Mobile Computing and Internet of Things

Project Report

[ECEN 5023] -Cloud Connected Smart Wearable System

Date: 28 April. 2017

Team Members:
Name: Maitri Chattopadhyay
Contact: +1 720 755 7694
email: mach5953@colorado.edu

Name: Richard Noronha
Contact:+1 720 755 7820
email:richard.noronha@colorado.edu

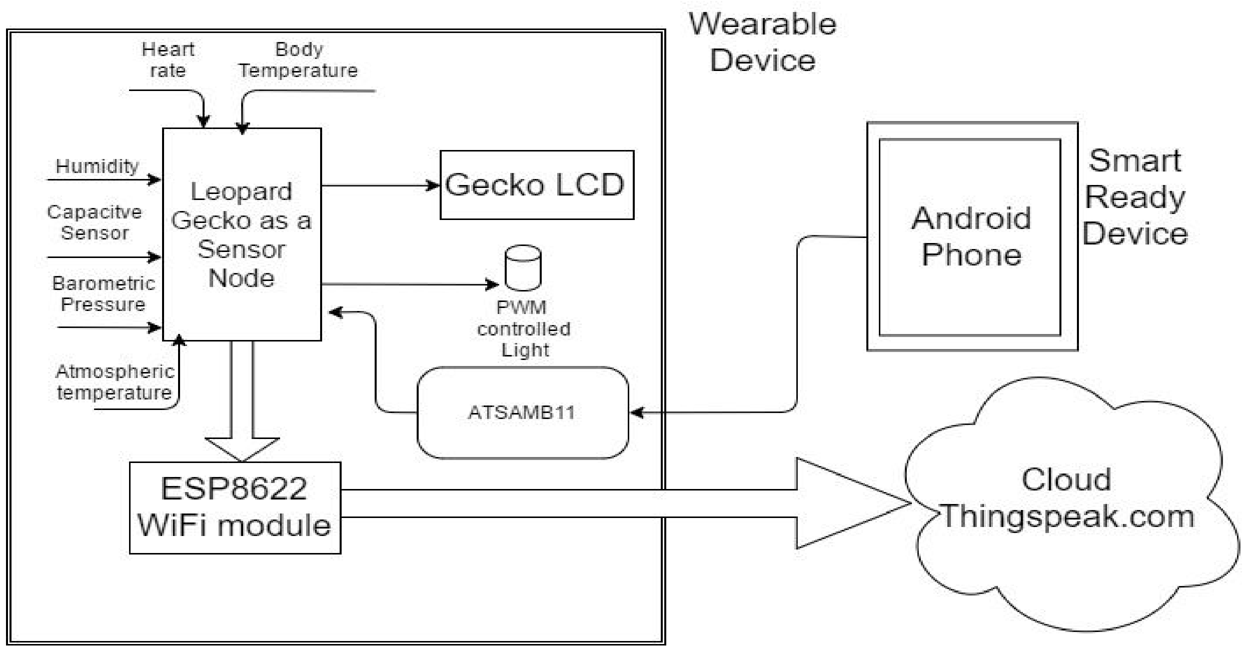
Objective:
To develop a Cloud based Smart Wearable system to make the user more situationally aware while on outdoor trips. All the data (atmospheric temperature, humidity and pressure, heart rate and the body temperature) is stored on the cloud so as to help him make better decisions about the location of such trips. Energy is conserved by giving the user the ability to turn the device on and off whenever needed.

Problem Addressed:

- Most wearable devices only track the important readings of the user wearing it. We live in Boulder, where the erratic climate plays a very crucial role in planning trips, so we wanted to make a system which not only covers the essential features of the existing wearable devices, but also make the user aware of other atmospheric features while on outdoor expeditions.
- We also wanted to store the data captured so that the user can refer it later. For this, we connected the device to the cloud whenever it is powered on. After establishing a successful connection, the device continuously logs data to the cloud.
- A device which interfaces many sensors consumes a lot of power and hence needs frequent charging. To reduce the frequency of charging the device, we have incorporated a feature which enables the user to turn on and off the device whenever needed from his smart phone.
- Measuring humidity will help the user of the device know when dew point is approaching. In colder climates once the dew point has been reached and when temperature is below freezing temperature, there may be icy roads outside. This can be very dangerous as there will be no friction when between the users shoes and the road. There is a greater chance of slipping and falling in these conditions. If this device is worn by a person on bike, this same principle can make the difference between life and death.
- The atmospheric pressure data is also useful in climates which are prone to storms. In a storm a low pressure region is created. The user of the device will thus know when a storm is approaching, much before hand. This data can also have other implications like calculating the altitude of the user especially when the user goes on mountainous trails.
- The LED light is like a torch that can be turned on by the user and even its brightness can be controlled. This will come in handy when it gets dark and the lighting in the vicinity is poor.

Hardware Block Diagram:

Wearable
device



BLOCK DIAGRAM

Figure 1: : Hardware diagram of Smart Wearable System

Photograph of the working device

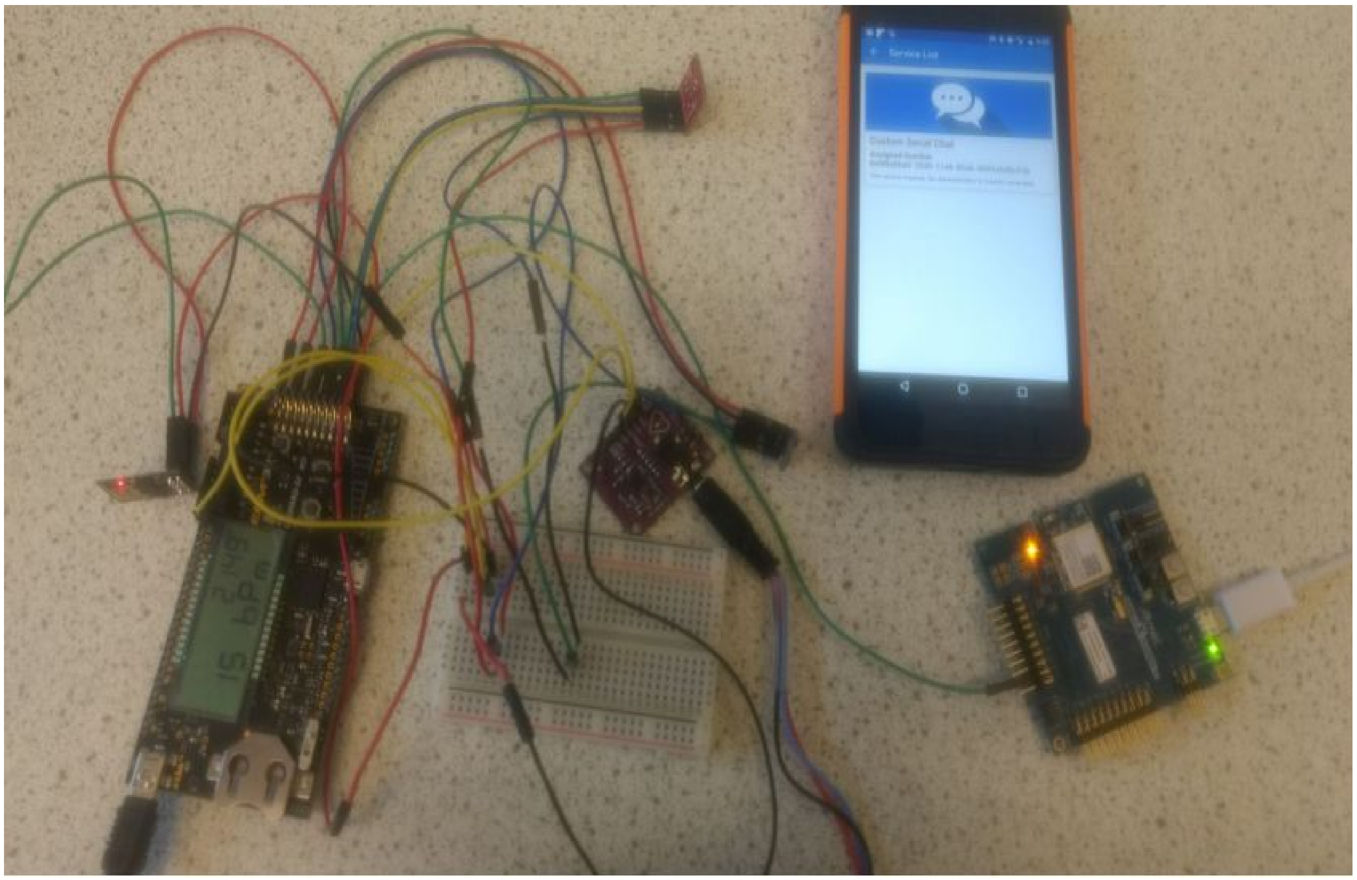


Figure 2: Photograph of the working device

Key Components:

- 1 Leopard Gecko
- 1 Heart Rate Sensor
- 1 ESP8266 WiFi module
- 1 BME 280 (Pressure, Temperature and Humidity Sensor)
- Internal temperature Sensor
- Ambient light sensor
- Atmel SAMB11 Xplained Board
- Cloud server (Thingspeak.com- An open IoT platform with MATLAB analytics)

Target person:

Venture Capitalist or Angel investor to make this into a product or business. The target customer is anyone who actively goes on outdoor trips like hiking, biking etc

Software Flow Diagram:

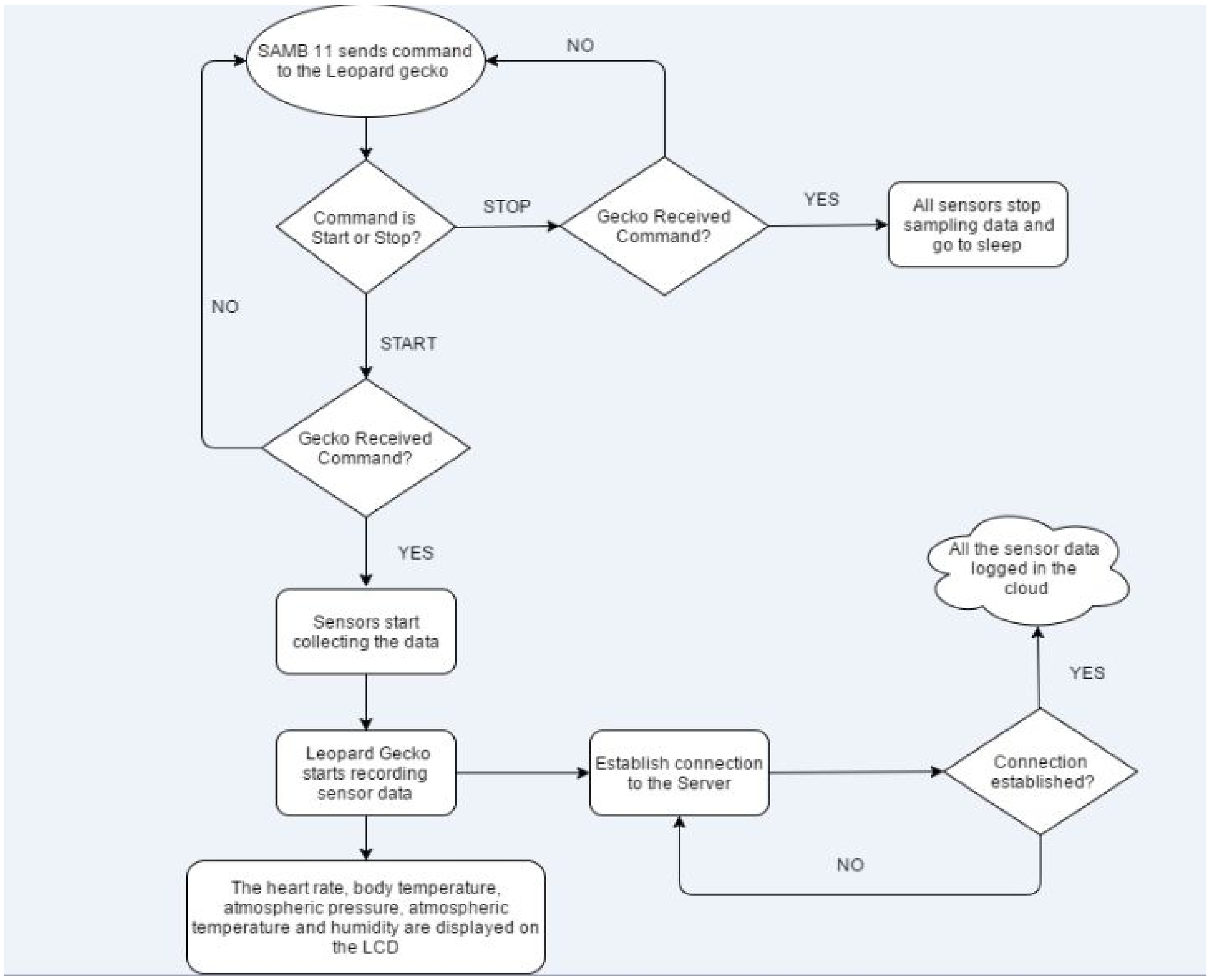

SAMB 11 sends command to the Leopard gecko
NO
Command is Start or Stop?
STOP
Gecko Received Command?
YES
All sensors stop sampling data and go to sleep
NO
START
Gecko Received Command?
YES
Sensors start collecting the data
Leopard Gecko starts recording sensor data
The heart rate, body temperature, atmospheric pressure, atmospheric temperature and humidity are displayed on the LCD
Establish connection to the Server
Connection established?
YES
All the sensor data logged in the cloud
NO

Figure 3: Software Flow diagram of the Wearable System

Testing and Verification:

Detailed testing and verification spreadsheet is attached at the end of this report. Below are the energy profiles and the data analyser graphs recorded while testing the device.

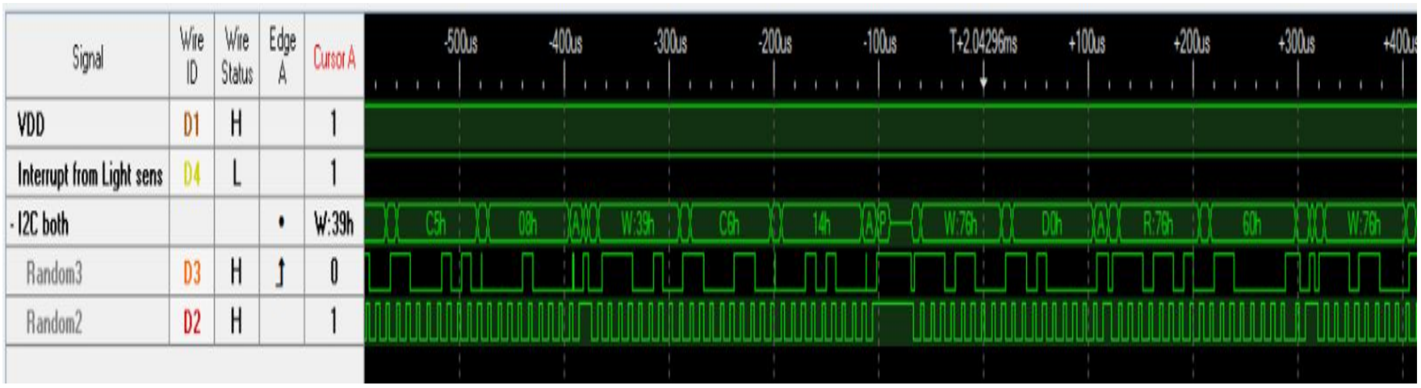
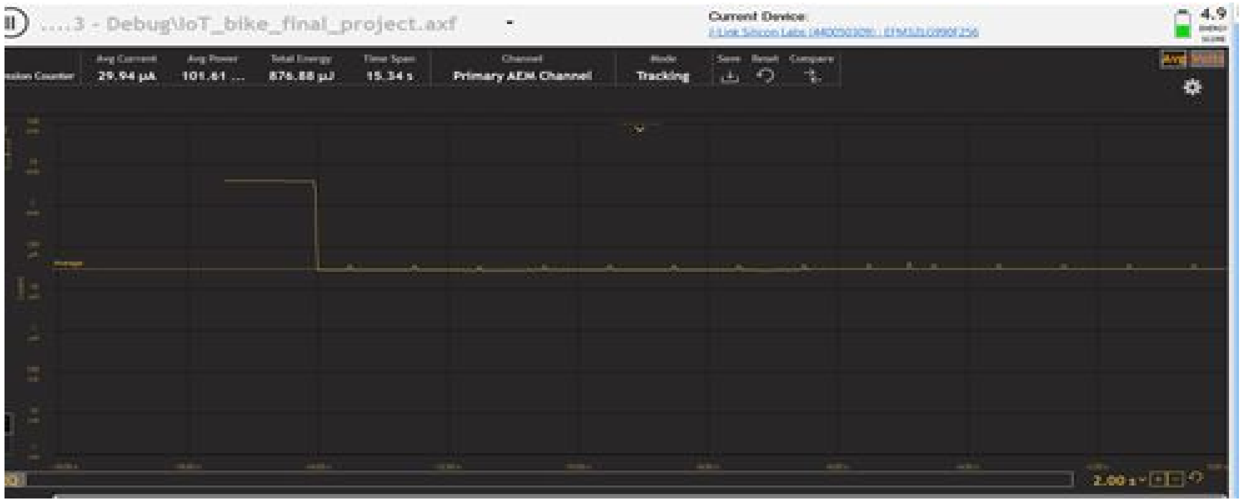


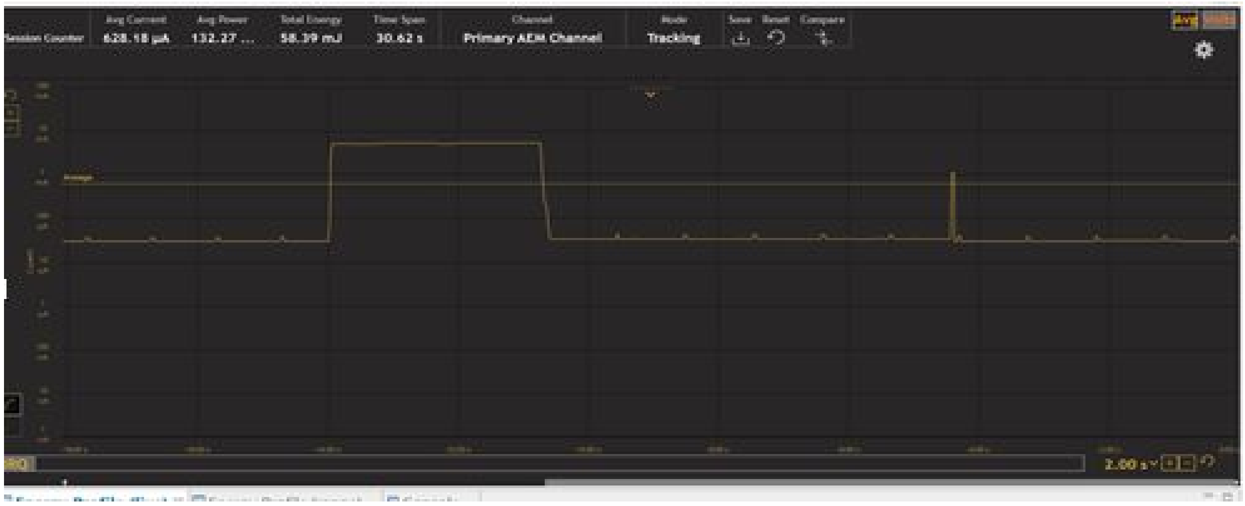
Figure 4: Multiple read writes to different I2C devices done successfully


....3 - Debug\IoT_bike_final_project.axf

Current Device:

Avg Current	Avg Power	Total Energy	Time Span	Channel	Mode
29.94 µA	101.61 ...	876.88 µJ	15.34 s	Primary AEM Channel	Tracking

Figure 5: The initialization of the device



Avg Current	Avg Power	Total Energy	Time Span	Channel	Mode
628.18 µA	132.27 ...	58.39 mJ	30.62 s	Primary AEM Channel	Tracking

Figure 6: The first RTC and corresponding Turn off

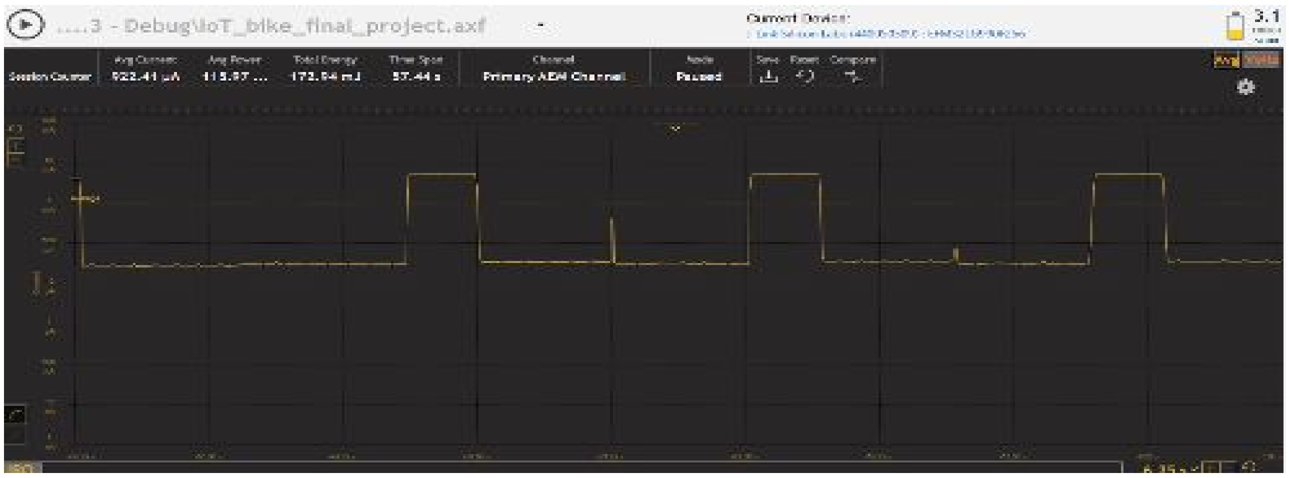


Figure 7: Multiple RTCs together

List of commands used in the Program:
The device is automated and is capable of operating with least assistance from the user, the commands needed from the user are kept at minimal.

- The user is required to give commands from his smart phone when he wants to turn the device On or Off. The default state of the device is ON.
- He can also turn the onboard LED (which acts as a torch light for the device) by touching the LESENSE Capacitive sensor.

List of trigger points in the program:

- Initially all the various on-board modules are initialized and set up. The last one to be set up is the Real time clock
- The Real time clock(RTC) is the main underlying clock which controls when the sampling is to take place. This is set at every 15 seconds sampling.
- When the RTC interrupt of COM0 fires, the sensors are set to sample the data. This includes the BME280 and the Heart rate sensor. The heart rate and body temperature modules do not work in the first interrupt but start from the second interrupt. This is because the DMA needs to sample data.
- In this same interrupt, the data is displayed on the LCD in quick succession. This adds a slight delay but greatly improves the Human Interface.
- The ESP8266 then takes that data from a circular buffer and transmits it to the Thingspeak.com server.
- The LESENSE Capacitive sensor starts scanning every 1 second as soon as the device goes to sleep.

- In the COM1 of the RTC, the LCD is disabled and the device goes into Energy Mode 3.
- An asynchronous interrupt can be received from the GPIO pin which is connected to the ATSAMB11.
- The SAMB11 receives the command from the Android device to turn on or to turn off the device.

Development Schedule:

Activity	**Planned to complete**	**Percent complete**	**Completion Date**
Capacitive Touch Sensor	3/23/2017	100%	3/25/2017
Capacitive Touch Sensor using LESENSE	4/10/2017	100%	4/10/2017
PWM output	3/27/2017	100%	4/12/2017
Gecko LCD interfacing	3/29/2017	100%	4/2/2017
Interface BME280	4/22/2017	100%	4/20/2017
Heart Rate Sensor interface	4/7/2017	100%	4/17/2017
BLE connections	4/26/2017	100%	4/27/2017
ESP8266 interface	4/24/2017	100%	4/24/2017
Pushing data to the cloud	4/26/2017	100%	4/25/2017
Report and documentation	4/27/2017	100%	4/27/2017

Project Road Blocks/Difficulties:

- The Heart Rate sensor on most occasions does not power on properly which affects the reliability of the sensor being detected and hence it cannot generate the necessary
- An open-source program NodeMCU can be loaded to the ESP8266 to make the board connect to the server using LUA scripting. The ESP8266 module very

rarely fails to connect to the internet and push data to the cloud. This might affect the cloud data pushing feature of the device

- We could not establish multiple profiles in which one takes (heart rate) input from the Gecko and one gives input to the Gecko(Serial chat). Instead a single profile with a custom serial chat interface was used to give a power control command to the Leopard Gecko.
- Due to multiple sensors interfaced on the same board, there are multiple number of interrupts that are needed to be handled. Some interrupts are difficult to pick up and hence the PWM feature does not seem to be working
- The ESP8266 lua code only allows us to send positive integer values and not negative or decimal values.
- The display on the LCD has to go through a wait loop for the values to display.

Summary of the functionality of final project:
The project comprises of six constantly updating readings used to monitor different factors. It is built with the inspiration of providing the user with one gadget which is enough to rely on for planning outdoor trips. The various software modules in the project are described below:

1. Data Collection - This module deals with the data collection by the Leopard Gecko Board from the sensors connected to it.
    1.1. Temperature, Pressure and Humidity data collection: This module reads the current atmospheric conditions.
    1.2. Heart Beat data collection: This module reads the heart beats of the user in one minute.
    1.3. On-board temperature sensor: This module is meant to collect the samples of the body temperature of the user.
    1.4 LESENSE Capacitive Touch Sensor: This module is meant to control to the brightness of the LED. The basic functionality of switching the LED on and off can be observed easily.
2. BLE Node - This module deals with Atmel ATSAMB11 Interaction with the Leopard Gecko and sends data from the BLE Smart Ready Mobile phone to the Leopard Gecko.
3. Server data- This module deals with the integration of the ESP8266 WiFi module with the Leopard Gecko in order to transfer data to the Server. Thingspeak.com is the server that has been used. Thingspeak is an open IoT platform with MATLAB analytics.

4. Power management: The Leopard Gecko works at the Energy Mode 2 mode when data has to be sent out. It further goes into Energy Mode 3 when sleeping. Along with this we have seen to it that the sampling of data only takes place once every 15 seconds. The pushing to the cloud takes place every 1 minute. Using a BLE application on an Android phone, we can turn the sampling and sensing of the Capacitive touch sensor OFF. This feature further enables us to save power for longer usage with a battery.

Future Scope:
Given the limited time to implement the project and the various problems faced along the way, many of the features we initially had in mind had to be eliminated.

- The BLE device can use multiple profiles on the Smart Ready Device.
- A more reliable heart rate sensor needs to be used with the device.
- The light sensor can directly interact with the LEDs.
- Custom characters can be used with the BLE to control the Leopard Gecko wirelessly via the Application.
- The user should also be given more flexibility as to what sensors he would like to use and what data is to be sent to the cloud.
- The server can be customised further and analysed.
- The device can be put into Energy Mode 4 and controlled via the GPIO. This feature is an extension of an already available feature of turning off all sampling.

Lessons Learned:

- Analysis and design of the clocking of the application so as to conserve the most amount of energy possible. This matters when we consider the power consumption.
- Interrupt handling with 6 interrupts simultaneously running along with a DMA routine has been implemented in this design. This was possible through a circular buffer and enabling and disabling interrupts as and when required.
- The operation of the ESP8266 module. There are different ways to use the ESP module to connect to the server. One method uses AT commands which is tedious and power expensive. A much more power economical way of executing the same functionality is to use firmware with a LUA script which is already written to the device before even mounting it to the microcontroller. As a result the number of UART commands to the ESP8266 are minimum thereby saving significant power.

- The ESP8266 currently works at a baud of 9600 on the LEUART. This means that the device can be in Energy mode 2 when transmitting the data to the module.
- Collecting multiple sensor data and display the data on the server. An open IoT platform with MATLAB analytics is an ideal solution for data like the ones we are collecting. The channels had to be set and write needed to be made to it.
- How to program the BLE to transmit data to the Leopard Gecko.