Physiological Sensors and Hardware Interfacing

Capstone Project: CSCI 6838.04

May 5, 2017

Mentors:

Dr.Ioannis Thomas Pavlidis
Director, UH Computational Physiology Lab
Email: ipavlidi@central.uh.edu

Ashik Khatri
UH Computational Physiology Lab
Email: ashikrk@gmail.com

Instructor:

Dr. Pradeep Buddharaju
Assistant Professor in Computer Science
Email: Buddharaju@uhcl.edu

Gryffindor Team Members:

Sanif Maredia
Zoosabali Maknojia
Cheena Saini
Khushali Mehta

Acknowledgements

The Gryffindor team would like to acknowledge and give thanks to the individual who help us to make this project possible:

Dr.Ioannis Thomas Pavlidis

For providing introduction to various physiological sensors and its uses.

Mr. Ashik Khatri

For helping to reach out to companies for SDK, proper selection of SDK for project and providing sample data output for various sensors.

Mr. Muhsin Zahid Ugur

For providing guidance with iOS application and exporting data to Excel file

Dr. Pradeep Buddharaju

For explaining the working, use of the existing systems as well as for developing plugins and project management

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Background

The increasing use of on-board electronics and in-vehicle information systems has made the evaluation of driver task demand an area of increasing importance to both government and industry and understanding driver frustration has been listed by international research groups as one of the key areas for improving intelligent transportation systems. Protocols to measure driver work load have been developed using eye glance and on-road metrics, but these have been criticized as too costly and difficult to obtain. As an alternative, this study shows how physiological sensors can be used to obtain electronic signals that can be processed automatically by an on-board computer to give dynamic indications of a driver's internal state under natural driving conditions. Such metrics have been proposed and have been used in simulations, but have not been tested on stress levels approximating a normal daily commute using sensors that do not obstruct drivers' perception of the road. This experiment was designed to monitor drivers' physiologic reactions during real-world driving situations under normal conditions. Performing an experiment in real traffic situations ensures that the results will be more directly applicable to use in these situations; however, it imposes constraints on the kinds of sensors that can be used and the degree to which experimental conditions can be controlled.

This project presents a method for measuring stress using physiological signals. Physiological signals are a useful metric for providing feedback about a driver's state because they can be collected continuously and without interfering with the driver's task performance.

1.2 Physiological Sensors

1.2.1 Shimmer GSR+ 3.0

Shimmer is small wireless sensor platform suited for wearable applications. It enables motion capture, long-term data acquisition and real-time monitoring. Shimmer units are pre-programmed with LogAndStream firmware program to capture data over Bluetooth as well as onto the SD card of shimmer. It has a Shimmer Capture Application compatible with both windows and Linux system. A host-side (PC) application measures Galvanic Skin Response (GSR). Single shimmer device is connected via Bluetooth. Allows users to save and display

data and gives user a graphical view of the live data. Output data generated are in raw and calibrated form. Saves data in CSV format to perform subject analysis. Shimmer Unit is integrated with many other component (like ECG, EMG, etc.), which includes complex configurations. It also has other functionalities, which can be used in measuring distracted driving.

1.2.1.1 Shimmer Capture:

It is a C# based console application that integrates all functionality of Shimmer device that is provided by the Shimmer API. Shimmer API includes Bluetooth connectivity, data streaming process, LED display for finding Shimmer status, etc. that forms the core features that Shimmer is embedded with. This Shimmer API is used as an interface between Shimmer Device and Shimmer SDK. For the Shimmer application to build and execute, Shimmer SDK is importing all the packages from Shimmer API that will give this application a working feature of all the methods that a Shimmer is supposed to have.

Shimmer capture application extracts real time data the Shimmer generates time to time when Shimmer gets paired with the PC via Bluetooth. This data includes Galvanic Skin Response (GSR), and its respective timestamps. The application is integrated with S-Interface which has plugins for many other physiological sensors. With application of S-Interface, user can invoke multiple physiological specific applications for capturing real time data at once. Its importance is that it will reduce the no of clicks to start many applications at once.

1.2.2 Apple Watch Series 2

Apple watch in smart watch which incorporates fitness tracking as well as health application. It is connected to iPhone wirelessly. It is compatible with iPhone 5 and later versions. It is supported with iOS 8.2 and later. Apple provides Health application which works on iPhone. Health application is not available on iPad.

1.2.2.1 Health Application

The application shows health related data like heart rate, breathing rate, calorie count for exercise, etc. This data is captured through Apple Watch. Application uses Health Kit

framework. Apple development provides this API that extracts real time user's health data. Health Kit store is a database which consist of encrypted data of Health Kit.

The framework restrains types of data and units into predefined list. This will guarantee that all applications understand data and it uses. Moreover, this framework provides variety of classes. Each class can have their own importance and uses.

The data displayed on the iPhone consist of many information like heart rate, blood pressure, respiration rate, etc. To measure the stress level in distracted driving, heart rate is only taken into consideration.

2. Requirements

The requirements of this project include business requirements, technical requirements and software requirements. A business requirement suggests business perspective of the application. Technical requirement suggest different technical modules i.e. technicalities of the application and system requirements specify all the hardware and software tools that will be used to complete the project.

2.1 Business Requirements

The Shimmer capture should be able to read and extract the real-time GSR and meaningful time-stamp that the shimmer is generating every second. User should also save the real time data in .CSV format for processing the data

The heart rate monitoring iPhone application should read the heart rate from the Apple Watch. It should generate logs of all the entries of BPM with corresponding time-stamp in the order of most recent to oldest. This data should be stored in the firebase, which is provided by Google cloud infrastructure.

BR-1	Develop Shimmer capture application which reads real-time GSR and save it to external location
BR-2	Integrating Shimmer application with S- Interface
BR-3	Develop an iOS application that can read the heart rate data and save it to external location

Table 2-1: Business Requirements

2.2 Functional Requirements

Shimmer API provides tools for communicating with device for developing application. Shimmer API provides necessary classes and interfaces for developing the application. Application provides a form view for performing operations. Develop plug-in for application so it can be integrated with S-Interface.

BR-1	Develop Shimmer capture application that can read and extract real-time GSR and time-stamp every second.
FR-1.1	Application should detect the connected device
FR-1.2	Application should provide device state
FR-1.3	Application should read GSR data with timestamps
FR-1.4	Application should provide save functionality for data processing.

Table 2-2: Business Requirement 1 and associated functional requirement

BR-2	Integrating Shimmer application with S- Interface
FR-2.1	Develop plug-in for application
FR-2.2	Develop S-Interface GUI for performing operation
FR-2.3	Provides graphical view in S-Interface

Table 2-3: Business Requirement 2 and associated functional requirement

In case of Apple Watch, an iPhone application is built and developed using swift programming language. For this purpose the iPhone needs to be synchronized with the Apple Watch via Bluetooth. The application should be able to extract and populate heart rate in beats per minute (BPM) with respective date and time that the watch measures of particular person. As soon as the application pulls the data, save this data is done on the firebase cloud infrastructure. Also, save the data into Amazon Dynamo DB. The summarization of this requirement is as follows.

BR-3	Develop an iOS application that can read data from Apple Watch
FR-3.1	Develop swift-based iOS application that extracts heartrate from Apple Watch
FR-3.2	As soon as the data gets populated, it is saved in firebase cloud infrastructure.
FR-3.3	Save the data to Amazon Dynamo DB.

Table 2-4: Business Requirement 3 and associated functional requirement

2.3 System Requirements

System requirement is comprised of both hardware and software requirements. Software requirement for Shimmer Capture and Pan-tilt is Microsoft Visual Studio to write C# code. And for iOS heart rate monitoring, XCode Integrated Development Environment (IDE) to write swift code need to be installed on the Mac PC. MS office and Notepad++ is used for documentation and thorough code review respectively.

For iOS Application, Xcode is used for developing swift application

SR-1	Microsoft Visual Studio – to write, compile and execute C# code for apple watch.
SR-2	For Shimmer Plugin: Microsoft Visual Studio – to write, compile and execute C# code for apple watch.
SR-3	S-interface and its dependencies
SR-4	For iOS Application - Xcode
SR-5	Notepad ++ - code review.

SR-6	MS office – documentation and presentation.

Table 2-5: Software requirements

For hardware requirements, Shimmer Capture requires Shimmer GSR+ sensor, Windows PC, Heart Rate Monitoring requires Apple Watch, iPhone and Mac PC.

HR-1	Shimmer GSR+ sensor
HR-2	Windows PC
HR-3	Apple Watch
HR-4	iPhone 6 series or 7 series
HR-5	iOS Mac PC

Table 2-6: Hardware requirements

3. System Details

3.1 Initial System Design

3.1.1 Shimmer Capture

Original Shimmer application provides all the functionalities for reading GSR data, Galvanometer, Accelerometer and many more. Moreover, all measurement parameters are not necessary for measuring stress level. Therefore, our application provides only GSR data as per the project requirements in order to measure stress. Shimmer application has simple architecture diagram. It consists of Shimmer Sensor, Bluetooth, Application and Database.

Communication between shimmer sensor and application takes place through serial port. Once shimmer is paired to the device, other functionalities like connect, disconnect and streaming can be implemented.

Shimmer C# API consist of four important classes: Shimmer Bluetooth, Shimmer32Feet, Shimmer and Object Cluster. Shimmer and Shimmer32Feet are classes responsible for connection and disconnection of the sensor with the device. Shimmer32Feet is used for advanced Bluetooth functionality only. Object Cluster class is used to hold the data and then it is sent to

User Interface (UI). Communication between API and UI takes place through handler. Handler is sends message to UI if the shimmer state is changed.

The live Galvanic skin response (GSR) data with time frame is displayed on the interface. This data can be stored in Excel sheet once user stops streaming.

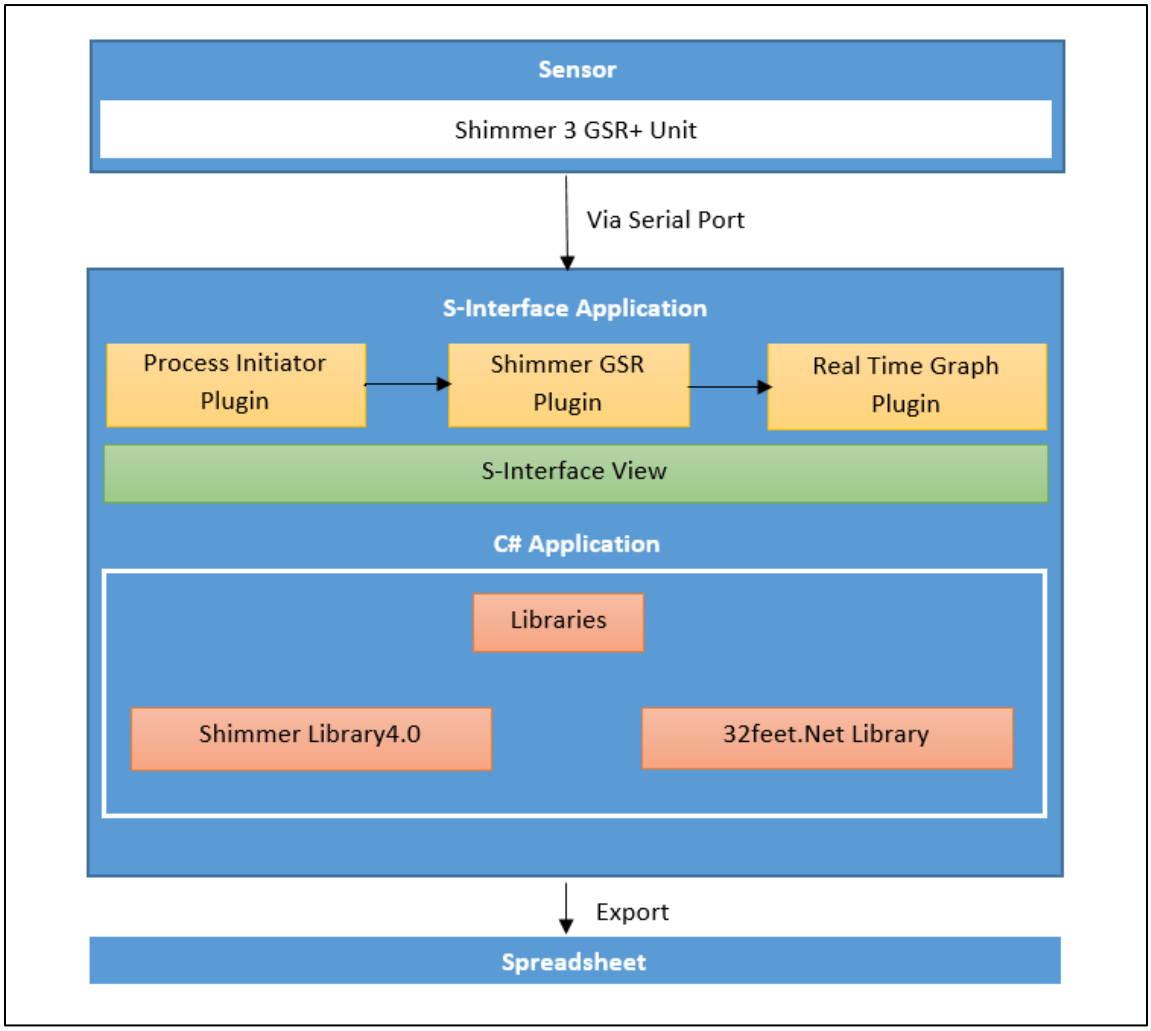

Figure 3-1: System Architecture Diagram-Shimmer

3.1.2 Heart Rate Monitor

Heart Rate Monitor is very similar to Apple's Health Application. But, Heart Rate Monitor is just used to record only heart rate of the user. Also, the recorded heart rate data is saved into Google's firebase.

Heart Rate Monitor application uses apple's health kit framework. Health Kit provides variety of quantity types, which helps to measure various health related data. Once the health store is initialized variety of health features can be accessed. As, all the measurement parameters are not required; particularly HKQuantityType class of HKSampleType is used to get heart rate. Sample queries are used to get sample data from Health Kit store. Sample query returns sample objects that match the provided type and predicate. Once the query is instantiated, it is executed using execute method. Once the query is completed, the results are dispatched to UI.

UI shows the recent heart rate with time and date. The recorded information is stored in Firebase.

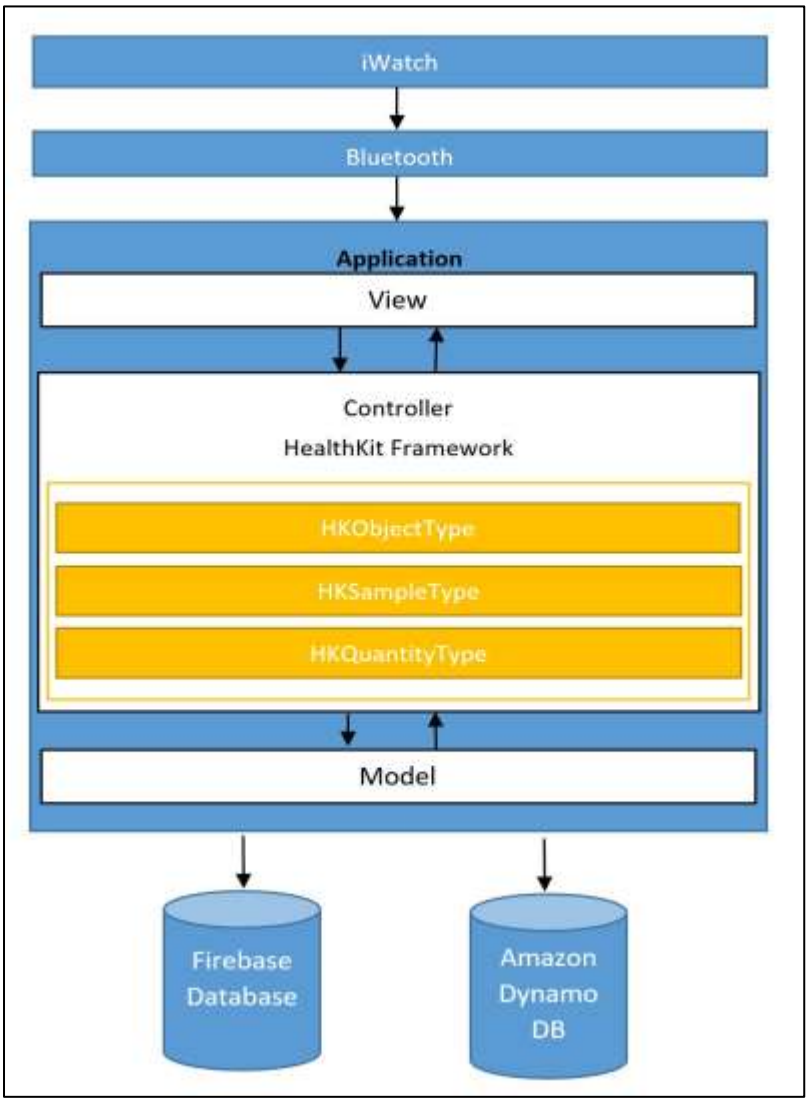

Figure 3-2: System Architecture Diagram- Heart Rate Monitor

3.2 Implementation

3.2.1 Shimmer Capture

Shimmer Capture is a developed in C#. This application allows the configuration of different parameters on Shimmer3 and is designed to demonstrate its functionality. It uses the shimmer SDK for development. It uses ShimmerClosedLibraryRev0-4 library for accessing data. Application uses Serial Port class for establishing the serial communication.

Connect module:

Application uses 32feet .NET Bluetooth library for establishing the Bluetooth connection between device and application. This library provides necessary classes for locating device through the ports

DataStream module:

Once the shimmer sensor gets connected, the streaming of shimmer data is done using the stream functionality of the same application. This streaming generates the live GSR Data and the timestamp. It uses serial communication for data transfer. Application uses Serial Port class for data transfer from device to the application. By implementing serial port class, we can use its methods for data retrieval.

Stop module:

The Stop module is used to stop the streaming process. Once the stop module is invoked user can go ahead with Save Functionalities.

Save module:

Application uses List class for saving newly generated GSR values. List grows dynamically, therefore it saves data continuously. This module uses SaveFileDialog class object for saving data to CSV. By using that object user can specify the filename and its location in dialog window.

Developing Plugins:

One of the major requirement of the project is to integrate the application it S-Interface. For integrating it with S-Interface, plug-ins are required. Shimmer Plugins have two main classes; one

which interacts with S-Interface (ShimmerPlugin) and other which is code-behind for S-Interface GUI. ShimmerPlugin class initializes the input and output pins.

S - Interface Implementation

The S-Interface application uses three plugins which are as follows:

Process Initiator Plug-in: Process initiator starts the process and it acts as an input pin to the Shimmer GSR plugin. Output pin of process initiator is connected with input pin of Shimmer Capture plug -in. It is the main entry point for the application. This plugin can initiate multiple plugins at the same time to measure various physiological parameters.

Shimmer GSR Plug-in: It is the intermediate Plug - in between the Process initiator and real-time graph plug in. The output pin of process initiator acts as an input to the Shimmer GSR and then the real-time data which is generated is sent to the output pin of Shimmer Plugin. This further acts as input for the graph plugin.

Real-Time Graph Plug-in: It plots the graph of the live GSR value generated with respect to the real time. This GSR data acts as an input pin to the Real-time Graph plugin which can be graphically viewed.

3.2.2 Heart Rate Monitor:

Data Retrieval Module:

Data from apple watch and iphone is stored in Health Kit framework. To implement and have access to all features of Health Kit framework, import Health Kit framework in the Xcode application. After importing, it is important to authorize to Healthstore.

After getting access to HealthStore, various types of object in Health Kit framework can be accessed. Data can be obtained using HKSampleQuery.

Recent heart rate with respective time is displayed on the user interface once play functionality is executed. User can stop recording data by invoking stop button once they are done.

Data Export Module:

1. **Firebase**

The heart rate with respective date and time recorded from watch is stored in Firebase provided by Google.

iOS application is linked with Firebase through application's bundle identifier. Once project is created in firebase, a file names 'Google-Info.plist' is generated. This file must be imported into iOS application in order to which will help in configuration of Firebase in the application. The application uses cocoa pods to add Firebase SDK into the application. A cocoa pod is a dependency manager. It manages and uses third party code. Pod file is generated once pod is initialized. Necessary pods for Firebase should be mentioned in the pod file before it is installed. After installation of pod, the application can access Firebase by importing necessary modules to save the data into Firebase [2].

Real time data generated is stored in the Firebase as JSON objects. Each time the record generated is stored with unique key. Data is stored in form of cloud based JSON tree.

2. Amazon Dynamo DB

Amazon Dynamo DB is fast, highly – reliable, NO SQL database. Alike to Firebase, AWS SDK for iOS is accessible through cocoa pods. The application uses cocoa pods in order to setting up the SDK.

It is necessary to set up roles and policies in AWS Identity and Access Management (IAM). Identity Pool is required for establishing an AWS identity with the application. Also, it is necessary to provide region and pool in the information property list of the application. This will configure AWS in the application. Table is created with primary key and required fields in AWS Dynamo DB Console.

Mapping of the values of the application with Dynamo DB table is done through high level library called Amazon Dynamo DB object mapper. This will help to populate the table created in Dynamo DB from the application.

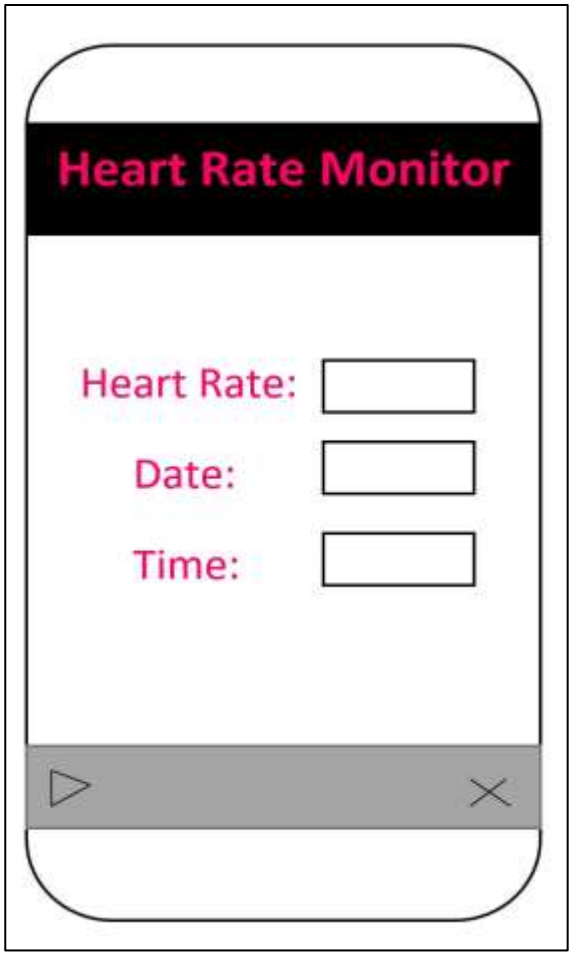

Figure 3-3: UI Diagram - Heart Rate Monitor

As shown in Figure 3-3, UI diagram represents play and stop button. When user clicks on play button, data – heart rate with respective date and time is captured from watch to phone. The application represents the most recent values of heart rate with date and time. As soon as, the apple watch notices change in data, the textbox in the UI is changed in seconds. The process of streaming continues and the values in the UI keeps changing.

Once, stop button is clicked the streaming is stopped and the value in the UI will not change further. Thus, the collection of measuring values is stopped. Once stopped, the Firebase and AWS Dynamo DB contains all the recorded values recorded between start and stop functionality.

3.2.3 Integration of Shimmer Plugin with S-Interface:

S-interface is software that uses algorithms to measure various physiological variables using thermal imaging for example perspiration rate and breathing rate. The modules that were

designed to integrate the ShimmerCapture Application and ShimmerPlugin are the three plugins i.e. Process Initiator Plugin, Shimmer GSR plugin and Real-Time Graph plugin as explained above in implementation.

Setting Configuration of Shimmer Plugin with S-Interface:

The initial configuration includes the adding of the source code of shimmer capture application into the plugin. Further various references are to be added to provide with the respective .dll files. The other important part of configuration is to add the assembly-info as two different namespaces are being used in this project. Lastly, it is necessary to set the output path to the plugins folder in S-Interface Additionally, to debug, the 'start external program' should have the path of the S-interface.exe.

Note: Due to privilege issues, visual studio needs to be started in an 'administrator' mode.

3.3 Testing and Verification

3.3.1 Shimmer Capture

Once application is developed, they must be tested. Application was tested for individual modules. It was installed in another desktop to see if it is working successfully. Shimmer uses Bluetooth for communication. Therefore, it will be operational within the Bluetooth range. Application was tested with lower version of Microsoft Excel so check whether the save functionality is operational. It was installed on different machine to check whether the saved configurations for S-Interface are captured and executed successfully.

3.3.2 Heart Rate Monitor

After implementing all necessary functionalities in the application, testing has to be performed. Once user clicks on play button, streaming of the data is started. User Interface will reflect with the most recent measured heart rate with respective time. Meanwhile, the data is stored into Firebase and Amazon Dynamo DB. After streaming, user can click on stop button and can stop streaming. Once the streaming is stopped, recording of the data is stopped. Moreover, the saving data to Firebase is stopped.

Checking the values on apple watch with the user interface performs verification of the application. The user interface displays the value of heart rate, which was measured before few seconds. Data in Firebase is saved in JSON format. The Firebase data is verified by comparing the JSON values

with the values shown on the watch. The data into Amazon Dynamo DB is in non-relational form. Those values are also compared with Firebase JSON value and Apple Watch value. After comparing the values, we come to conclusion that the both values were similar in both database.

3.3.3 Shimmer plugin with S-interface

To test the Shimmer plugin with S-interface, three plugins i.e. process initiator, Shimmer GSR and real time graph are added from the drop-down list of 'create configuration' in the Plugin Graph tab and suitable connections are made as described above. Next, the user interface tab shows all the UI i.e. process initiator, Shimmer GSR UI and Real-time graph UI. When connect, functionality is invoked in Shimmer-GSR UI, Shimmer device gets connected to the SDK of Shimmer unit. And accordingly, status of Shimmer connectivity as 'connected' and 'not connected' is displayed. When the play button is invoked in the process initiator, the real-time GSR data streaming will start and that can be viewed graphically on the graph. Once the stop button is clicked, the data streaming will be stopped.

Verification of the S-interface can be seen with the fluctuating graph generated with the real-time GSR values. On invoking the save functionality, all the data that is collected of a user during the streaming will be saved in a spreadsheet to an external location.

4. Project Management

The Gryffindor team was made of four team members. The main role of each team member was split as follow:

- Project manager: SanifMaredia
- Technical lead: Zoosabali Maknojia
- Presentation and Website lead: CheenaSaini
- Documentation lead: Khushali Mehta

All the team members contributed to the overall project development process. All the members fulfilled all the roles been assigned during the development process. Our project was implemented using agile methodology.

4.1 Project Timeline

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Resource Names
1		**Physiological Sensors and Hardware Interfacing**	**65 days**	**Wed 2/1/17**	**Tue 5/2/17**	
2		**Requirement Analysis**	**62 days**	**Fri 1/6/17**	**Tue 4/4/17**	**Zoosabali,Khushali,Cheena,Sanif**
3		Research and brainstorming	2 days	Mon 1/23/17	Tue 1/24/17	Cheena,Khushali,Sanif,Zoosabali
4		Building Project Website	1 day	Wed 1/25/17	Wed 1/25/17	Cheena,Khushali,Sanif,Zoosabali
5		Analysis of Business requirements, functional/Technical requirements and System requirements	3 days	Thu 1/26/17	Mon 1/30/17	Cheena,Khushali,Sanif,Zoosabali
6		Contacting for SDK's of Shimmer GSR, Zephyr Bioharess and Empatica	5 days	Tue 1/31/17	Mon 2/6/17	Cheena,Khushali,Sanif,Zoosabali
7		Contacting for drivers for pan tilt unit	4 days	Thu 3/30/17	Tue 4/4/17	Cheena,Khushali,Sanif,Zoosabali
8		**Design**	**5 days**	**Tue 1/31/17**	**Mon 2/6/17**	**Cheena,Khushali,Sanif,Zoosabali**
9		Analysis of Software/System architecture	2 days	Tue 1/31/17	Wed 2/1/17	Cheena,Khushali
10		UI Diagram	1 day	Thu 2/2/17	Thu 2/2/17	Sanif,Zoosabali
11		Task division	1 day	Fri 2/3/17	Fri 2/3/17	Cheena,Khushali,Sanif,Zoosabali
12		Intial Technical Report	2 days	Sat 2/4/17	Mon 2/6/17	Cheena,Khushali
13		**Implementation**	**59 days**	**Tue 2/7/17**	**Fri 4/28/17**	**Cheena,Khushali,Sanif,Zoosabali**
14		**Shimmer Application**	**59 days**	**Tue 2/7/17**	**Fri 4/28/17**	**Cheena,Khushali,Sanif,Zoosabali**
15		Connection	3 days	Tue 2/7/17	Thu 2/9/17	Cheena,Zoosabali,Khushali,Sanif

Figure 4.1: Project Timeline-1

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
13		Connection	3 days	Tue 2/7/17	Thu 2/9/17		Cheena,Zoosabali,Khushali,Sanif
14		Streaming	7 days	Fri 2/10/17	Mon 2/20/17		Khushali,Sanif
15		Status and data display	16 days	Tue 2/21/17	Tue 3/14/17		
16		Exporting data to file	1 day	Wed 3/22/17	Wed 3/22/17		
17		**Apple Watch IOS Application**	**10 days**	**Tue 3/7/17**	**Mon 3/20/17**		**Khushali,Sanif,Cheena,Zoosabali**
18		Exploring Healthkit Framework	1 day	Sat 3/11/17	Sat 3/11/17		Cheena,Khushali,Sanif,Zoosabali
19		Healthkit data display(heart rate)	5 days	Mon 3/13/17	Fri 3/17/17		Khushali,Sanif
20		Testing Applications with sensors	4 days	Fri 3/17/17	Wed 3/22/17		
21		**Mid Term Presentation**	**8 days**	**Sun 3/12/17**	**Tue 3/21/17**		**Cheena,Khushali,Sanif,Zoosabali**
22		Preparing presentation slides	7 days	Sun 3/12/17	Sun 3/19/17		Cheena,Khushali,Sanif,Zoosabali
23		Preparing Technical Report	7 days	Sun 3/12/17	Sun 3/19/17		Sanif,Zoosabali,Cheena,Khushali
24		**Integration and Testing**	**25 days**	**Wed 3/22/17**	**Tue 4/25/17**		
25		Upload data to firebase and AWS for Apple Watch	5 days	Wed 4/19/17	Tue 4/25/17		Khushali
26		Testing shimmer application	5 days	Fri 3/24/17	Thu 3/30/17		Cheena,Khushali,Sanif,Zoosabali
27		Conversion into plugins	4 days	Wed 4/19/17	Mon 4/24/17		Cheena,Sanif,Zoosabali
28		Integrating into OTACS and Testing application	4 days	Wed 4/19/17	Mon 4/24/17		Sanif,Zoosabali,Cheena
29		**Final Presentation**	**7 days**	**Fri 4/21/17**	**Sun 4/30/17**		
30		Preparing presentation slides	7 days	Fri 4/21/17	Sun 4/30/17		Cheena,Zoosabali,Khushali,Sanif
31		Final Technical Report	7 days	Fri 4/21/17	Sun 4/30/17		Khushali,Sanif,Cheena,Zoosabali

Figure 4.1: Project Timeline-2

There were five main stages of development process:

- Requirements analysis (Figure 4-1): This task took 62 days. It includes analyzing business, functional and technical requirements. As, we are following Agile Methodology, new sensors were added to requirement later.

- Design (Figure 4-1): This task took 5 days. It includes analyzing software architecture, developing UI diagram and task division for implementation.
- Implementation (Figure 4-1 and Figure 4-2): This task included 32 days. This stage consist of two main parts:
    - Developing C# windows based Shimmer Capture application. This included capturing real time data from shimmer and saving it into csv format.
    - Developing iOS application to read heart rate data from apple watch.
- Integration (Figure 4-2): This stage took 25 days. This stage includes converting plugins and integrating with OTACS for shimmer application. This stage also includes Integration of Heart Rate Monitor with Firebase and Amazon Dynamo DB and exporting data into it.
- Testing (Figure 4-2): This stage took 5 days. The stage included testing both applications; shimmer and apple watch.

4.2 Task Division

Task was divided amongst team member in order to achieve the project goal within deadline. Most of the team members were new to C# and iOS development but they learned it and implemented that concept in the project successfully.

Task was divided amongst team member in order to achieve the project goal within deadline. Most of the team members were new to C# and iOS development but they learned it and implemented that concept in the project successfully.

The team members were assigned following roles: The team members were assigned following roles:

Task Name	**Explanation**	**Members**
Establishing shimmer connection	This task was to establish shimmer connection to the application so as to read the real time data.	Khushali, Cheena, Zoosabali, Sanif

Reading Shimmer data to the form	This task was to read the data and display it to form in readable format	Zoosabali and Cheena
Saving Shimmer data to csv and testing the application.	This task includes saving data to csv format as per the requirement and also testing the developed windows application.	Khushali, Cheena, Zoosabali, Sanif
Reading Apple Watch data and building GUI	This task includes developing apple application for reading data from watch to the application	Sanif, Khushali
Exporting iOS application data	This task include saving heart rate with respective date and time from application to firebase and Amazon Dynamo DB	Khushali
Integrating with S- Interface (OTACS).	This task includes integrating shimmer application with S-Interface.	Zoosabali,Cheena,Sanif

Table 4-2: Task Division

5. Conclusion

5.1 Project Summary

The Gryffindor team was successful in developing applications for reading GSR data and Heart rate data from Shimmer and Apple watch respectively. They were able to comply with the

requirement in their application. In order to achieve task within deadline work was divided and all the members did proper justice to work been assigned to them.

The team was able to learn new technologies and implement it is a short span in the project. Project was a good learning curve for all the team members. It was challenging for the team to understand the shimmer API and Apple heath kit API and to use it in the project. It was a good exposure to work with devices, understand its functionalities and been able to develop its application.

5.2 Lessons Learned

It was a good exposure to work with sensors. It was learning experience to implement its functionalities in the application. Most of the team members did not have experience with working on sensors. Therefore to understand the functionalities and finding the suitable API for its development was a lesson. Searching for necessary SDK and harness it in the application, necessary libraries and its use is a good experience. Integrating the application with S- Interface provide necessary learning because it required changes in the application calling so as to integrate successfully with S- Interface. One of the learning was to work with Firebase and Amazon Dynamo DB. Both are emerging Cloud platform. It was great learning experience to understand both technology.

5.3 Roadblocks

One of the roadblocks that intervened along project initiation was access to the SDK's. Most of the SDK's of the sensors were restricted to registered users only. All the applications was to be developed in C#. So, the foremost requirement to get in touch was with C# SDK of each sensors. But SDK with C# support was not available for Zephyr Bioharness. SDK was available in MATLAB which is not open source.

In case of Shimmer, the C# SDK was available. But understanding the relationship between the Shimmer device and its API was challenging. Most of the parameters are generated at run time. Thus, understanding connectivity between various parameters was demanding. Also, analyzing the data and its units was time consuming.

In case of Heart rate monitor application, saving the data obtained from watch into an excel file was challenging. After spending a lot of time in research for saving the data, Gryffindor was not able to create an excel and save data into the file.

Integrating Heart rate monitor application with Amazon Web Services was challenging. Due to limited open source resources, it was complicated to use AWSDynamoDB into an iOS application. Mapping each item of the application to AWSDynamoDB using AWS DynamoDB Modeling protocol was complex. Interacting of the application with the objects in the cloud using mapping class and object mapper was tricky. Also, assigning roles and policies in Amazon Web Services Console was little bit difficult to understand. Besides all such difficulties, Gryffindor was able to export data to Amazon Dynamo DB.

For Pan-Tilt Unit, the equivalent driver was not available that could connect the Pan-Tilt Unit with the SDK for performing Pan and Tilt movement.

In implementing the plugins for Shimmer using S-Interface, took lot of time because of enormous setting in configuration and references and bugs while integrating Shimmer Capture and Shimmer Plugins.

Also, for displaying data onto the graph was challenging. Gryffindor spend gradual time for displaying data on the graph.

6. References:

6.1 Coding References:

[1] AWS Mobile SDK, *Amazon Web Services* [Online]

Available: http://docs.aws.amazon.com/mobile/sdkforios/developerguide/Welcome.html

[2] Add firebase to your IOS Project, *Firebase Documentation* [Online]

Available: https://firebase.google.com/docs/ios/setup

[4] Timer Class, *Microsoft-.Net Development* [Online].

Available: https://msdn.microsoft.com/en-us/library/system.timers.timer(v=vs.110).aspx

6.2 Handbook Reference

[1] Shimmer Capture / Shimmer C# API, *Shimmer-Discovery in motion* [Online]

Available: http://www.shimmersensing.com/products/shimmercapture

7. Appendix

7.1 System Manual

Shimmer Capture:

Shimmer Capture application requires Visual Studio and S- Interface with its software dependencies. Visual Studio is downloadable from the Visual Studio website. Also one can download S-Interface its dependencies from link *http://cpl.uh.edu/software/s-interface/*

In order to successfully run the application one has to make sure below references are added in the Visual studio project. If the references are not added than the project will not build successfully.

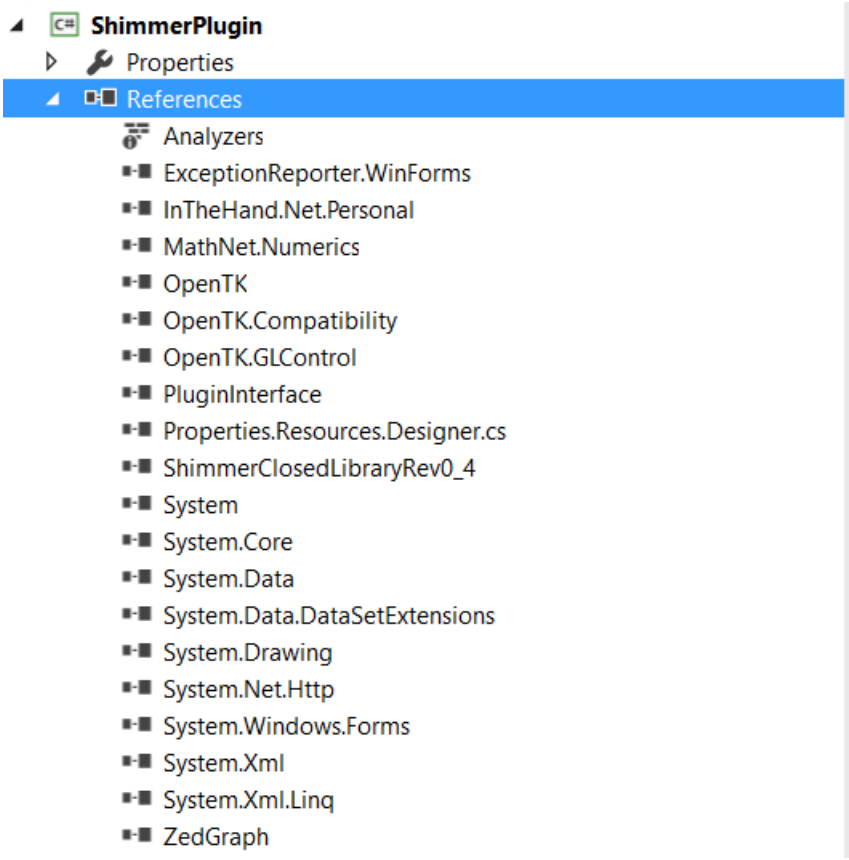

Figure 7-1: References for S-Interface with Shimmer plugin

Heart Rate Monitor:

Heart Rate Monitor requires Xcode. Application also requires to download cocoa pods for integration with Firebase and Amazon Dynamo DB.

7.2 User Manual

Shimmer

In S-Interface, we require three plugins connected which are as follows:

- Process Initiator: Output pin of process initiator is connected to input pin of Shimmer Capture plugin
- Shimmer Plugin: Output pin of Shimmer Plugin is connected to Input pin of real time graph.
- Real-time-Graph Plugin: Real-time graph shows the graphical view of the real time data read from Shimmer Plugin.

Button Functionalities:

- Process Initiator UI buttons:
    - Start: It will start streaming of the application. It will generate real time value and plots the same on the graph.
    - Stop: It will stop streaming of the application
- Shimmer plugins UI buttons:
    - Connect: It will connect device with the application. In order to start the streaming process, device should be connected to the application
    - Disconnect: It will disconnect the device with the application.
    - Save: It will save the data in CSV format. User can specify the name of the file and browse to the location where file is to be saved,

Display text:

- Status label: It will show the status of the connection between device and the application.
- Graph: Graph will show the real time GSR data coming as an input from Shimmer plugin.

Following are some additional screenshots of the system:

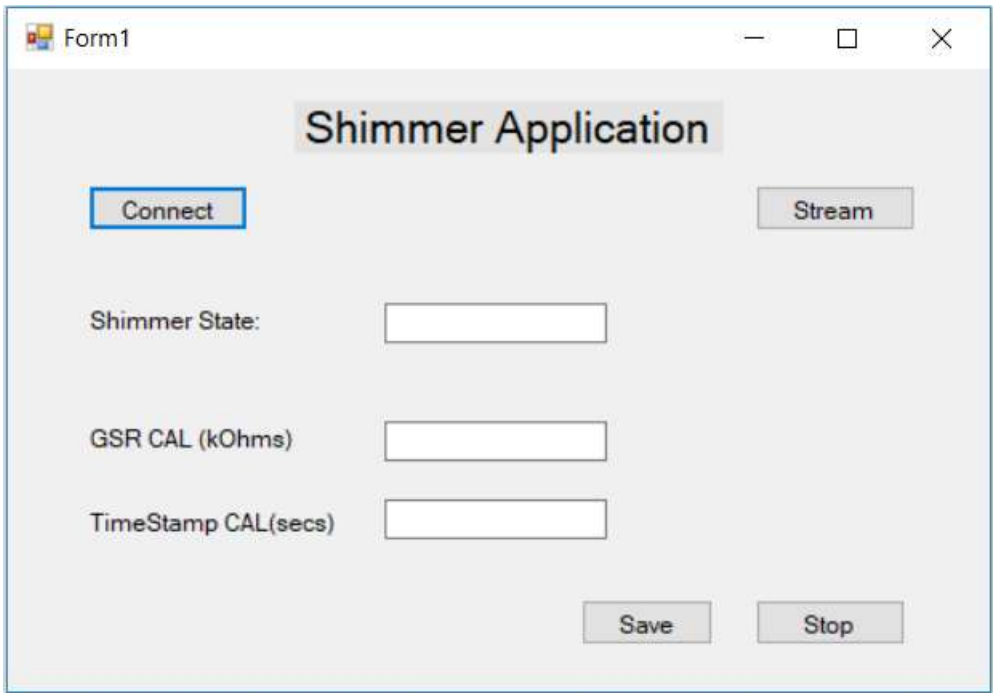

Figure 7-2: Shimmer standalone application

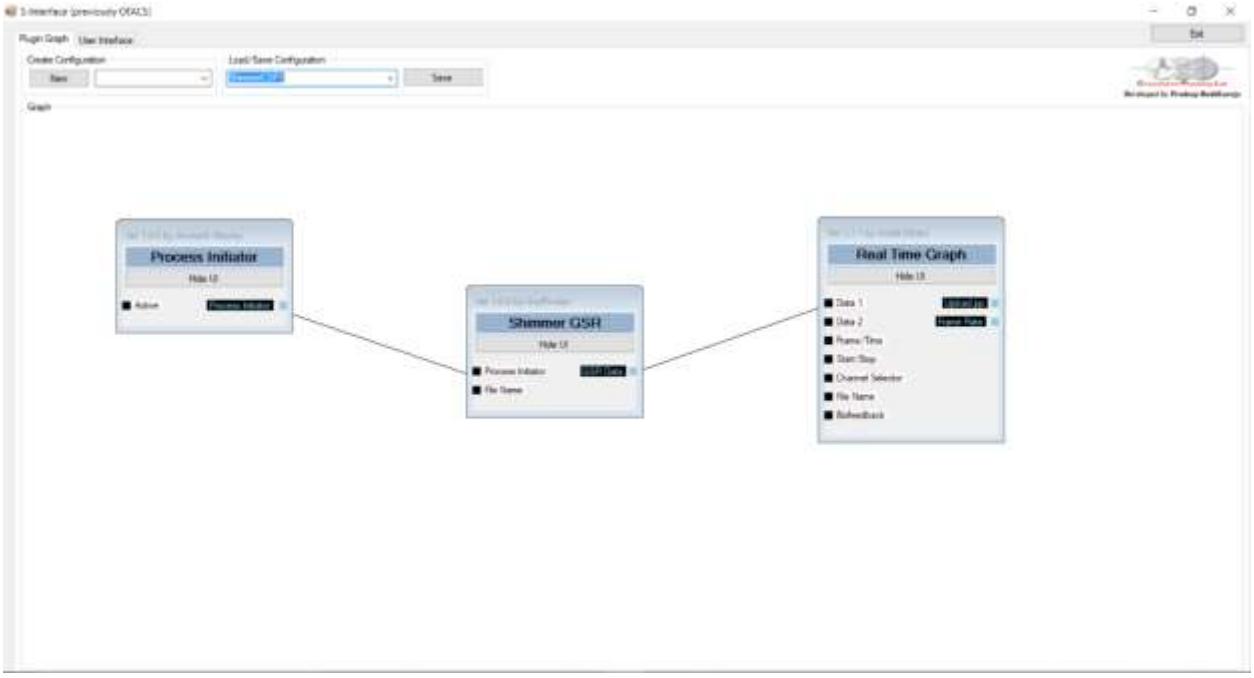

Figure 7-3: S-Interface with Shimmer plugin

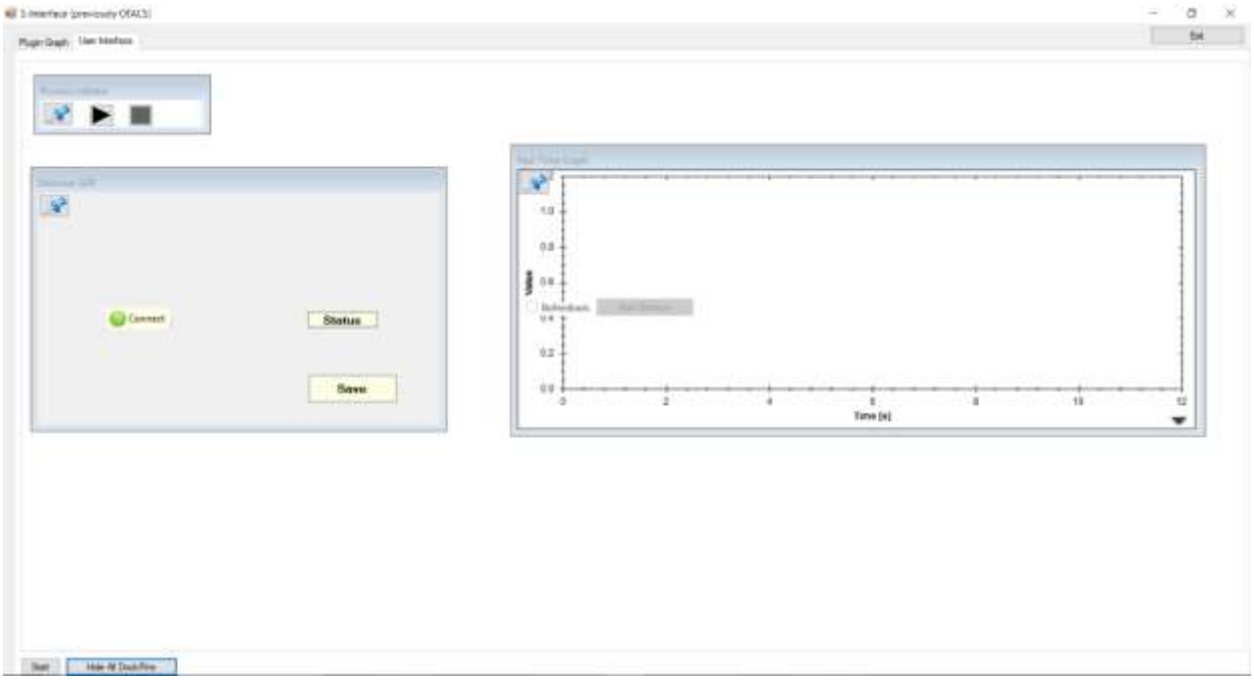

Figure 7-4: S-Interface with UI of Shimmer plugin – Before connect

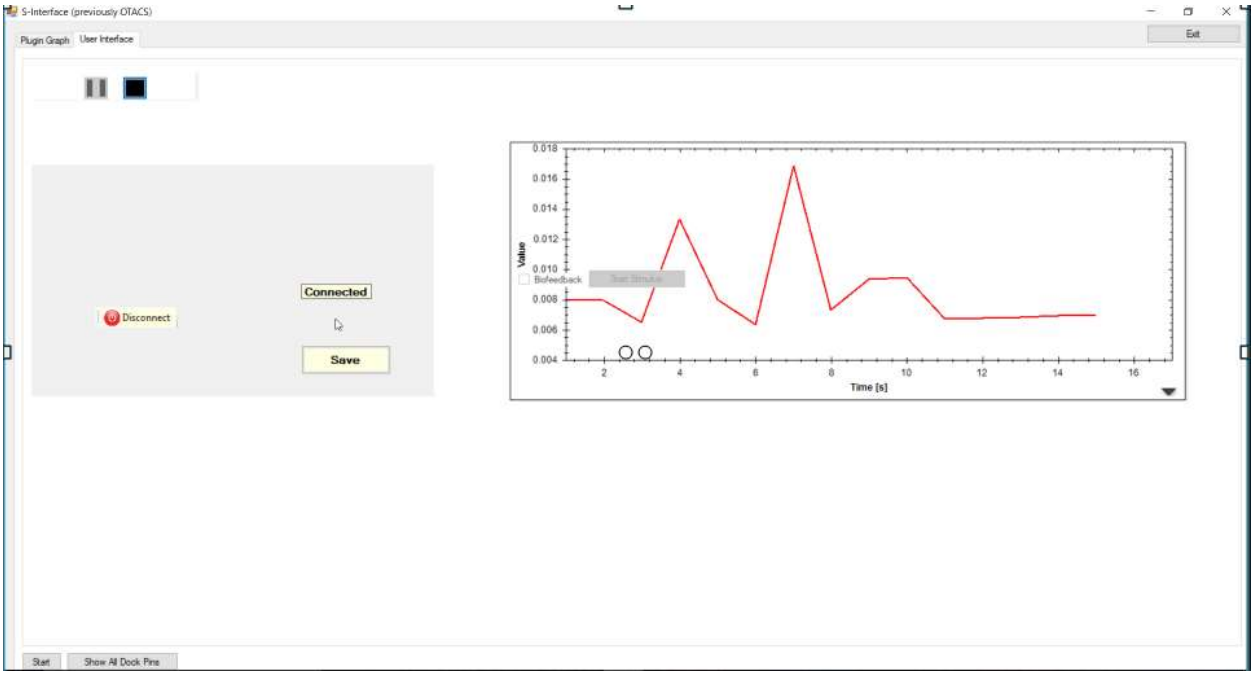

Figure 7-5: S-Interface with UI of Shimmer plugin – After connect

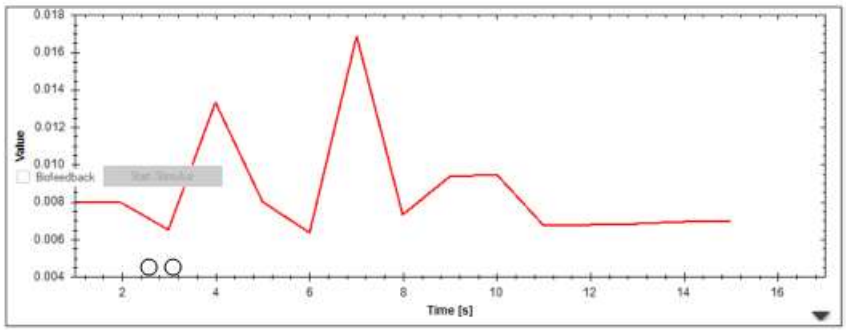

Figure 7-6: Real Time Graph

Heart Rate Monitor:

In order to save data externally in Firebase and Amazon Dynamo DB, following steps are necessary:

Firebase:

In order to import Firebase in the application, one must consider following dependencies:

- The application needs to have necessary pods for using Firebase and its database.
- Application must be connected to Firebase using bundle identifier.
- Moreover, a file named Google-Info.plist must be present in the application.

Amazon Dynamo DB

In order to import AWS in the application, one must follow following dependencies:

- The application needs to have necessary pods for using AWS and its Services.
- In AWS Console, it is necessary to create identity pool and name.
- This identity pool is required in Information property list of XCode.

- After creation of table in Dynamo DB console, it is necessary to set roles and policies.
- Credentials are provided in App Delegate of the Xcode application.
- Dynamo DB Object Mapper will help to save, update, retrieve and delete from AWS Dynamo DB.

Button Functionality:

Play: Starts streaming of heart rate with date and time measured from apple watch.

Stop: Stops streaming of heart rate with date and time measured from apple watch.

Display:

Three text box with most recent value of heart rate in BPM, time and date.

Following are some additional screenshots for Heart Rate Monitor application:

Figure 7-7: Heart Rate Monitor UI

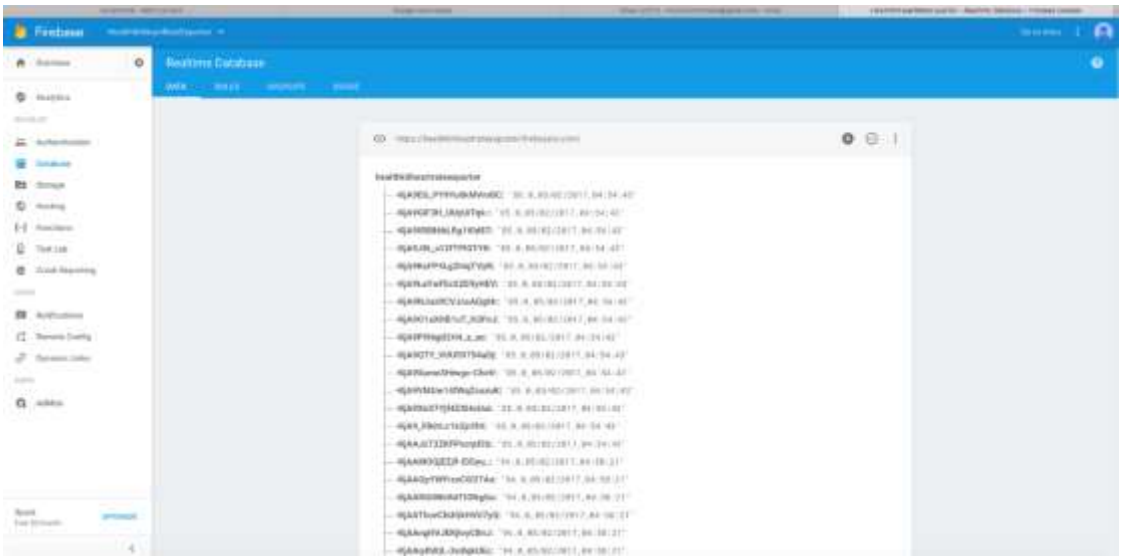

Figure 7-8: Firebase Console

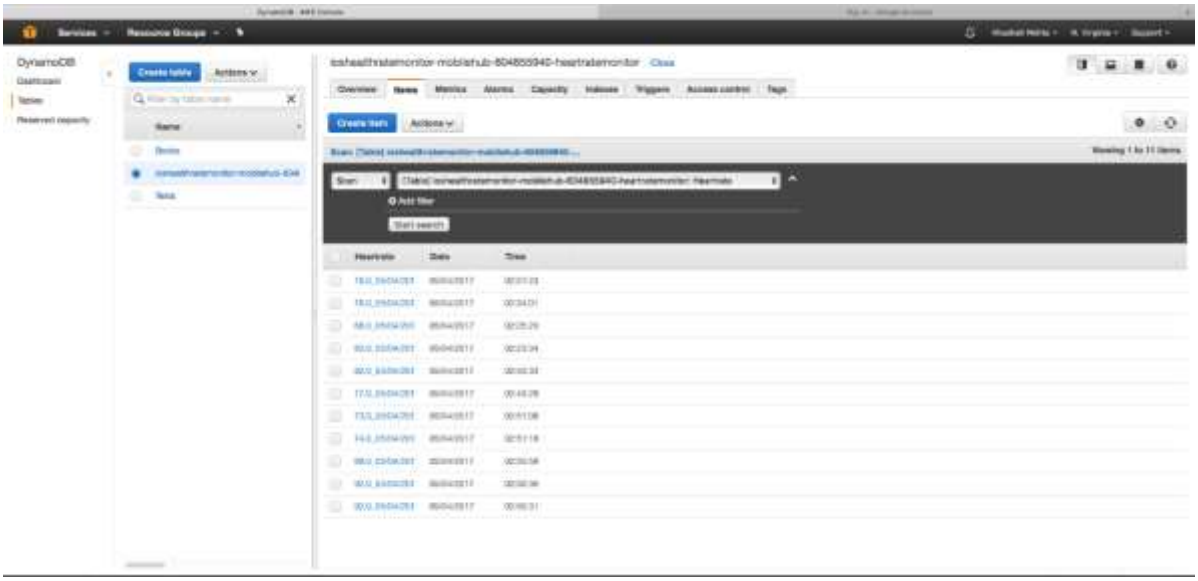

Figure 7-9: Amazon Dynamo DB Console