometer values would be more energy efficient. We will be sampling data from the MPU at 250ms and take about 10 samples to data each time for processing purposes. As the wake-up time for the MPU6050 is 100ms it may not be valuable to turn it off using Load Power management techniques.

Also, we updated which values we will be sending to the client. As sending the count of each violation satisfies the requirements the values which will be sent will change as shown below

Measurement	Units	Data Type	Valid/Allowed Values (Range)	Update Rate
Motion Violation count	-	uint8_t	0 to 255	1/3 Hz (1/3 bytes/sec)
Distance Violation count	-	uint8_t	0 to 255	1/3 Hz (1 bytes/sec)

Based on the changes made to the data types the Gatt server will also change as shown below

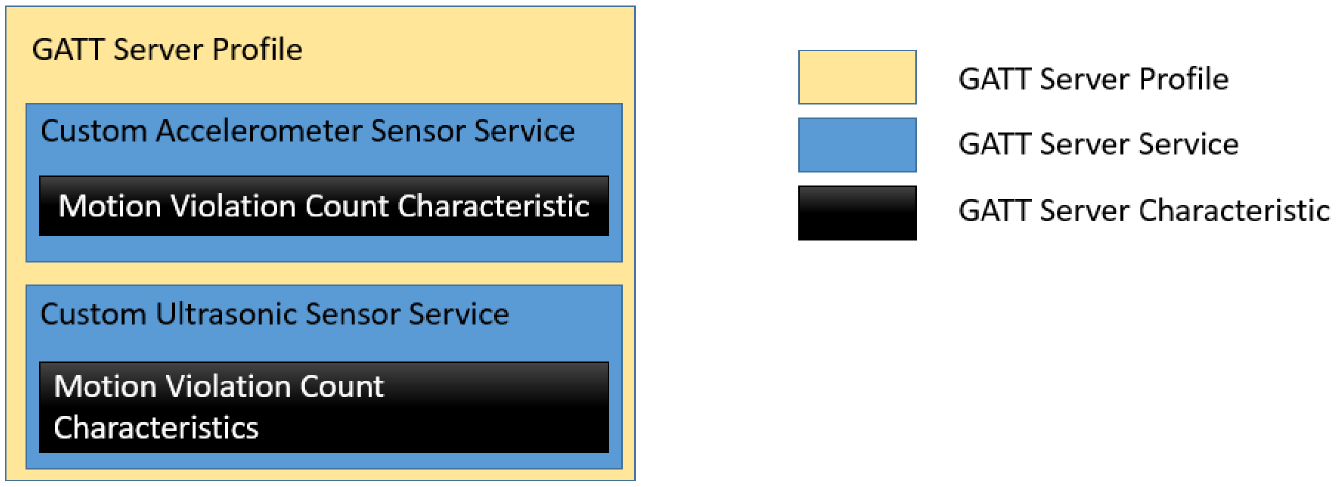
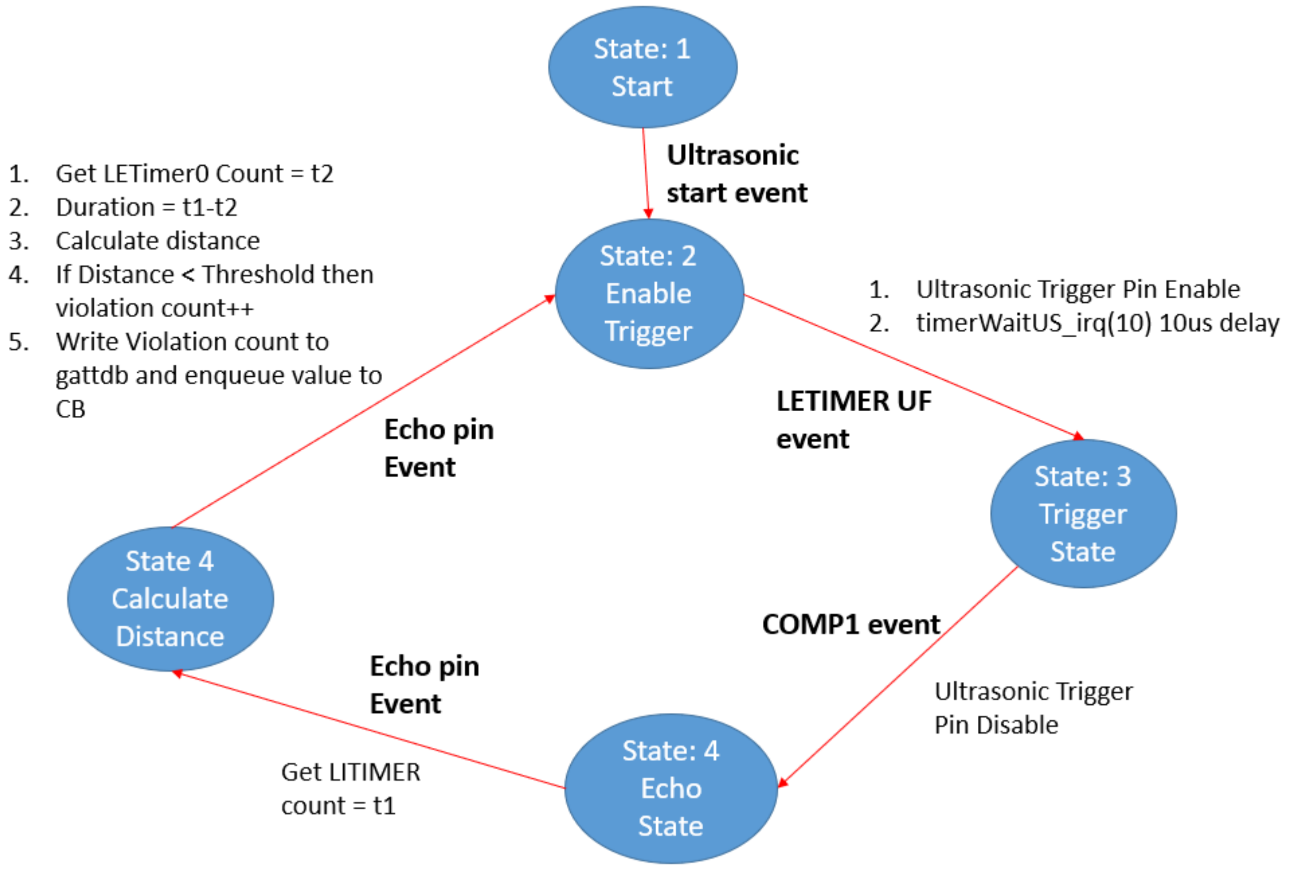

GATT Server Profile
Custom Accelerometer Sensor Service
Motion Violation Count Characteristic
Custom Ultrasonic Sensor Service
Motion Violation Count Characteristics
GATT Server Profile
GATT Server Service
GATT Server Characteristic

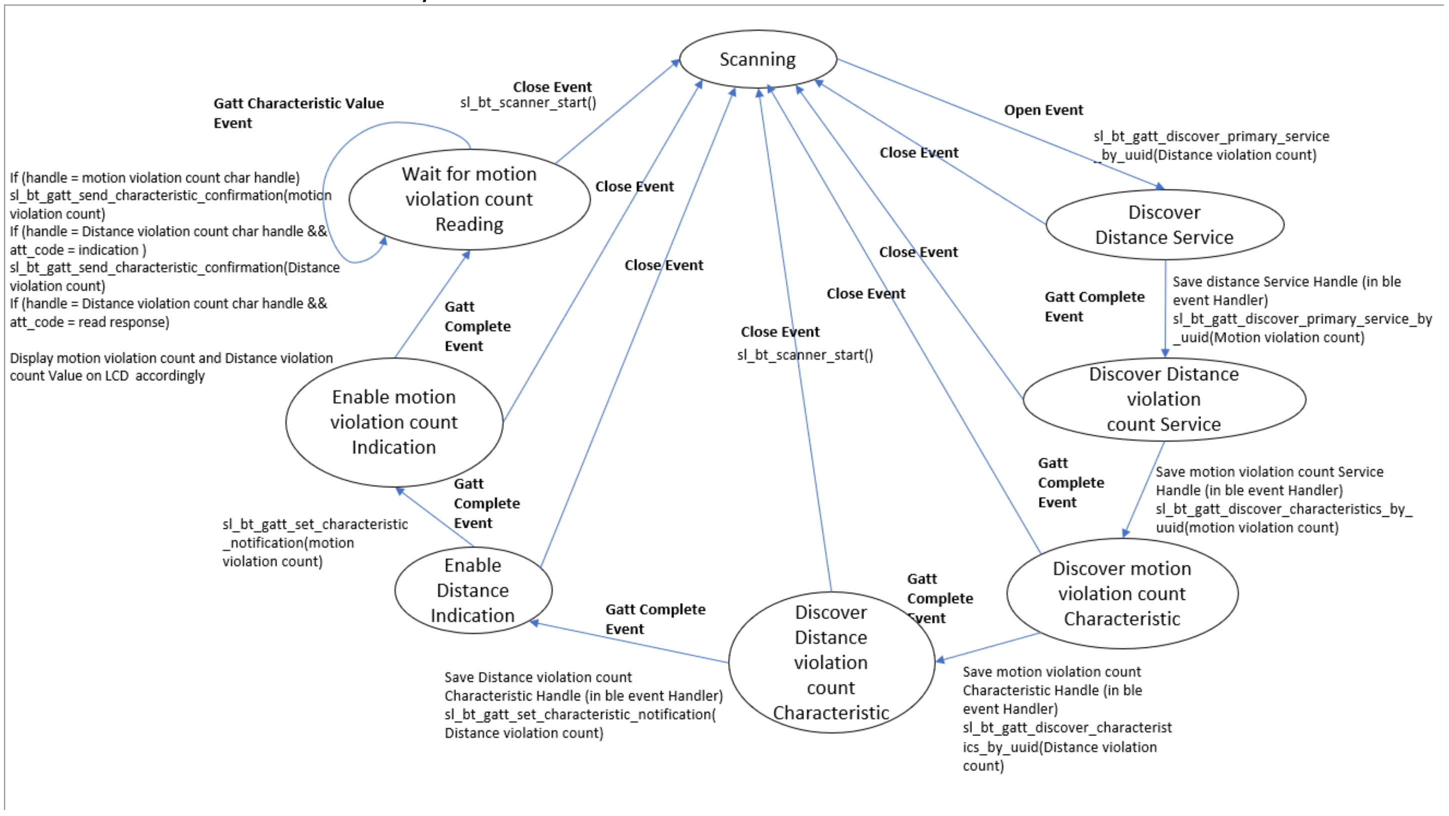
Figure 1. Updated Gatt Server

Additionally, the state machine for the ultrasonic sensor is updated as shown below.

Ultrasonic Sensor Measurement State Machine



Below shows the client discovery state machine.


Scanning
Close Event
sl_bt_scanner_start()
Gatt Characteristic Value Event
If (handle = motion violation count char handle)
sl_bt_gatt_send_characteristic_confirmation(motion violation count)
If (handle = Distance violation count char handle && att_code = indication)
sl_bt_gatt_send_characteristic_confirmation(Distance violation count)
If (handle = Distance violation count char handle && att_code = read response)
Display motion violation count and Distance violation count Value on LCD accordingly
Wait for motion violation count Reading
Close Event
Close Event
Gatt Complete Event
Enable motion violation count Indication
Gatt Complete Event
sl_bt_gatt_set_characteristic_notification(motion violation count)
Enable Distance Indication
Gatt Complete Event
Save Distance violation count Characteristic Handle (in ble event Handler)
sl_bt_gatt_set_characteristic_notification(Distance violation count)
Discover Distance violation count Characteristic
Gatt Complete Event
Save motion violation count Characteristic Handle (in ble event Handler)
sl_bt_gatt_discover_characteristics_by_uuid(Distance violation count)
Discover motion violation count Characteristic
Gatt Complete Event
Save motion violation count Service Handle (in ble event Handler)
sl_bt_gatt_discover_characteristics_by_uuid(motion violation count)
Discover Distance violation count Service
Gatt Complete Event
Save distance Service Handle (in ble event Handler)
sl_bt_gatt_discover_primary_service_by_uuid(Motion violation count)
Discover Distance Service
Open Event
sl_bt_gatt_discover_primary_service_by_uuid(Distance violation count)
Close Event
Close Event
Close Event
Close Event
sl_bt_scanner_start()

Hardware Block Diagram Update

Section Author: Swapnil Ghonge

Implemented Hardware Block Diagrams are shown below.

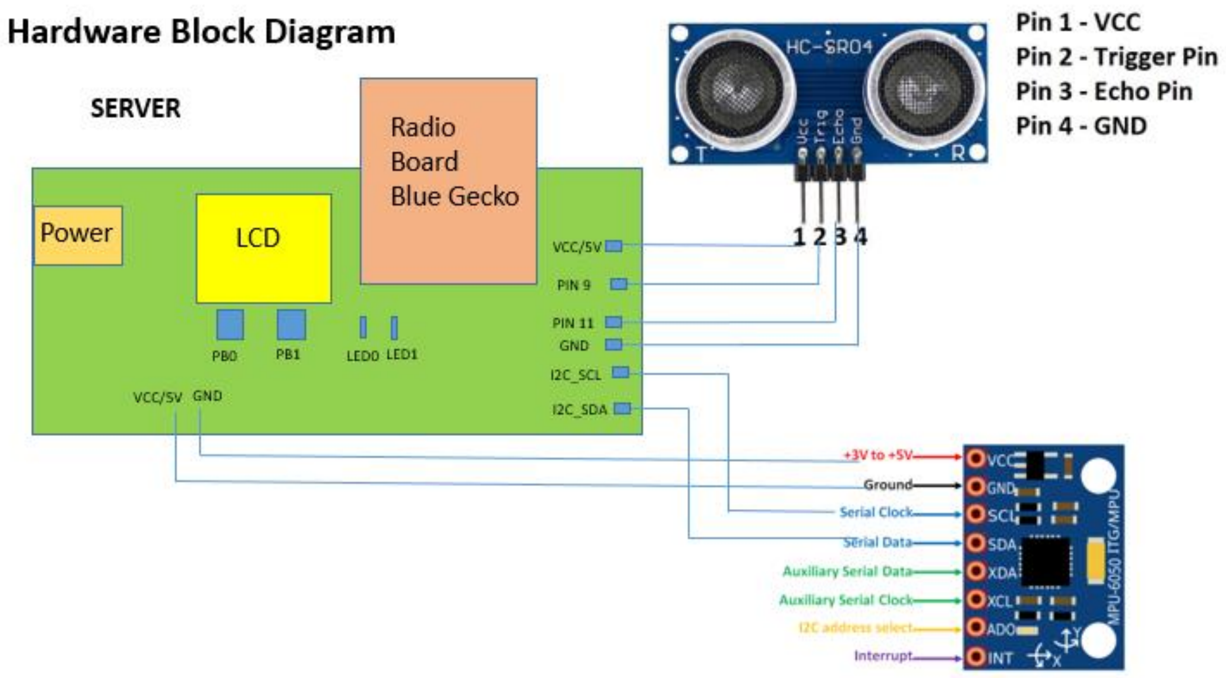


Figure 2. Server Hardware block diagram

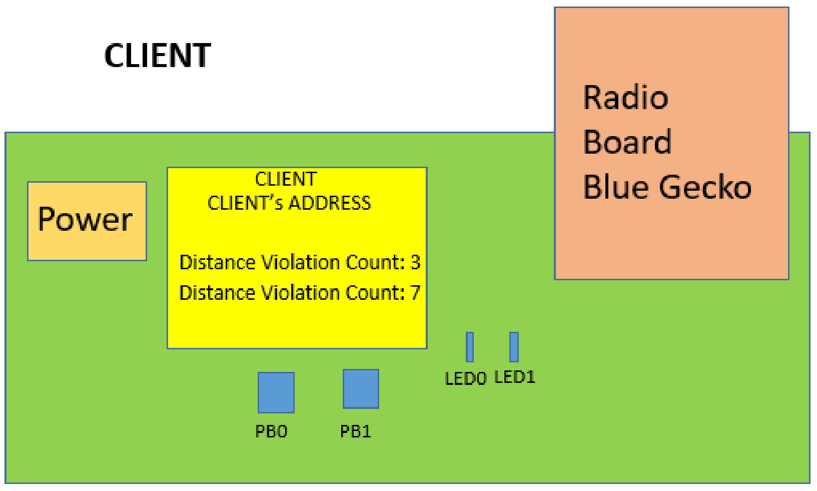


Figure 3. Client Hardware block diagram

Test Plan

Section Author: Swapnil Ghonge

At this stage 65% of the Project is completed.

Test Plan Table

Test Number	Test Description	Date last run on	Test Result (Pass/Fail)	Notes
1	Test to check IMU accelerometer sensor measurements are returning valid readings on serial monitor	11-15-2021	PASS	Completed on Nov 16
2	Test to check ultrasonic sensor measurements are returning valid readings on serial monitor	11-15-2021	PASS	
3	Test to check buffer data in MPU6050 is read correctly	11-20-2021	PASS	
4	Test to check if connection and bonding occurs and notifications are enabled followed by continuous transmission of accelerometer characteristic and distance characteristic values.	11-20-2021	PASS	
5	Ultrasonic sensor test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021	PASS	This will be useful for further low power management
6	Test to check is hysteresis is implemented correctly to avoid false triggering of Ultrasonic sensor	11-26-2021	PASS	
7	Accelerometer meter test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021		In Progress This will be useful for further low power management.
8	LED and LCD Display test: A test to display accurate alert/warnings on LCD i.e. corresponding to distance violations and sudden motion violations.	11-29-2021	PASS	Implementation of LED is pending
9	Full functional test: At full functionality level that subsystems should be able to perform acceleration, distance measurement and should send out appropriate alerts to the user.	12-03-2021		
10	Low power energy management tests - To test power consumptions using the energy profiler.	12-05-2021		

11	Test to check best values for connection parameters after low power requirements and circular buffer have been implemented	12-07-2021		This will also include test for max disconnect time This is also part of low power energy management.

Challenges:
Section Author: Swapnil Ghonge and Peter Braganza

In this week, the challenges we faced a challenge that included both software and hardware. In the hardware, we faced issues with the hardware loose connection of sensor and Gecko Board. Due to this reason, we struggled with the GPIO interrupt, for which our state machine was not progressing. To debug this issue, we checked first the software part. We have added LOG statement to check the interrupt flag is set or not. It came to our notice that the GPIO flag is not getting set, that's how concluded that there is a loose connection in hardware part. We fixed the issue by changing the wires and state machine progressed and we started getting the value

For the MPU6050 understanding how the on-board FIFO buffer works took time and documentation for the same fell short on explaining the updating/sampling rate. However, Once configured correctly, we found that the rate at which it updates values is very high and using the FIFO would not save power as the I2C bus would be active for long duration of time and very frequently.

GitHub repository URL(s) = https://github.com/CU-ECEN-5823/ecen5823-courseproject-PeterBraganza

Section 4 - Final Report

Status

Section Author: Peter Braganza

We completed most tasks planned and caught up with completion of the project toward the end. There was one feature which we were not able to complete which was to connect and disconnect the client and server after bonding. Further reasons and explanation regarding this are given in the below sections.

Task	Student(s) Responsible	Target Completion Date	Expected Completion Date
Getting Accelerometer Reading	Peter Braganza	Nov 16	Nov 17
Getting Ultrasonic Readings	Swapnil Ghonge	Nov 29	Nov 17
Setting up connection and bonding with client and server	Swapnil Ghonge and Peter Braganza	Nov 30	Nov 23
Sending Notifications/Indications to Client	Peter Braganza	Dec 2	Nov 27
LED and LCD Functionality	Swapnil Ghonge	Dec 3	Dec 3
Functional system testing (client and server)	Swapnil Ghonge and Peter Braganza	Dec 7	Dec 6
Low power requirements	Peter Braganza	Dec 8	Dec 8

Requirements Completion

Section Author: Peter Braganza

We have completed most requirements as proposed in the project proposal except requirement 8.2. in which we proposed to disconnect and reconnect the server and client to keep the radio off for the maximum amount of time. We were able to disconnect and reconnect both devices however, after the devices reconnected again indications were unable to be sent due to bonding between the devices not being detected.

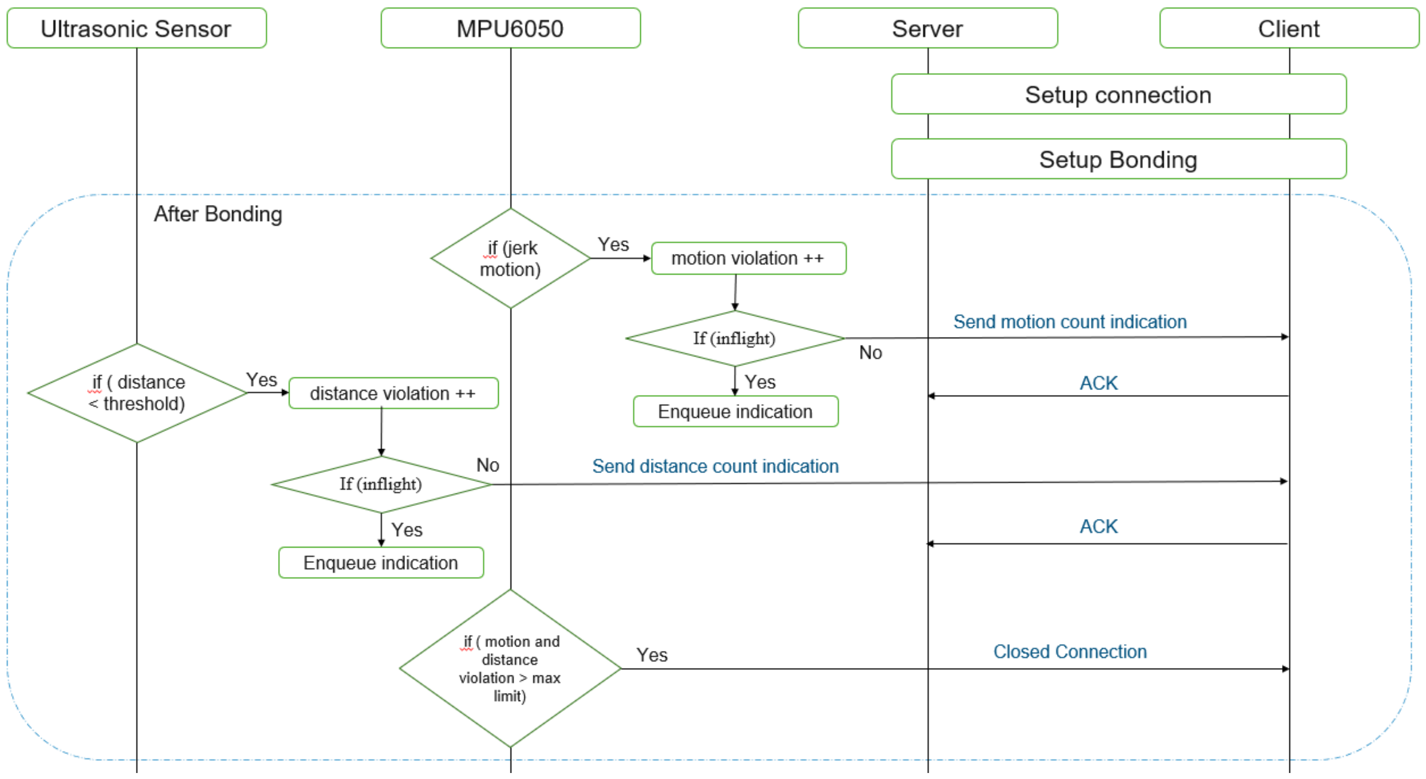
Additionally, the requirements mentioned below are slightly different than the high-level requirements mentioned in the course proposal as there have been design changes since then. However, the main functionality has not been changed.

Sr. No.	Requirements	Completed
1	Ultrasonic Sensor on the server will be able to detect if an object is in proximity up to a range of 3cm to 4m	Yes
2	Ultrasonic Sensor will be able to detect false triggering using hysteresis	Yes
3	IMU sensor will get accelerometer values and use it to detect sudden motion	Yes
4	Communication will take place over an encrypted link using bonding	Yes
5	Indication will be sent periodically after bonding	Yes
6	LED and LCD functionality implemented	Yes
7	Circular buffer to store indications if indication is in flight	Yes
8.1	Low Power requirement: Using energy mode to sleep in the lowest possible energy mode	Yes
8.2	Low Power requirement: Disconnect and reconnect devices after bonding to save more energy	No

Updates

Section Author: Peter Braganza

After we updated the data types and made other changes in the project the software data flow was modified to accommodate the changes made to the system. Below shows the final data flow of the system



Test Plan

Section Author: Swapnil Ghonge

We completed 92% of the tests in the course of 4 weeks. Following table shows the tests, their statues and description

Test Plan Table

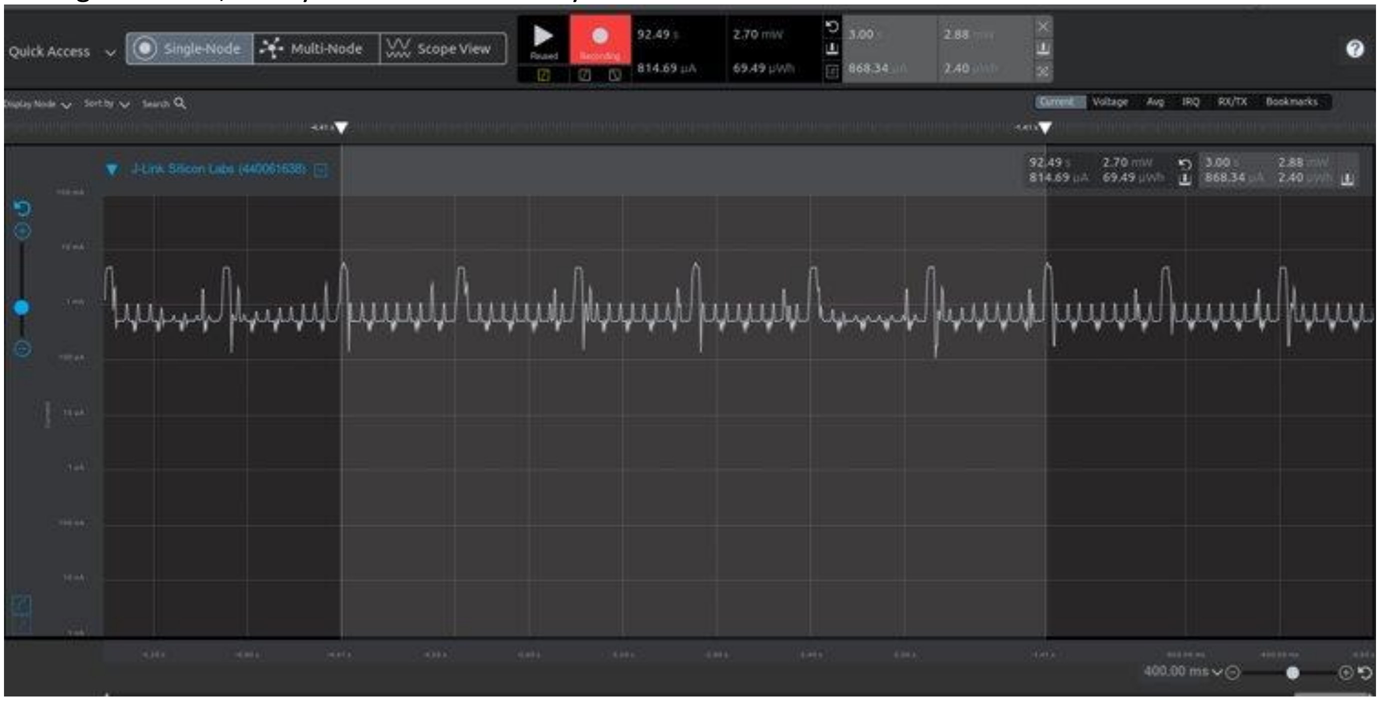
Test Number	Test Description	Date last run on	Test Result (Pass/Fail)	Notes
1	Test to check IMU accelerometer sensor measurements are returning valid readings on serial monitor	11-15-2021	PASS	Completed on Nov 16
2	Test to check ultrasonic sensor measurements are returning valid readings on serial monitor	11-15-2021	PASS	Used a scale and serial terminal to verify correct readings
3	Test to check buffer data in MPU6050 is read correctly	11-20-2021	PASS	Used Self-test registers to verify sensor data. Also used serial monitor and verified against expected values
4	Test to check if connection and bonding occurs and notifications are enabled followed by continuous transmission of accelerometer characteristic and distance characteristic values.	11-20-2021	PASS	
5	Ultrasonic sensor test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021	PASS	Used serial monitor to print serial buffer values
6	Test to check is hysteresis is implemented correctly to avoid false triggering of Ultrasonic sensor	11-26-2021	PASS	
7	Accelerometer meter test: A test to store the value in circular buffer and send the value to client when the connection is established and bonded.	11-26-2021	PASS	Used serial monitor to print serial buffer values
8	LED and LCD Display test: A test to display accurate alert/warnings on LCD i.e. corresponding to distance violations and sudden motion violations.	11-29-2021	PASS	
9	Full functional test: At full functionality level that subsystems should be able to perform acceleration, distance measurement and should send out appropriate alerts to the user.	12-03-2021	PASS	
10	Low power energy management tests - To test power consumptions using the energy profiler.	12-05-2021	PASS	

11	Test to check best values for connection parameters after low power requirements and circular buffer have been implemented	12-07-2021	PASS	This will also include test for max disconnect time This is also part of low power energy management.
12	Test to connect and disconnect after bonding has taken place	12-07-2021	FAIL	

LOW ENERGY PROFILING

Section Author: Swapnil Ghonge

During the sprint of the 4 weeks, we reduced the energy used by our system from 3mA to 800uA. To measure the current and low power consumption we used the Silicon Labs Energy Profiler. The system was sleeping in Energy Mode 2 (EM2) for most of the run-time. The system wakes up when there is an I2C transfer taking place. During 3seconds, our system consumed only 814uA and 2.88mW.



Screenshots of Project Demonstration

Section Author: Swapnil Ghonge

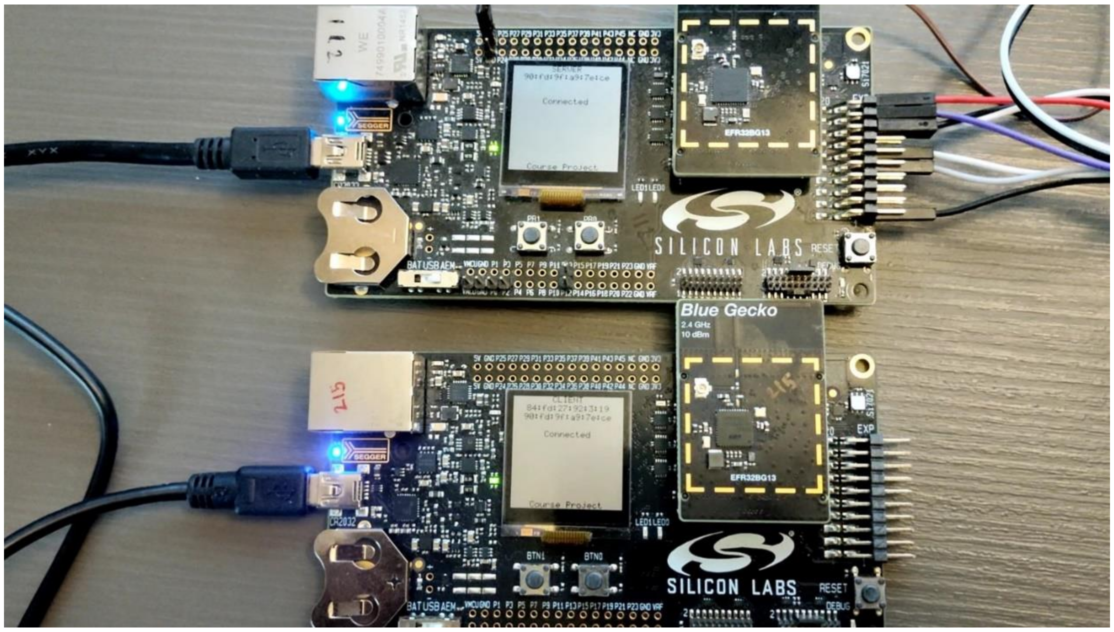


Fig:1 Gecko Boards Connected not Bonded

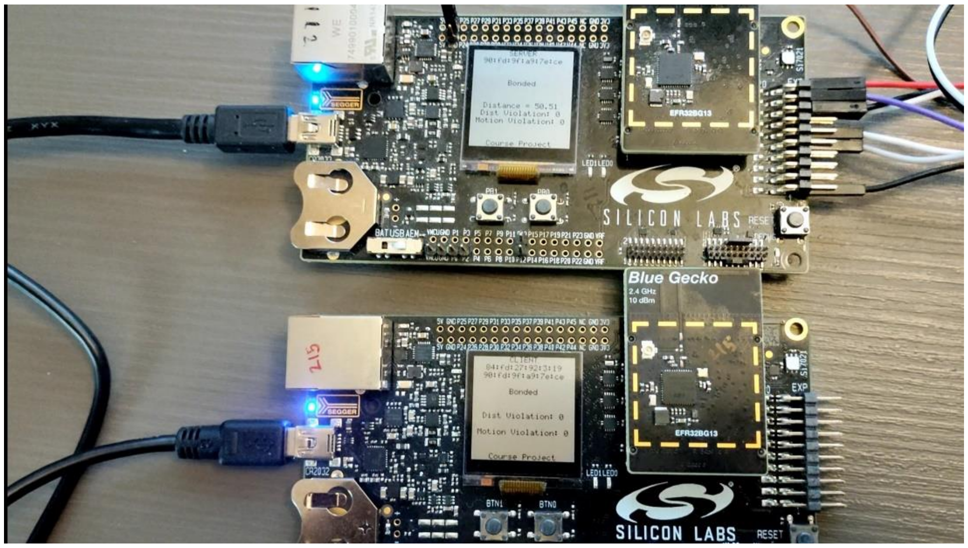


Fig:2 Connected and Bonded

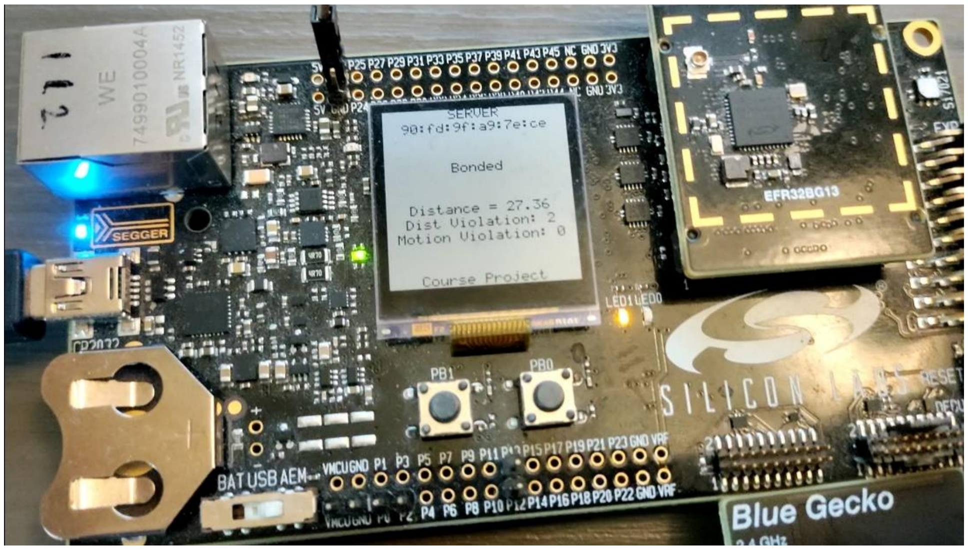


Fig: 3 LED1 illuminates when distance violation occurs on Server

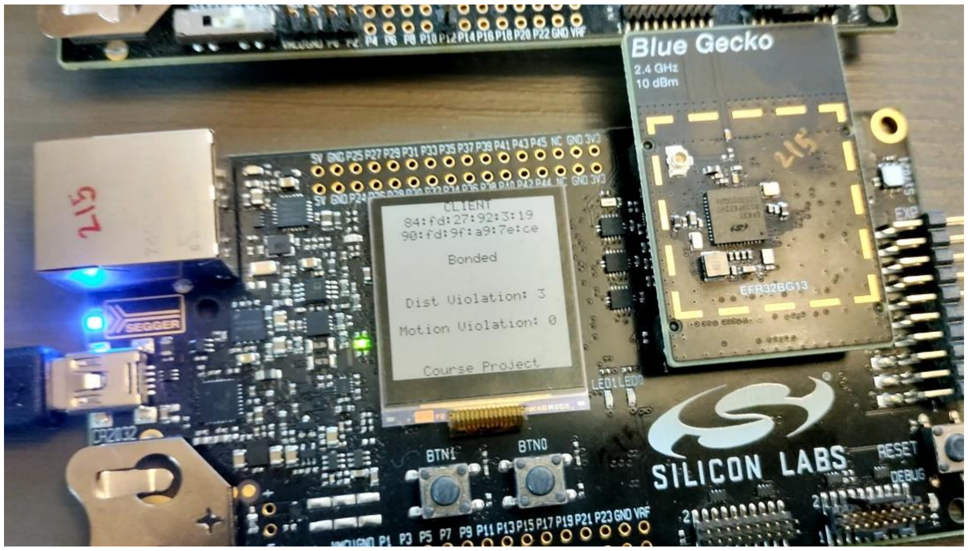


Fig: 4 Distance Violation data sent to Client over Bluetooth Low Energy Stack

Distribution of Work

Section Author: Peter Braganza

To have nearly equal distribution of work we divided the task and testing equally at the time of the project proposal. Even though we divided the task we found it useful to discuss each other's tasks and help each other out in their respective task. This helped both team members stay up to date with the project as a whole and not just their individually assigned task. Additionally, this provided more learning to each member and a better understanding of the working of each sub-system. This came in use when we had to integrate the whole system together. For these reasons we feel each member did 50% of the work.

The table below shows the ownership of each file.

File	Server/Client code	Team member responsible
main.c/.h	Server code	Peter Braganza
	Client code	Swapnil Ghonge
gpio.c/.h	Server code	Swapnil Ghonge
	Client code	Swapnil Ghonge
scheduler.c/.h	Server code	Peter Braganza and Swapnil Ghonge
	Client code	Peter Braganza
timer.c/.h	Server code	Swapnil Ghonge
i2c.c/.h	Server code	Peter Braganza
ble.c/.h	Server code	Peter Braganza
	Client code	Peter Braganza
oscillators.c/.h	Server code	Swapnil Ghonge
irq.c/.h	Server code	Swapnil Ghonge
	Client code	Peter Braganza

What Was Learned

Section Author: Swapnil Ghonge
There were many learning outcomes from the project. **One** key learning that I learned was assigning tasks appropriately that really helped in the smooth execution of the project. I learned how to coordinate in a team with proper communication, with assigned tasks in stringent deadlines. The **Second** takeaway from this project was how to configure ultrasonic-sensor using an event-driven state machine. I learnt how to integrate the learning from assignments 1 to 9 in the project of building robust state machines, isolation of functionality, sending data over Bluetooth low energy stack using API's. I also learned to make custom GATT Services to send data from the server to client. The **third** takeaway from this project was to how to debug a program step by step to know the root cause of the problem. I spend more time in designing a UML diagram before coding for interfacing the ultrasonic sensor. This reduced my time for debugging to a greater extent.

Section Author: Peter Braganza
Project Learnings:
I learnt how configure a sensor by reading the data sheet and understanding which values have to written to the sensor in order to configure it for our project usage. Learnt how to use an on-board FIFO buffer to read values from it and also learn about burst I2C commands. Burst I2C commands were useful as it allowed us to read values from 6 registers at one instance which was useful to get accelerometer values. These values needed to be read in sync to get correct readings. In the project proposal we proposed to use the on-board FIFO buffer of the MPU605 however, we learnt that the sensor write values of the accelerometer to the FIFO buffer at a 1kHz frequency which was too high for out purpose and hence switched to using burst I2C read commands instead. Hence, a valuable lesson here is to read the datasheet of the sensor which you intend on using well. Or at the very least read the parts which are extremely relevant in the datasheet very well before finalizing a sensor. Lastly, while I was not able to get the system to send indications while disconnecting and reconnecting after bonding I, was able to disconnect and reconnect the client and server. And depending on the interval at which that occurs I was able to see considerable power saving, more than what was mentioned in the Low power requirements section above.

Things I would do differently:

1. Read each sensor datasheet before finalizing a purchase order
2. Keep much more time for testing than developing code
3. During the planning stage itself identify alternative method to achieve the same requirement. This is useful when the method proposed fails similar to what happened to using the FIFO buffer and we had to switch to an alternate method

GitHub repository URL(s) = https://github.com/CU-ECEN-5823/ecen5823-courseproject-PeterBraganza