GoActive

A Wearable Device for Posture Tracking

Peter Fidelman
University of Washington
Computer Science and Engineering
fidelp@uw.edu

Zishen Hu
University of Washington
Electrical Engineering
shen0924@uw.edu

Ka Ho (Steven) Guh
University of Washington
Electrical Engineering
kguh@uw.edu

Zachary Amador
University of Washington
Computer Science and Engineering
zea@uw.edu

ABSTRACT

Many studies have linked a sedentary lifestyle to health problems. Unfortunately, it can be inevitable in some cases, such as after major injury. Doctors and healthcare professionals want to track the activity levels of their patients, and especially how they change over time. There are existing wearable devices that track time spent sitting, but they are either unreliable or expensive.

We introduce GoActive, a posture monitor worn on the hip. GoActive recognizes when the wearer is sitting, standing, and moving. It logs data about the activity of the wearer to its internal storage, and sends this data over Bluetooth Low Energy to a compatible iOS device, where our app shows statistics about the user's activity.

Keywords

quantified self, human activity recognition, pattern recognition, wearable sensors

1. INTRODUCTION

GoActive is a belt-mounted device that monitors its wearer's posture throughout the day, sending this data to an iOS app for real-time examination. Data logging and retrieval is also possible using the app.

GoActive was inspired by widespread interest in long-term posture-tracking. Studies have linked more time spent standing to positive health outcomes. Researchers want a small, convenient posture-tracking device that they can use for further research. Consumers, too, may wish to track their own activity to understand and improve their own health. Some existing products cannot meaningfully classify posture; others are bulky and inconvenient to wear, are too expensive, or lack real-time feedback. We aimed to address these problems by making GoActive inexpensive, easy to wear, and provide real-time feedback.

This paper will describe the GoActive's capabilities and implementation, our team's design process and testing procedures, and future directions the project may take.

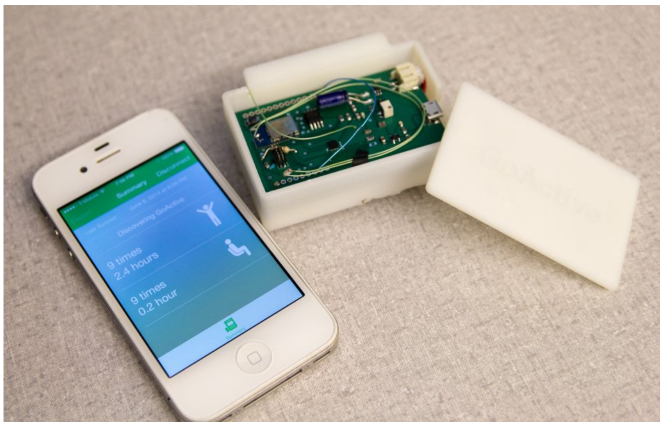

Figure 1: The GoActive prototype and iOS app

2. RELATED WORK

Several existing consumer and medical devices try to solve the problem of posture tracking.

No consumer device solves the entire problem. There is a wide variety of consumer activity-tracking devices based on accelerometers, such as Fitbit, Jawbone, Lumoback, etc. These conveniently mount to the wrist or belt. While they do a decent job tracking overall mobility, they aren't good enough to distinguish whether the user is sitting or standing, mainly due to their location. Another project, SmartMove, takes an opposite approach – pressure-sensitive insoles. These could certainly distinguish sitting from standing, but only if the user is wearing shoes – not too useful in a medical setting. Furthermore, SmartMove is just a Kickstarter project and is not yet available.

As for medical devices, the most prominent is the Paltech

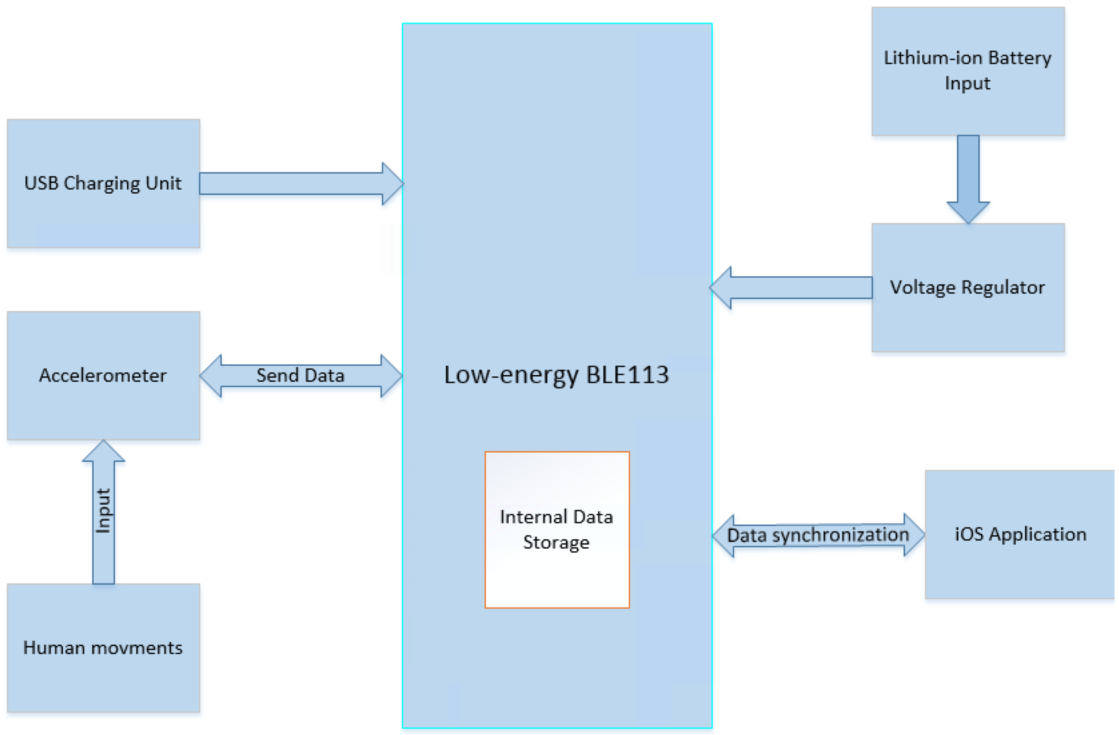

Figure 2: Block Diagram of the GoActive

activPAL leg-mounted accelerometer. Its stated purpose is posture tracking, and many medical researchers use it for just that. While it undeniably works, it is not a satisfying solution: expensive, bulky and attaches via proprietary sticky pads. There is plenty of room for a competitor.

3. TECHNICAL DETAILS

3.1 Overview

The components of the full GoActive system were:

- Hardware
- Case
- Firmware
- iOS App

3.2 Theory of Operation

The GoActive firmware performs data collection, posture recognition (via machine learning), and posture logging. It transfers data to the iOS app over a BLE connection. The app, in turn, allows users to visualize and explore their posture history and receive real-time feedback. Briefly, the firmware's purpose is data collection and classification, while the app's purpose is data aggregation and visualization.

We designed GoActive as an autonomous device – it should not break as soon as it leaves the smartphone's Bluetooth range. If the phone is expected to be absent for long periods, the device must be able to log long time periods to internal storage, and running out of space (or battery) is instantly fatal. The best way of preventing this is to keep the device's storage requirements modest. We were able to quickly rule out logging raw accelerometer data , instead focusing on extracting the events the user actually cares about: transitions between postures.

3.3 Implementation Details

3.3.1 Hardware

The GoActive's custom PCB has a BLE (Bluetooth Low Energy) module, a three-axis accelerometer, a USB charging unit and a voltage regulator. The BLE 113 module is produced by Bluegiga, a company who provides short-range wireless connectivity solution, and BLE's design is based on the Texas Instruments CC2541 low energy bluetooth system-on-chip with on-chip antenna.

The three-axis accelerometer is used to detect human postures and generate raw data for BLE 113 module to determine for pre-defined postures. The GoActive wearable is currently limited to sitting and standing postures, and more postures like running, walking, or even biking can be added to GoActive system using the same accelerometer and move advanced software control.

GoActive is powered by a rechargeable 1100mAh Lithium-ion battery and a battery management controller, the Microchip MCP73831T. The capacity of the battery can last for at least a week under intense use in terms of frequent data synchronization. The charging circuitry is designed to be fast and safe such that the battery takes 2 hours or less to fully charge. In addition, low battery detection circuitry informs users about the battery status. A red LED indicator represents low battery state. The long battery life provides convenience to users as well as overhead power so more fea-

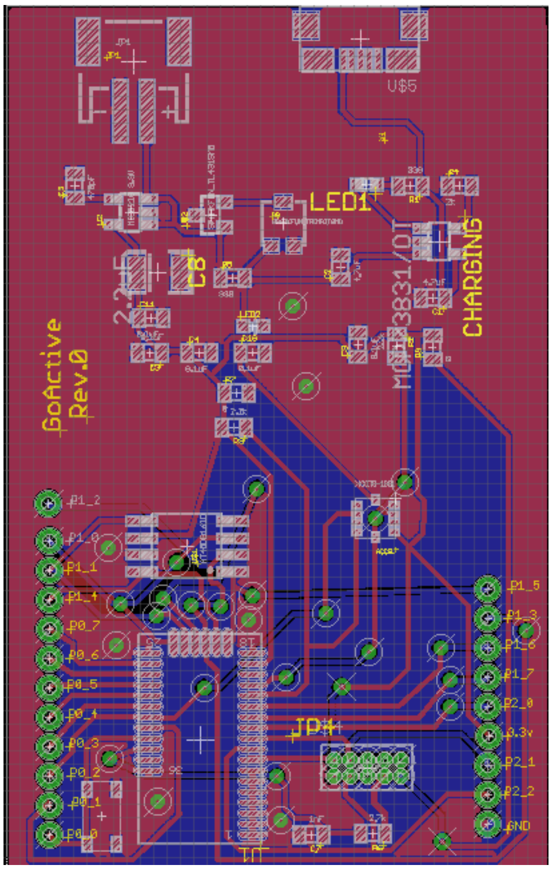

Figure 3: PCB layout

tures could be added to the system in the future.

To ensure all components in GoActive system operating under safe voltage range, a voltage regulator is utilized to maintain a constant voltage level. The voltage regulator chip is a MICREL MIC5219. The MIC5219 is an efficient linear voltage regulator with high peak output current capability, very-low-dropout voltage, and better than 1% output voltage accuracy. The power rail of the GoActive is 3.3V which is a safe and stable voltage for every component in the system.

In addition, the GoActive system has a 16Mbit external SPI flash memory chip along with the internal storage on the BLE113 module. Software support for the AT45DB161D flash memory is currently under testing and will be added to the system in the future. The 16Mbit capacity will allow the GoActive system to store data for up to ten weeks between synchronizations with an iOS device without loss of data.

3.3.2 Firmware

The GoActive firmware is responsible for converting raw accelerometer data into a series of timestamped postures (SIT/STAND/etc), then buffering them until they can be sent to the smartphone app. These tasks must coexist with the BLE stack and its overhead, as all run on a single CC2541 SOC with 8k of RAM.

At the highest level, the firmware's structure was dictated by a design decision Texas Instruments made years ago. Specifically, TI does not provide source code to their BLE stack, just object files. These are divided into ten tasks, one for each level of the stack. Unfortunately for the implementor, these tasks expect to be called by TI's OSAL (Operating System Abstraction Layer) effectively dictating its use. They do not run in interrupt context, meaning all user code must adhere to strict – but ill-defined – timing and memory-usage constraints so as not to disrupt the BLE stack and cause disconnection. Additionally, the overhead of OSAL, the BLE stack, and the accelerometer driver, took roughly 6K of RAM, leaving a bit under 2K for GoActive's custom tasks. Some creativity was required in dealing with these limitations.

GoActive is built around a circular buffer of timestamped postures. The firmware's responsibilities are thus simple: enqueue new postures to this buffer as the user moves around, and dequeue/send postures whenever a BLE connection to the GoActive smartphone app is available. We configure an OSAL timer (presumably implemented by a real timer interrupt) to schedule the main GoActive task every 100ms. This 'tick' is GoActive's atomic unit of time.

To enqueue new postures to the buffer:

- Every 100 ms (one tick): the accelerometer is sampled once, and this (x,y,z) triple is placed in a 20-triple buffer. The resulting sample rate is 10Hz.
- Every 2 s: this buffer fills, and is passed to the machine learning functions for feature-detection and classification. The exact nature of the ML classifier will be described in the Machine Learning section. For now, know that it outputs one posture. The posture is placed in a 5-long buffer for deflickering.
- Every 10 s: the 5-long buffer fills. Majority voting is used to pick a 'dominant' posture from this buffer, which is then timestamped. (in other words, deflickering through majority vote). This datapoint is then logged into the main circular buffer of postures, if and only if it is different from the last posture. That is, only posture transitions are logged.

The worst-case runtime (new sample triggers ML, then ML triggers deflickering and logging) is rare, occurring only once per one hundred ticks. Moreover, testing reveals that even this worst case does not last long enough to disrupt a BLE connection.

To dequeue and send postures:

- Every 100 ms (tick): check for an active BLE connection; if one is found, repeatedly pop a datapoint from the main circular buffer and send it as a BLE Notification, until the circular buffer is empty.

This scheme has one fundamental weakness: if there is no active BLE connection, the circular buffer eventually fills up and any newly-collected posture data will be lost. This is where the flash memory comes in. If an append causes the circular buffer to fill completely, the buffer contents should be dumped to flash, and the buffer emptied. This prevents posture data from being overwritten or lost. Unfortunately, we did not have a PCB with a working flash chip ready until less than 36 hours before the Mother of All Demos deadline; although a flash driver was written far in advance, there was

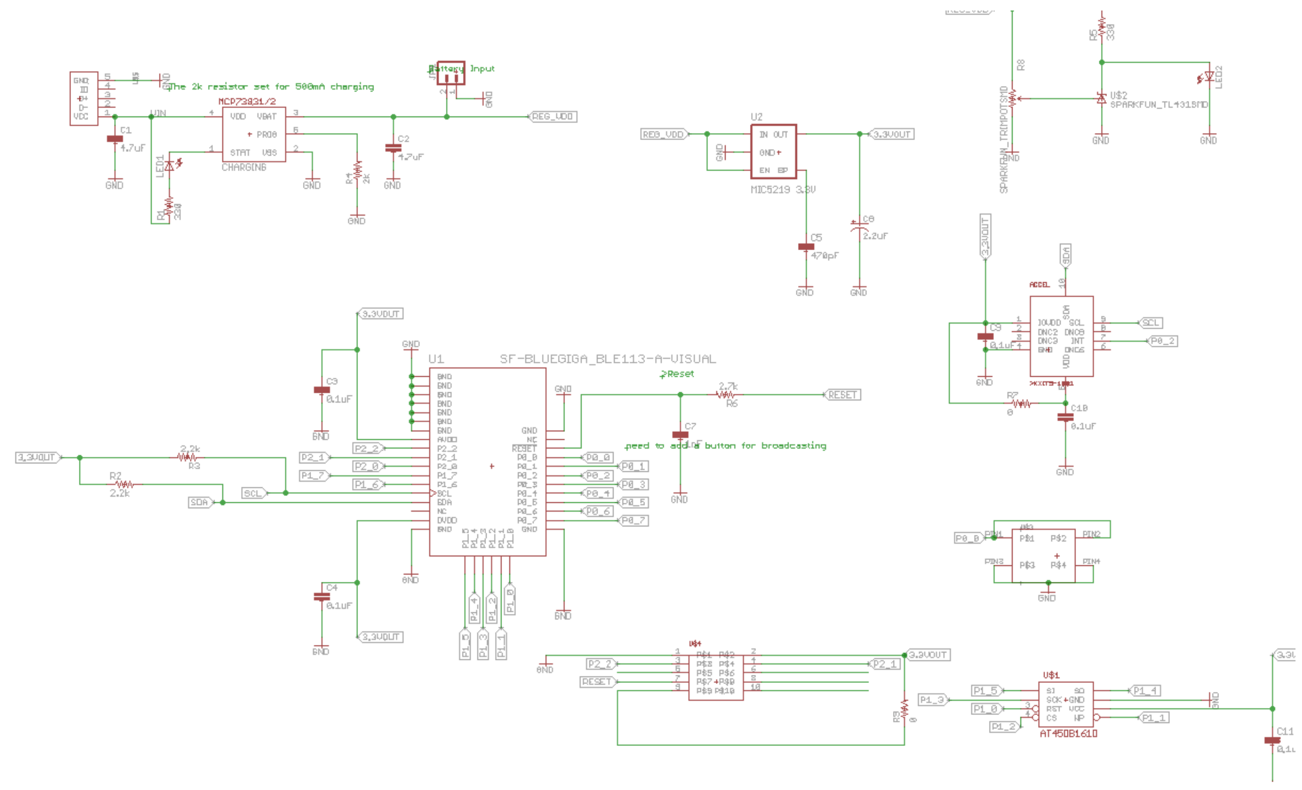
Figure 4: Schematic of PCB

simply not enough time with the hardware to sufficiently debug and test it. The lack of working flash memory limits the time GoActive can remain disconnected from the smartphone before losing data. Fortunately, there was enough free RAM to buffer approximately 30 minutes of data, and this fallback plan proved sufficient for our demos.

3.3.3 Machine Learning

First, three features were extracted from three-axis accelerometer data in a two-second window. They are variance of movement intensity(VI), mean of movement intensity, normalized signal magnitude area (SMA). Movement Intensity is defined as the Euclidean norm of the total acceleration vector after removing the static gravitational acceleration, where $a_x(t)$, $a_y(t)$, $a_z(t)$ represent the t^{th} acceleration sample of x, y, and axis of the accelerometer[2].

$$MI(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2} \tag{1}$$

Movement intensity is independent of the orientation of the accelerometer; however, this feature is a time series data, which just like raw accelerometer data. Therefore, movement intensity were not used directly in the machine learning algorithm. Instead, average and variance of movement intensity were used.

$$AI = \frac{1}{T}(\sum_{t=1}^{T} MI(t)) \tag{2}$$

$$VI = \frac{1}{T}(\sum_{t=1}^{T}(MI(t) - AI)^2) \tag{3}$$

Furthermore, normalized signal magnitude area (SMA) were used to feed in the machine learning algorithms. SMA is defined as the acceleration magnitude summed over three axes within each window normalized by the window length[2].

$$SMA = \frac{1}{T}(\sum_{t=1}^{T}|ax(t)| + \sum_{t=1}^{T}|ay(t)| + \sum_{t=1}^{T}|az(t)|) \tag{4}$$

To detect posture, the three features extracted from accelerometer data over two seconds window were used to feed into number of machine learning algorithms, such as logistic regression, k-nearest neighbor (KNN), and neural network. 40% of the total data was partitioned into test set, and the reset was used to feed in the machine learning algorithms. The accuracy of three algorithms on the triple axis accelerometer data on test set was above 90%. However, large number of floating point calculations and large memory were needed in order to compute those machine learning algorithms.In order to fit a machine learning algorithms in the embedded systems especially the 8051 microcontroller inside of the BLE 113, we discover condensed nearest neighbor (CNN). CNN is similar to KNN except, it uses a reduced set of data pool and classifies new data using KNN when k equals to one.

To condense data using CNN, three features were first extracted out from the raw accelerometer data. Second, we choose a random classified sample x in the pool. Then, we find a nearest classified sample y that is not same we chose using Euclidean distance. Remove x in the pool, and add to a second pool. Repeat the process from finding the nearest classified sample until no more data is add to the second

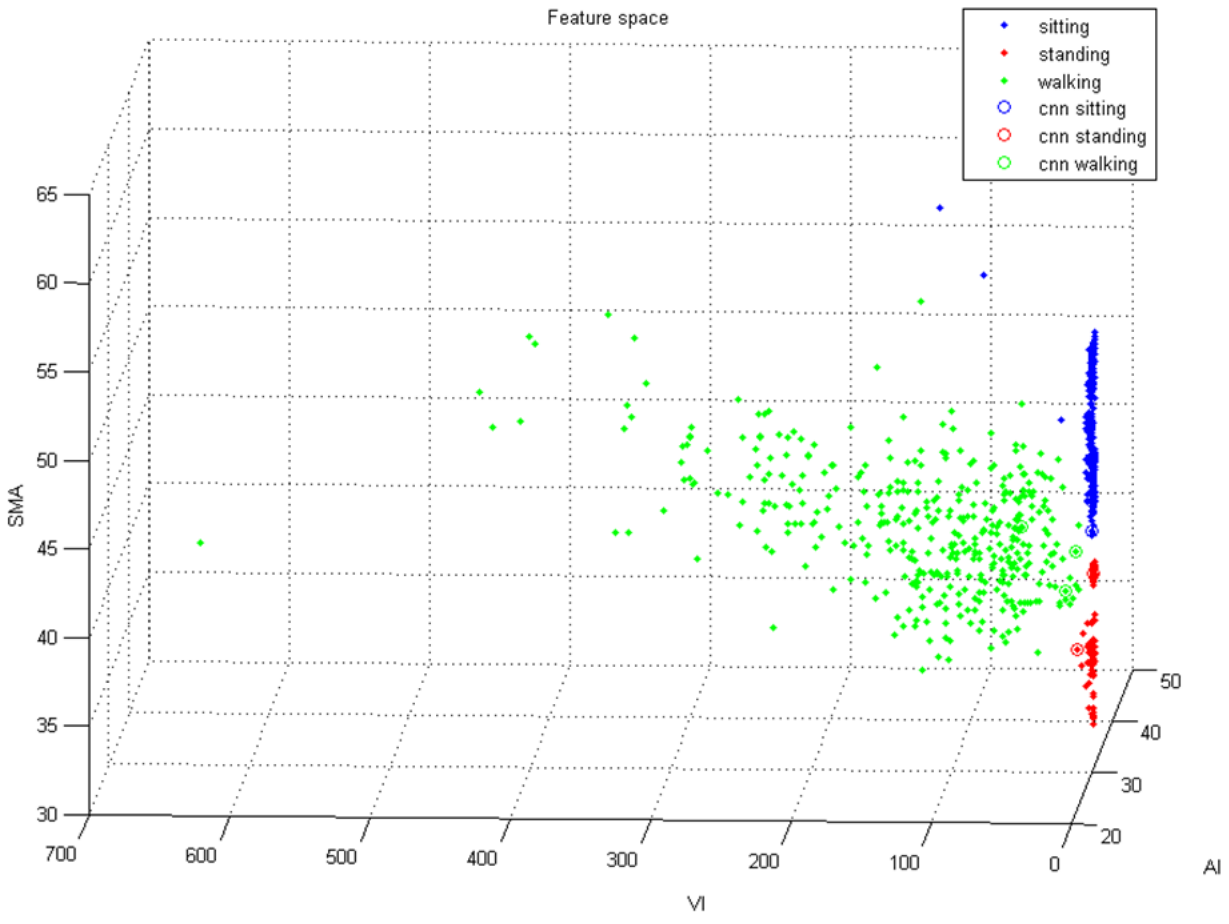

Figure 5: Graph of the three features we extracted from the accelerometer data with a two-second window

pool.

Once this process is done, we have a condensed data set and the original data set. We managed to condense of our large pool of data set to 6 point of data as shown in Diagram 4. We would not able to do this if our data does not form distinct cluster or if our data have a lot of noise. Fortunately, we have the data that met those conditions. The CNN classifier has high accuracy (98%-100% dependent on the data randomization) on the test set, which was partitioned from total data as mention previously.

When the data pool is relatively small, the CNN does not use a lot of computing power and memory, because the algorithm finds the nearest point in the data pool against the new sample by computing the euclidean distance of three features Concretely, our embedded system extracts those three features, which are VI, AI, and SMA, from a 2-second window, and finds the nearest data point in the data pool, then classifies the new sample point as the nearest data point's classified label. This process is relatively less computational hungry, and memory inexpensive compare to other machine learning algorithms. Therefore, KNN was an ideal machine learning algorithm for us.

3.3.4 *iOS Application*

The iOS application in this project served as a visual representation of the data in the GoActive device. Our application used Core Data, which is an iOS framework developed by Apple that provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management[1], as its object model to store all data from the GoActive device. The application provides a connect and export button and visual representation of transition count and the duration of sitting and standing as shown in Diagram 5.

When the connect button is pressed, the application will first check if the application is allowed to use Bluetooth and the application has bluetooth available. Second, it will discover any device nearby that has offered the unique bluetooth service. Third, it will try to connect the bluetooth and turn on the notification of specific characteristic. Once the notification has turn on, iOS app process the the notification update on the GoActive. We managed to used the notification to received historical data in the device, and sync the current posture update. The first notification received is expected to be a special state that has a timer timestamp representing the timer in the GoActive device when the iOS device turned on the notification. Then, we will store a wall clock time stamp when this special state has received. After we received this special state, the appication is expected to received different states of posture with a timer timestamp that has the same reference in the first special state. We can calculate the exact wall clock time on each posture data by subtracting the stored wall clock time stamp by the special state timer timestamp and summing the posture timer timestamp. The notification will update a single posture data at a time, the application will check if the incoming posture data has the same posture state as the last posture data. If they are the same, then the application will ignore the incoming posture since logging this new incoming posture does not provided any new information. Also, this will likely happened when the GoActive device is in sync mode (finished transferring all historical data) because the sync mode in the device will send any data in a 10-second period. If a user wants to terminate the bluetooth transmission with the GoActive device, simply click disconnect button or press home button. If the user wants raw data in the Core Data, a export button is designed to export all data in Core Data as csv format.

4. EVALUATION AND RESULTS

The PCB was able to detect pre-defined postures, currently limited to sitting and standing. All collected data was stored in BLE 113 module's internal flash, and processed in BLE before synchronize with iOS application. The user interface of iOS application displayed real-time feedback about number of times of sitting and standing users have done, and the duration of each category in hours. Software infrastructure worked satisfactorily. The firmware successfully logged posture data and transmitted it to the iPhone app, which in turn successfully logged and analyzed it. The system's main weakness was the inaccuracy of the detected postures. In our own testing, the device produced accurate results, but during the Mother of All Demos many spurious posture transitions were detected. Some possible explanations of this are presented in the Discussion section.

5. DISCUSSION

There are several possible reasons for the disappointing Mother of All Demos results.

- The device was never trained for its wearer, Tien. It is possible that his body geometry or movement style was different enough from our group members' that our ML classifier's predefined sitting/standing/etc. points were not accurate for him.

- During the demo, Tien wore the device far forward on his hip. The device was designed to sit flat against the right side of the beltline, and this is how it was trained. The difference in location would likely have produced unfamiliar accelerations. Providing clearer instructions with the device (such as a diagram) might mitigate this problem in the future.

- There may be unaddressed problems in our design. Maybe we need to use a more sophisticated machine learning algorithm. Maybe the firmware isn't sampling at high enough rate. Perhaps a hip-mounted device is not sufficient to detect posture changes in 100% of wearers.

During testing, we concluded that ML accuracy was sufficient (as GoActive works reasonably well on all members of our team). This demo proves there is much room for improvement. If further development takes place on the system, it would be prudent to recruit a much larger group of testers, preferably with diverse body geometries.

A few new features can be added to GoActive system utilizing the same hardware in the near future. For instance, running and biking are very popular exercise in people's lives, it would provide higher quality of exercise outcome with GoActive system that keeps track of the users' postures and to provide more details about their exercise cycles. Users will have accurate and sufficient data to vary their exercise accordingly. Working out with consistency is important for achieving fitness results.

There are three improvements can be made to the hardware of the current GoActive prototype. The first one is to reserve clearance area on the PCB for better connectivity of on-chip antenna. The clearance area will provide longer range reception and also stronger wireless signal when performing data synchronization between BLE113 and iOS application. Clearance area means that there should not be any passive components or ground plane that possibly generate signal interfere with antenna signal. The second improvement is to minimize ripple voltage from power supply. A large capacitor was added to system stabilize output voltage from voltage regulator. It is always a good idea to make input voltage as consistent as possible. Lastly, all decoupling filters should be as close as possible to power source to block certain noise and passing others. Any small signal in low energy circuit could have significantly affected the whole system in mysterious ways.

6. CONCLUSION

In this paper we have introduced GoActive, a hip-mounted posture tracking device that distinguishes sitting, standing, and walking states. As part of a senior capstone project, our group designed, built, and tested a working prototype. In the wearables market, GoActive is unique: many existing products cannot meaningfully classify posture; others are bulky and inconvenient to wear. GoActive is important as a specialized, real-time posture-tracking device with the potential for both consumer and scientific uses.

7. ACKNOWLEDGMENTS

The team would like to thank Shwetak Patel for running this capstone class. He gave us enormously useful advice and direction, and connected us with Dori Rosenberg from Group Health to get feedback on the needs of medical researchers.

We'd like to thank Dori Rosenberg for giving us a lot of insight into the inadequacies of existing posture tracking devices, which helped inform our design.

We would also like to thank Tien Lee, the Teaching Assistant for CSE477. He was very helpful for getting us hardware to prototype with, giving us advice and feedback on our PCB and case, and testing the device for the Mother of All Demos.

8. REFERENCES

[1] Apple. Core data programming guide. `https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/Articles/cdTechnologyOverview.html`, 2014.

[2] M. Zhang and A. A. Sawchuk. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In *Proceedings of the 6th International Conference on Body Area Networks*, BodyNets '11, pages 92–98, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).