ECEN 5823

IOT Embedded Firmware

# SYMBIOTIC HEALTH MONITORING

Individual Report

Preshit Harlikar
May 2, 2018

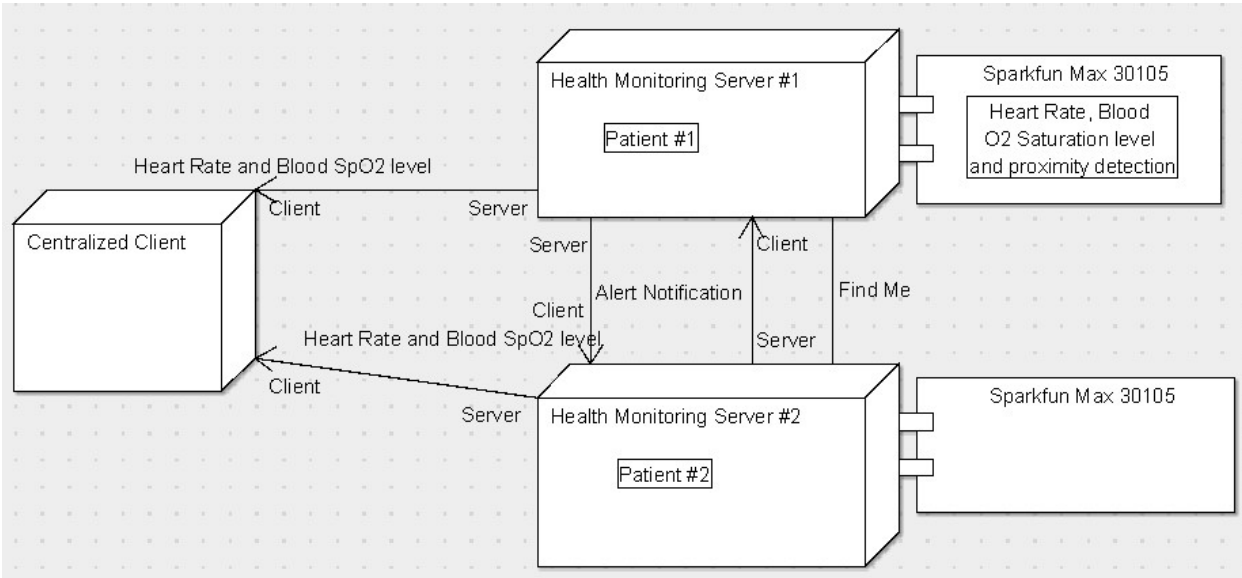
## What Problem the Project Addresses

Heart ailments remain the No. 1 global cause of death with 17.3 million deaths each year, according to "Heart Disease and Stroke Statistics". Every year about 735,000 people have a heart attack. Of these, 525,000 are a first heart attack and 210,000 happen in people who have already had a heart attack. Apart from heart attacks, Hypoxemia is a widespread problem. Hypoxemia is a below-than-normal level of oxygen in your blood, specifically in the arteries and may result in various symptoms such as shortness of breath. In this age of technology, for problems as severe as these, there needs to be a low power device capable of efficiently monitoring a person's heart rate and blood oxygen. If the results from the monitoring are abnormal, then the device should be capable of informing the person's partner and/or a higher authority.

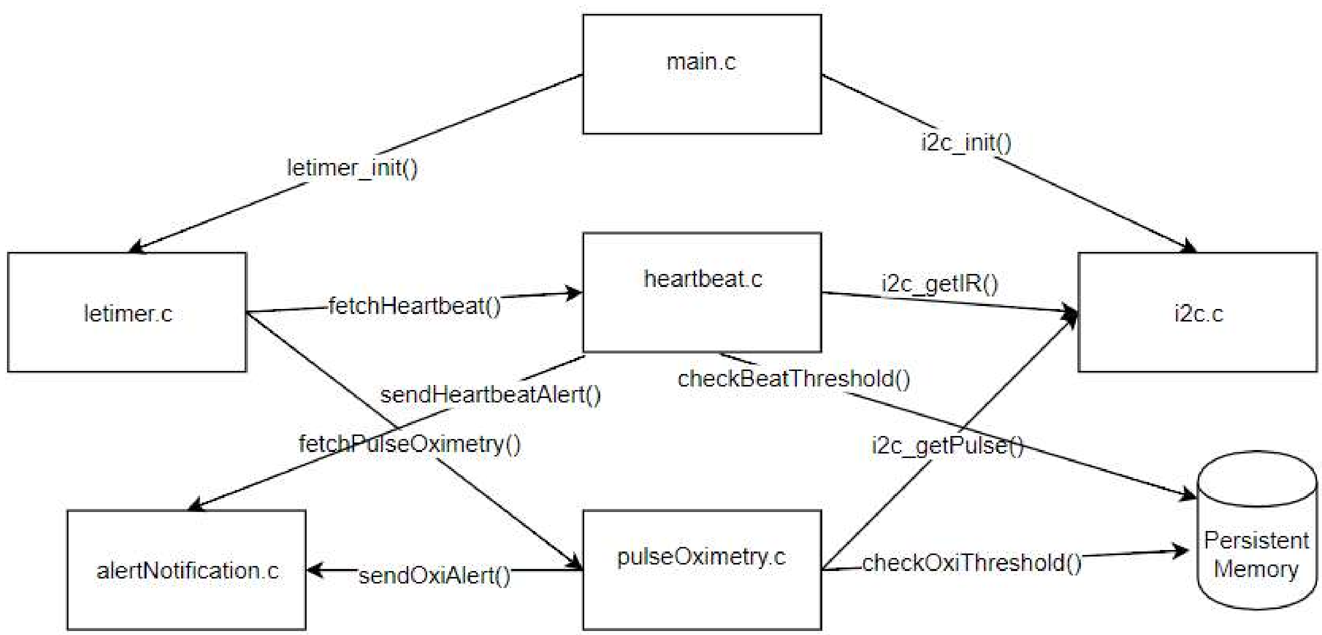
## How this project alleviates or solves the problem

In this project, we consider a scenario of a couple in which both husband and wife have heart ailments. So, there would be a device for both that would monitor their heart rate and the blood oxygen saturation level after specified time intervals. If any abnormal activity is sensed on one device and the second device is in the proximity of the first, it will receive an alert message. There would also be a centralized client device, to which the heart rate and pulse oximeter data would be sent for such a scenario. Since the monitoring is done in specific time intervals and the alerts are sent only in case of abnormal activities, it will consume less power leading to efficient monitoring.

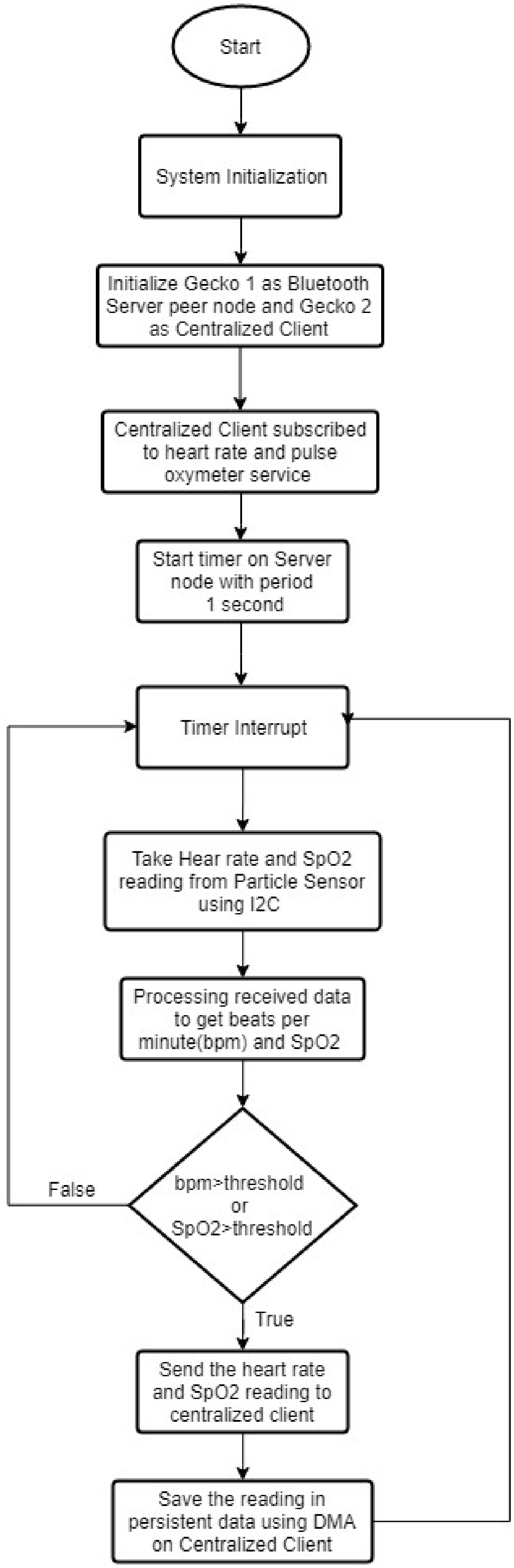
## Functional Block Diagram




## Software block diagram



## Software organization or flow chart/diagram



## Development Schedule

| To be completed by | Task | Team Member | Completion (as on 4/15) |
|---|---|---|---|
| 04/02/2018 | Interface code to SPI LCD Display | Ashish | LCD successfully interfaced using Si Labs' graphics library |
| 04/02/2018 | Interface code for heart rate values via I2C. | Preshit | Heart rate reading received and processed from sensor |
| 04/02/2018 | Interface code for pulse oximeter values via I2C. | Preshit | SpO2 reading received and processed from the sensor |
| 04/02/2018 | Implementing use of persistent data | Ashish | The threshold values for heart rate and pulse oximetry are successfully written to persistent memory and fetched on startup |
| 04/02/2018 | Test for sensor data by displaying values on LCD. | Ashish | The values for heart rate and pulse oximetry are observed to be stable |
| 04/15/2018 | Setting up the four services on the servers | Ashish | The server-side code required for each service is set up successfully |
| 04/15/2018 | Testing of services using Silicon Labs' mobile application as a client | Preshit | The basic characteristic values for all the services are received successfully |
| 04/15/2018 | Write code for subscribing for heart rate and pulse oximeter service on centralized client | Preshit | The client side code is set up for the centralized client to subscribe to these 2 services |
| 04/15/2018 | Write code for subscribing for Alert Notification and Find Me service on each of the peer nodes to work as client. | Ashish | The code is set up for the nodes which will be acting as the clients to each other for these 2 particular services |
| 04/25/2018 | Load power management during sensing and data transmission | Preshit | Load Power Management implemented on both the centralized client and individual peer nodes (details in verification plan) |
| 04/25/2018 | Integrating DMA data transfers in application code | Ashish | DMA transfer offloaded successfully and correct values are observed on reading the memory locations |

## Summary of each individual project

Broadly, the project can be divided into two parts:

- Centralized Health Monitoring, and
- Alert notifications between peer monitoring devices

The Centralized client subscribes to the Heart Rate and Pulse Oximeter services from each of the Health Monitoring Servers via BLE. Whenever the values for either of them cross the thresholds specified for the patients, the corresponding data is reported to the centralized client. In such a scenario, the second part of the project is also activated, where it would attempt to connect to the other node via BLE and send an Alert Notification. Additionally, any of the peer nodes can use the Find Me service to activate a buzzer alarm on the other device. In the current setup, there are 2 use cases perceived for this service: to locate the other device, or to explicitly signal the partner if the person is not feeling well and an Alert Notification is not yet generated due to some sensing error.

The two parts thus help in solving the problem in a mutually exclusive fashion. The Alert Notification / Find Me services between the two peer nodes help in immediate detection of the problem, so that the partner can take the necessary first-hand corrective measures. Reporting of the abnormalities observed during the day (or week) to a centralized client device can help the physician or any other family member to keep track of the patient's wellbeing and periodically alter their medications if required, without explicit intervention by the patients.

## List of sensors for this project

- **SparkFun Particle Sensor - MAX30105**
  Features:
  1. Built-in Red, IR, and Green LEDs
  2. 5V operation (3.3V is allowed, but green LED is not guaranteed to operate)
  3. Onboard 1.8V regulation and I2C interface circuitry
  4. Sensitivity configurable down to 7.81pA
  5. 3200Hz maximum sample rate
  6. Built-in 32 sample FIFO (First In, First Out)
  7. Applications include presence detection, distance sensing (18" max), pulse oximetry, blood oxygen saturation level (SpO2), smoke and particle detection.

## What BLE exposed services and client profiles will be implemented

For my part of this project, I will implement a centralized client device to which the server would send the heart rate and pulse oximeter data when an abnormal activity is sensed by the sensor. So, the Centralized client should be implemented that subscribes to two services: Heart rate service and Pulse oximeter service. This centralized client will be used to keep a track of the person's abnormalities, and can be used by a physician to alter his/her medications if required, without explicit intervention by the patients.

## Verification Plan

| Module | Sr. No | To be verified | Definition of Passing | Date test performed | Tested by | Measured result | Passed? |
|---|---|---|---|---|---|---|---|
| **I2C** | 1 | Check for Bus busy or any pending commands before performing i2c read or i2c write | In Debug mode, check the I2C Register values before sending the first Start command. Test passed if the registers have desired values | 4/2/2018 | Preshit | The registers have the desired values and no commands are pending | Passed |
| | 2 | Perform i2c single byte read operation from sensor | Reading a value from a specific register of the sensor, and check if the value is valid | 4/2/2018 | Ashish | The value from FIFOREADPTR register is read. | Passed |
| | 3 | Perform single byte write operation to the sensor | Writing a value to a register, and checking if the value is written by performing single byte read on the same register | 4/2/2018 | Preshit | Value is written to FIFOCONFIG register and read back from same register | Passed |
| | 4 | Check if the pulse is detected by the sensor | If the pulse is detected samples would be available in FIFODATA register of the sensor | 4/2/2018 | Ashish | New samples were available in FIFODATA register. Checked FIFOREADPTR and FIFOWRITEPTR registers and (readpointer!=writepointer) | Passed |
| | 5 | Perform i2c burst read on FIFODATA register of the sensor to get the value of Red and IR LED | Desired number of bytes are read from the sensor. | 4/13/2018 | Ashish | I2C burst read performed and received the desired bytes of data. (A NACK was being sent during i2c write operation that had caused a pending NACK problem for burst read operation which was later resolved) | Passed |
| **Particle Sensor** | 6 | Get a stable Red and IR led reading from the sensor | After 60 seconds of burst read from the sensor, the readings must be stable. | 4/13/2018 | Preshit | Using letimer, a 60 second stabilizing time is used, after which the sensor readings obtained are stable. | Passed |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 7 | Valid Heart rate reading from sensor after processing | Heart rate reading after processing must be within range of 20-255 | 4/13/2018 | Preshit | The values are within the range of 40-145 | Passed |
| | 8 | Valid Pulse oximeter reading from sensor after processing | SpO2 value for a normal healthy person should be in range of 90% to 99%.<br>For patients with mild respiratory diseases, the SpO2 should be 80% or above | 4/13/2018 | Preshit | The values tested were within the specified range | Passed |
| **BLE Server** | 9 | Create a Heart rate service on the server and check if the service can be subscribed through blue gecko app and the readings are correct | The blue gecko app should be able to see the heart rate service | 4/13/2018 | Ashish | The heart rate service can be subscribed on the app and the readings are accurate. | Passed |
| | 10 | Check if the custom pulse oximeter service can be subscribed through blue gecko app and the readings are correct | The blue gecko app should be able to see the heart rate service | 4/13/2018 | Preshit | The custom pulse oximeter service can be subscribed on the app and the readings are accurate. | Passed |
| **BLE Client** | 11 | Check if the connection is opened between client and server | If the connection is opened, *gecko_evt_le_connection_opened_id* is triggered on the client | 4/8/2018 | Ashish | Opened Connection is displayed on LCD of the Server Gecko | Passed |
| | 12 | Check if the Services advertised by the server can be discovered by the client | For every service detected *gecko_evt_gatt_service_id* event is triggered on the client | 4/8/2018 | Ashish | The services advertised by the server are displayed on the LCD of the client | Passed |
| | 13 | Check if the characteristics | For every characteristic | 4/9/2018 | Ashish | The characteristics of the service are | Passed |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | of a service can be discovered by the client | discovered *gecko_evt_gatt_characteristic_id* event is triggered on the client | | | displayed on the LCD of the client | |
| | 14 | Check if a valid characteristic value is received by the client. | For every abnormal value of heartrate or spo2 sent by the server, a notification is received through *gecko_evt_gatt_characteristic_value_id* event | 4/9/2018 | Ashish | A valid characteristic value is displayed on the LCD of the client. | Passed |
| | 15 | Check if the client can subscribe to two services simultaneously. | The client should subscribe to two services i.e. heartrate and pulse oximeter service for same connection. | 4/17/2018 | Ashish | The client subscribes to two services and valid readings are received | Passed |
| **Dual-Mode Client/ Server** | 16 | Functional findme service on the dual-mode client/server peer nodes | The peer node can subscribe to findme service from other peer node and the centralized client is not triggered for this service | 4/27/2018 | Preshit | A peer node subscribes to the findme service from another node and the client does not. | Passed |
| | 17 | Functional alert message service on dual-mode client/server peer nodes | When either heartrate or spo2 value goes above the threshold, an alert message is sent on other peer node. | 4/27/2018 | Preshit | Peer node subscribes to alert message service from other peer node for abnormal values of heartrate and spo2 | Passed |
| | 18 | Power off Sensor | Power off sensor by Pressing the button PB1 on either of the peer nodes | 5/1/2018 | Preshit | The sensor is put in shutdown mode when the button PB1 is pressed and turned back on if the button is pressed for the second time | Passed |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 19 | Working as a server only if the readings are above threshold | The peer nodes should function as a server only if its heartrate and spo2 readings go above threshold | 4/27/2018 | Preshit | The peer node operating as a client switches to server mode only if the readings are above threshold | Passed |
| **MITM** | 20 | Random and identical passkeys are generated on both the client and server | The passkeys generated on both the client and server are identical and random | 4/30/2018 | Ashish | The generated passkeys are displayed on the LCD and can be observed that they are random and identical | Passed |
| | 21 | Only secured connection is established | The client subscribes to the services from the server only when a secured connection is established | 4/30/2018 | Ashish | When the passkeys are displayed on the LCD, a button press request is offered to the user. If the user presses the button, only then is the connection established | Passed |
| | 22 | Does not ask for the passkey if a secured connection is already established | The client does not request the passkey if the bonding is established with the server. | 4/30/2018 | Ashish | If the server has already established a secured connection with the client, the passkey request is not done and the client subscribes to the services offered by the server | Passed |
| **DMA** | 23 | The data is logged to the desired location | The data characteristic value received by the client is logged by the DMA at given location | 4/28/2018 | Ashish | The data is logged by the DMA at given location(0x2000000) and is displayed on the LCD | Passed |
| | 24 | The data logged by the is the abnormal heartrate and spo2 values. | The characteristic value received by the client and the data logged by DMA should be the same | 4/28/2018 | Ashish | The data logged by the DMA are accurate i.e. the abnormal values received are being logged with any error | Passed |
| **Persistent Memory** | 25 | The threshold values for heart rate and spo2 are stored in the | The heart rate and spo2 threshold values can be loaded from the | 5/1/2018 | Preshit | The values are loaded from the persistent memory and displayed on the LCD. | Passed |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | persistent memory | persistent memory | | | | |
| | 26 | If no values are loaded in the memory, the threshold values must be stored in the persistent memory | Check the persistent memory for any stored values. If no values stored, then save the threshold values on persistent memory | 5/1/2018 | Preshit | New values are stored in the persistent memory and loaded back and displayed on the LCD | Passed |
| **Load Power Management** | 27 | Check if the Server always runs at lowest possible energy mode | Observe the current and average current values using Energy profiler | 5/1/2018 | Preshit | Load power management was done on the sensor by giving it a off-time of 200 ms for a period of 2 seconds. The average current was significantly reduced from 10 mA to 4.5 mA. | Passed |
| | 28 | Power off Sensor | Power off sensor by Pressing the button PB1 on either of the peer nodes | 5/1/2018 | Preshit | The sensor is put in shutdown mode when the button PB1 is pressed and turned back on if the button is pressed for the second time. The average current in power off mode goes down to 2 mA | Passed |
| | 29 | Check if the client always runs at the lowest possible energy mode | Observe the current and average current values using Energy profiler | 5/1/2018 | Ashish | Load power management was done on the client by decreasing its scanning frequency using *gecko_cmd_le_gap_set_scan_parameters().* The average current significantly dropped down to 6 mA from 10 mA | Passed |

## Energy Optimization in the Project

The project has one centralized client and 2 peer nodes which operate in dual mode as either a client or a server at a instant. All 3 nodes are scan for advertising devices, and thus it was very important to implement energy optimization techniques in all of them, since scanning does indeed involve high energy usage.

Scanning was optimized by setting the scanning parameters individually for the client and the peer nodes. The peer nodes, being the resource-constrained ones, scanned after an interval of 100ms with a scan window of 10ms, thus achieving a very low duty cycle of 10%. On the centralized client however, scanning was done after an interval of 100ms with a scan window of 50ms. This ensured that the centralized client did not miss any events from the servers, and moreover, since the client is a central device which would ideally be plugged in to power for the overall monitoring of both the patients, so energy optimization was not so critical for it.

Next, to implement energy optimization for the particle sensor using Load Power Management, several attempts were made to keep the sensor in a low power mode after pre-specified time intervals. However, the reliability of the readings obtained was getting severely affected by this since heart rate is such a feature which must be monitored continuously for accuracy, whenever it is.

So, energy optimization was achieved for the sensor by shutting down power to it through a GPIO pin not after pre-specified time intervals, but through a push button whenever desired by the patient. This helps in reducing the overall power consumption of the peer node, and also helps in accurate readings since whenever the readings are being fetched, they are fetched continuously without any power interruption. Further, for safety purposes, if in case the patient has not yet turned on the heart rate and pulse oximetry monitoring and he/she suddenly starts feeling uncomfortable, the system has the provision of a Find Me service to immediately send an alert to the other peer node, irrespective of the measured heart rate and pulse oximetry values.

## Implementation of Man-In-The-Middle Security

The project implements Man-In-The-Middle security between the centralized client and the peer nodes, and thus protects from any corruption of data (Heart Rate and Pulse Oximetry values) being sent to the centralized client.

- To implement MITM security, identical configurations for the security requirements are initially done on both the centralized client and the peer nodes on boot-up.
- Both types of devices are set to accept new bondings.
- Next, whenever the values for heart rate or pulse oximetry cross the specified thresholds, the peer node advertises its services to the centralized client and attempts to connect to it
- Once a connection is opened, the peer node sends out a command to enhance the security of the connection
- This triggers an event on both the devices to display an identical but randomly generated passkey on the LCD
- Both the nodes need to accept this passkey using a push button, for the connection to be authenticated
- Once authenticated, the centralized client starts the service and characteristic discovery on the server node
- For subsequent connections between the same 2 set of devices, this procedure is skipped through since the connection between them is already encrypted. This is done by checking for the 'bonding' parameter, whenever the connection is opened between the 2 devices. Any value other than 0xFF for this parameter indicates a previous bonding handle.

## Lessons learnt throughout the Project Implementation

- **Dual Mode Master/Slave:** Implementation of a peer node as a dual mode master/slave has been one of the biggest learning experiences while developing this project. Implementing a centralized client which subscribes to services and a server which provides characteristic values for such services was relatively straightforward. However, synchronizing the two modes in a device at once involved an in-depth understanding of the all possible states that a device can go through. A state-machine like structure was implemented to keep track of the current state and the current mode of operation of the device. Since there are some events which can be triggered for both the master and slave, it was pivotal that the state of the device be managed through such a state machine to accurately handle the incoming events. Implementing such a state machine also helps in increasing the extensibility of the project for any new kinds of services which might be required in the system.
- **Interfacing the Particle Sensor for Heart Rate / Pulse Oximetry:** Interfacing the MAX30105 for life-critical and high frequency services such as the Heart Rate and Pulse Oximetry was indeed a challenge. Even though a comparatively small volume of data was being sent by the sensor, yet managing the timing nuances for these services was the most important aspect. This involved adding a stabilization time after startup, fetching the readings after specified intervals and then continuously maintaining a running average. Detailed analysis and calculations from the sensor's datasheet were required for this to interpret the digital packed values sent by the sensor.
- **I2C Implementation:** Working on the project actually gave novel experiences regarding the earlier known I2C protocol. The first one was regarding pending commands. It was observed that whenever any unnecessary command is transmitted to the slave via I2C, it is sent at the next available opportunity. This has the potential of corrupting the next transactions with the slave. The corruption of data for the further transactions was what led us to the second learning: of carefully analyzing the vendor's sequence for I2C. Any command the master sends, which is not explicitly required as per the vendor specifications, does not get ignored but gets sent at the next chance, possibly damaging that transaction. The analysis for all this bit-level sequence was done using a Logic Port Analyzer, thus enhancing the hands-on exoerience with that equipment.
- **DMA:** Working on the add-on aspect of this project, Direct Memory Access, helped realize the importance of offloading the memory transfers to another piece of hardware. The project implemented DMA on the centralized client to move the data (abnormal heart rate and pulse oximetry values) to a specified memory location. The idea is to help a family doctor or any other family member to note the abnormalities for the patient over the day or week. Offloading the data transfer work helped achieve a consistency in the scanning intervals for the client to not miss any real-time requirements of the system.
- **Managing Limited Resources:** Management in a device with constrained resources was another important learning outcome through this project. The project did not implement any UART logging and simply used the on-board LCD itself for observing all kinds of data, connection details and alert messages. Since the system is designed to be a wireless and low power one, using another interface for logging purposes was not practical as per a product perspective. The system also made the best use of the 2 onboard buttons and multiplexed 4 functionalities through the 2. The state machine for the system, which was described earlier, was used to keep track of the mode to help determine which of the functionality is served by a button at a particular instant of time.

## Final Status of the Individual Project:

This particular part of the project has been associated with the I2C interfacing for the sensor, and setting up the centralized client for subscribing to the heart rate and pulse oximetry. The sensor has been successfully interfaced to provide consistent values for heart rate and pulse oximetry without any breakdowns. The centralized client is also able to receive data from the servers whenever the values cross the thresholds and a substantial scanning interval is maintained on the client side to ensure that any packets are not missed. The add-on feature of DMA is also successfully implemented in the centralized client to move the incoming data values to memory through an offloading hardware. Overall, the project has reached completion with regards to the planned features.