SpinBit: A Wearable Rotation Counter

Katie Kemp, Adam Ryman, Isaac Perry
CSE 477, Spring 2015

ABSTRACT

The quantitative self-movement has created hundreds of niche devices for recording and analyzing specific details of everyday life. In this work, we present SpinBit, a lightweight wearable device that allows the user to locally record and share their average angular velocity. Our system uses a three axis gyroscope/accelerator to evaluate when a user completes a revolution and records that data locally, and allows the user to upload that data to a mobile android device for analysis and sharing. SpinBit has applications in performance arts and dancing where users may want to increase their spins over short or long periods of time and/or compete with other performers in spins. Furthermore, our device is awesome.

Keywords

Spins, wearable, sensing, gyroscope

1. INTRODUCTION

In dancing and performance arts, performers often do not have the capacity to hold a multifunctional device (smartphone) on their person for clothing and/or weight restrictions. This drastically reduces their ability to gather data about their respective performances. We envision a wearable device that is very small and lightweight and self-contained that a user can clip to themselves to gather rotational data.

Currently all smartphones and several other devices that people use regularly have 3 axis gyroscopes already included. Theoretically spin detection should be trivial to implement on these devices in software, but the problem is that these device are either bulky, difficult to attach to a variety of clothing and/or an embedded device that is not easily programmed. SpinBit solves this problem by being designed specifically to do one thing, record spins, and do it very well.

Figure 1. Contra Dancing is a type of dancing that involves lots of rotations, and thus a fun application of the SpinBit. [1]

2. THEORY OF OPERATION

Small lower power gyroscopes have been on the market for several years; their footprint is so small and their power usage so low that they have been incorporated into every smartphone ever created. MEMS Gyroscopes (the type of gyroscope on circuit boards) use the Coriolis Effect to detect angular velocity along an axis of rotation. The premise is that there is a small object vibrating along a linear axis up and down, and this object affixed to a surrounding structure without impeding its vibrations [2]. If you take the plane that the object is vibrating within and rotate the plane, the object will want to keep moving in its linear direction and will move (from a non-rotating perspective) in the direction opposite of the rotation. The movement brings the vibrating object closer / farther from fixed walls and the capacitance between these objects can be amplified and outputted as the sensor values.

Mounting the gyroscope so that it is parallel to the user's body in the standing vertical direction, we are able to detect the angular velocity of the user in the plane that is perpendicular to their standing direction.

This angular velocity data can then be integrated to find the users angular position relative to the time the sensor values were first read. Once this angular position exceeds 2*pi radians we record the time that has passed since the last event. This time is logged locally on the device and can be requested via Bluetooth.

3. SYSTEM IMPLEMENTATION DETAILS

The SpinBit system consists of a physical device clipped onto the user's waist band and an app that this clip-on device connects to, where the user can see the spin data collected.

3.1 Hardware

The SpinBit uses an ATMEGA328 microcontroller as the central part of the system. There is a button on the device which when pressed by the user signals to the processor that it should begin collecting spin data. When the button is pressed a second time, the device exits data collection mode and waits for a Bluetooth signal from the android phone app requesting spin data.

The microcontroller is connected to a MPU-6050 triple-axis combined gyroscope and accelerometer. When in data collection mode, the processor requests acceleration data from the gyroscope.

The SpinBit stores spinning data in a directory of text files hosted by a microSD card. Every time the processor determines that a spin has happened it notes the time since the device entered data collection mode, and saves this value into storage in a text file created for the current spin session. The SpinBit uses a Bluetooth to communicate with the android application. When the application requests data, the device sends the oldest spin session text file on the microSD card over Bluetooth to the application. When the SpinBit receives conformation from the app that a data set has been successfully sent, it deletes the text file associated with the oldest spin session currently in storage. Whenever the device is powered on and not recording, it is waiting for a command from the mobile device.

There is a reset button that will not be easily accessible to the user as well as pin headers that we used for debugging and programming. The power switch is located on the side, so as to be easily accessible to the user. The SpinBit uses a 3.7V lithium polymer re-chargeable battery. The board that we designed

includes as charging IC and micro USB port for charging the battery.

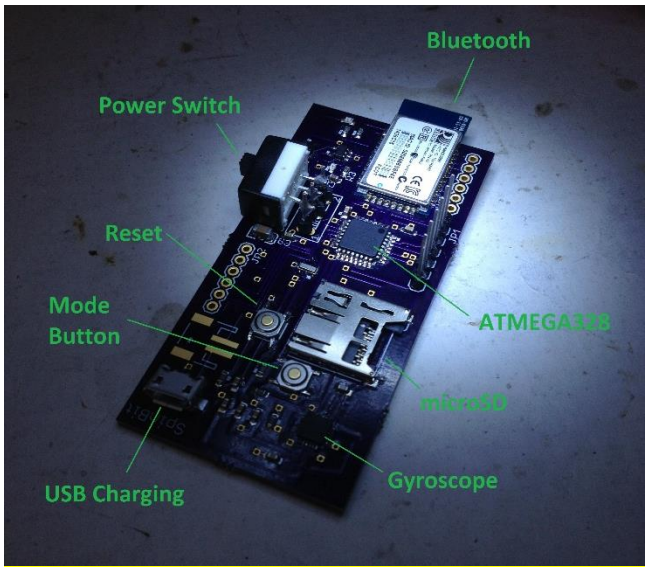

Figure 2. In an effort to make our prototype device as small as possible, we custom designed the PCB used.

3.2 Data Processing

All data processing is currently done on the SpinBit in real time concurrent with data collection. When in data collection mode, the processor requests the angular velocity from the gyroscope. Because of the orientation that the gyroscope is mounted in, we look just at the Y component of the gyroscope data; this component corresponds to the plane of rotation experienced by a dancer wearing the SpinBit in the proper orientation. The value read by the processor from the gyroscope is adjusted based on the current level of sensitivity of the gyroscope to obtain a positional value in degrees. If the processor determines that the user has not changed direction, it adds the degrees rotated to a running sum. When this sum reaches 360 degrees, a spin is logged. If the processor determines that the user has changed direction it checks if the sum is over certain threshold. If the sum is over this threshold it logs a spin before resetting the sum.

The prototype that we are demoing has a simpler version of the data processing on it that does not check to see how far the user has rotated when a change of direction occurs.

3.3 Android Application

The main features of the app are to retrieve data sets from the SpinBit and display data sets to the user. These main features require a number of secondary features: Bluetooth configuration; establishing a Bluetooth connection between the mobile device and the SpinBit; requesting datasets into and putting them into permanent storage; and enabling the user to browse different data sets.

The home page of the app has four buttons: "My Spins", "Download from SpinBit", "Reconnect" and "Set Up". "My Spins" will display a list of all the data sets that have previously been downloaded. "Download" requests a data set from the SpinBit and saves it into permanent storage. "Reconnect" initiates the app side of the Bluetooth connection with the SpinBit. "Set Up" allows the user to specify the unique name of the specific SpinBit this mobile application is capable of connecting to.

Before the user employs any of these features there is some initial setup. First, the user must pair their mobile device with the SpinBit. This is done through the settings of the mobile device. Bluetooth should be turned on and the device is visible. Select the name of your SpinBit and the devices will be paired. Now the app needs to connect to the SpinBit.

Initially the app is incapable to making a Bluetooth connection to a SpinBit. In order for the mobile device to connect, the application needs to know identifying information about the Bluetooth module the SpinBit is using to make a connection. The homepage of the app has a "Set Up" button. Pressing this button brings up a dialog where the user can enter the name of the Bluetooth module in the SpinBit. The name is saved as a preference and will persist even after the app is closed. At any time the user can press this button to change the name of the SpinBit the app is able to connect to. This must be done at least once before any other of the apps functions can operate, but once it is done the app is now fully operational.

If the app has a name for the SpinBit it will connect to, it will automatically connect at startup. After specifying a name or if the connection was disturbed, the "Reconnect" button will initiate a new connection between the application and the SpinBit the app knows the name of.

Once the app has been properly set up and is connected with SpinBit, it is ready to receive data. To request a data set from the SpinBit press the "Download" button. This sends a request to the SpinBit, which must be in transfer mode. The SpinBit will send characters over Bluetooth to the mobile device; these characters are stored in a buffer in the app until a specific character sequence that signifies the end of the data set is added to the buffer. This sequence is a newline character followed by a return character ("\r\n"). Once a whole data set has been received a data set object is instantiated which holds the data from the SpinBit, a Timestamp for this data set and a name that the user can set later. This data set object it is added to a library of data set objects, which is implemented with a List. This library is regularly backed up and saved to a JSON file. The data sets in this library can be accessed by the user.

The app lets the user browse their saved data sets, tag information about their data sets, and view visualizations of the data sets. Pressing the "My Spins" button from the home screen displays a list of datasets, ordered chronologically. The data sets of spins have names and timestamps associated with them; these are displayed in the list along with the total number of spins in that set. The user is able to click on a data set and is taken to a new page. This page displays the name of the data set, a graph of the data. The data that is saved are time periods in milliseconds between spins captured by the SpinBit. This data is converted into angular velocity in complete spins per second and the app generates a graph of this trend for the user. Below the graph there are buttons which launch dialogs that allow the user to specify a name for the data set, and modify the date and time for this data set.

4. EXPERIMENTAL PROCEDURE AND RESULTS

Most of the testing for the SpinBit consisted of testing each individual section. For instance, we tested parts on a bread board, got what we had connected working before we added more parts. Once we had things working on a breadboard we designed a PCB based off of our breadboard design.

4.1 Breadboard Design and Testing

After device conception we quickly ordered parts to build a functional prototype on a breadboard. We used this prototype to test various assumptions about product feasibility. To keep our budget low we ordered the cheapest breakout board components we could find. We chose the atmega processor with the Arduino development board because of it's easy of use and heavy community support and documentation. After hooking up and configuring each component we built three basic systems, one to put data onto a SD card, one to send data from the SD card to Bluetooth, and one to process the sensor data.

4.2 Tests of Spin Detection

Initially we took transferred our raw sensor data to a PC and used Python to get a sense of what processing techniques would give us clean spin detection. The first task was correlating the raw sensor values with actual angular velocities and positions; we did this through a combination of reading our datasheets to find theoretical correlated values and then doing controlled spins to tweak these values for a full 360 degree rotation. After being able to reliably detect a rotation while moving in a single direction, we started work on detecting when the user changed directions and how to deal with partial spins. We eventually determined that we could sense if the user was changing direction by storing a small amount of the previous sensor values in memory and checking if the running sum of sensor values were trending in the other rotational direction. If this was the case we would check how close the user was to completing a rotation and if they were within 80% of a spin we would count that as a spin event. This 80% of a spin check was also run at the end of a spin session to determine if the user had almost made a spin before ending the session.

We then later moved a simpler version of this processing code to the embedded device code.

4.3 App Design and Testing

The app was built in many stages, each adding a single feature at a time. The first feature implemented was the home screen, containing four buttons that initially did nothing.

The next step was to implement model object classes that represent a single data set; they store the data, the date, the time and the name of the data set.

The next hurdle was connecting from an android app to an Arduino over Bluetooth. A separate app was made that would simply be able to communicate with an Arduino and toggle an LED. Using an open source app that implemented a Bluetooth connection as a guide, the app was able to connect with another Bluetooth device and exchange data. Then we wrote an Arduino sketch which would listen to serial communication and if it received the correct character it would toggle the LED connected to one of its' digital output pins. We were able to verify that the Bluetooth connection was working because pressing a button on the app would toggle the state of the LED. Once this was working properly, we added the Bluetooth management code to the home screen implementation in the original app. Before we could test this, we needed to build a SpinBit emulator on an Arduino; this device would wait for a data request, generate a random data set and send it over serial through a Bluetooth module to the application. This was verified using console logs in both the Android and Arduino software. We then added the details of handling data coming from the emulated SpinBit, using that data to instantiate the model objects and save them in a collection. This step was also verified using console logs in the Android application.

While this data could be stored in a collection available to the app, this collection would not persist if the app was closed. We next added the ability to save the contents of the data set collection in a local file on the mobile device. This was verified by logging the contents of the collection to the console, closing the app, and then logging the collection to the console again; the system works properly if the contents of the collection didn't change after closing the app.

Once the data persisted, the next goal was to add a screen that would allow the user to browse the data sets. There is a screen in the Android library that accepts a collection and displays the contents in a list-view. We were able to confirm this feature was working properly when the contents of the collection logged to the console matched with what was displayed to the user when they launched the new screen.

The purpose of browsing data sets is to choose one and have it visualized for the user. We wanted to create a graph of a single dataset for the user. We were able to find an Android library that would accept an array of data and graph it on a screen. We built a separate app that took an array of fake data and graphed it for the user. It was easy to verify the library worked because we knew how the fake data set should look when graphed. Once this was working properly, we added a feature that when the user clicks on an item in the list-view of data sets, it would first launch a new screen. It also passed the data set to the new screen. In this new screen we included the graphing library and graphed the dataset for the user. We confirmed this was working by comparing the graphs to the data sets logged in the console.

The last features added were displaying more information to the user about the data set with the graph, and adding buttons that let the user modify this information.

We then extensively tested the app by doing everything in our power to break the app. One of the first noticeable bugs was that values would reset (including the values that maintain the Bluetooth connection) when the orientation of the mobile device changed. We were able to fix this issue by changing a setting that forces all of the apps' screens to retain their state through an orientation change. This also allowed the app to retain its state when it is paused, in the background.

Another issue to be tested was protecting the integrity of the data. We tried exiting the app and shutting down the device in places where the data was being handled. Because the data is saved in a local file and backed up any time it was changed the data was sufficiently protected from corruption.

The last outlet for bugs was the Bluetooth connection. The app was able to handle various edge cases of the data sets: an empty data set, a very large data set, a data set containing negative numbers and a data set containing zeros. Measures were taken in the design of the app to handle each of these scenarios in a reasonable way that doesn't break the app or cause an inconvenience for the user.

4.4 Full-System Testing

The app and the SpinBit device were designed and built separately and the last step for each side was integrating the two halves of the product. The only interface between the two sides of the system is the Bluetooth communication. Because each side was able to independently connect and communicate with other devices, we were able to verify Bluetooth was working for each side properly;

as a result we were able to develop and test both sides knowing only each other's Bluetooth interface. Partially through the design process, the two teams met to decide on a protocol for transferring data over Bluetooth so that the integration would be seamless. We then went on to design the rest of the project with this protocol in mind. The first time we attempted to integrate the app with the SpinBit we were able to successfully transfer data, save it to the app and display a graph of that data. When gathering this data we periodically spun slowly to fast so that the graph of the data would be recognizable. The graph for this data demonstrated this expectation.

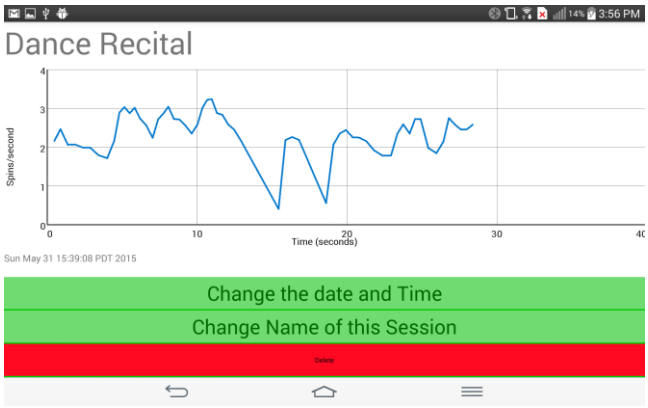

Figure 3. The graph of a dance session displayed on the mobile device; the peaks and valleys correspond to when the tester was spinning quickly and slowly.

5. CHALLENGES

One of the main challenges that we faced was programming the microcontroller on our PCB. When designing the PCB we believed we would be able to program the processor with FTDI pins that were intentionally exposed and soldered to pin headers. It turned out that this was not sufficient for flashing the bootloader onto the microcontroller chip that we are using. In the end we soldered wires directly to the ATMEGA in order to program it. This was a slow process and it frequently required reattaching and often re-soldering wires to the processor just to load a new program onto it; but this did in fact work. We still don't know how to properly program with the FTDI, and so have been programming with an Arduino Uno as ISP. This means that we are programming at 5V which was not our intent when designing the device. This resulted in various components on the PCB being programmed at a higher voltage than they were specified to be. So while programming we fried our Bluetooth module. Our solution to this was to solder on the 5V Bluetooth module that we used for our breadboard prototype and run it at 3.3V. This seems to be working fine so far.

6. FUTURE WORK AND DISCUSSION

In the future we would like to make the device much smaller. We think that this could be done through changing some of the PCB components. The first change that we would make is to use an SPI memory chip instead of the microSD card. The IC for the SPI memory will take up much less space than current microSD card reader. The only downside to using the SPI memory is that we would need to write our own code for memory management instead of being able to use the Arduino SD library.

The second change that we would make is to the Bluetooth module. Instead of the current breakout board that we are using, we would ideally use a low power Bluetooth module. It would also be an improvement if we got a stacked chip such as the RFduino where the Bluetooth module is stacked directly on top of the microcontroller. If we do use a Bluetooth module that is lower power, then our device as a whole would be using less power. Switching from microSD to SPI chip should also reduce the power consumption of parts on the board. With both of these power reductions it could be possible to use a much smaller battery. With all of these size reductions in parts the device as a whole could be much smaller and less noticeable when clipped onto the waistband of the user.

While developing this device there was a lot of discussion as to what should count as a "spin". Potential users of the SpinBit whom we spoke with all had very different opinions as to what they believe counts as a "spin". We would like to add multiple spin options that the user can select from. For instance the user could want a spin to be counted only when they have rotated a full 360 degrees, or they may think that moving 180 degrees and then changing directions should count as a spin. There are several ways this could be implemented. One way would be for the user to send a signal from the phone app to tell the microcontroller on the device to process data differently. If we were to implement it this way it would not be possible to change how spins were counted after the data had already been collected. The second way that we could implement this feature would be to process data on the phone side of the system. If we were to implement the feature this way we would have to send all of the data collected by the gyroscope during the spin session over Bluetooth, which could potentially be a lot of data if the session lasts for several hours. The benefit of this would be that the user could go back to logged spin sessions stored on their phone and change the way that spins were processed.

7. CONCLUSION

This project was developing the SpinBit, a device for detecting, logging and quantifying the spinning motion of the user. The device is lightweight and can be used completely independently of any other device. It uses real time gyroscope data processing to detect and record spins locally and uses Bluetooth to transfer this data to a mobile device for viewing and sharing. We envision this device to be used by dancers and performance artists to improve their work and satisfy their competitive spirit.

8. ACKNOWLEDGMENTS

Our thanks to the CSE and EE departments. Especially to Professor Reynolds and all the people in his lab who helped us when we screwed stuff up.

9. REFERENCES

[1] Theportlandcollection.com, 'Contra Dancing', 2015. [Online]. Available: https://www.theportlandcollection.com/contra-dancing

[2] Electroiq.com, 'Introduction to MEMS gyroscopes', 2015. [Online]. Available: http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/

Columns on Last Page Should Be Made Equal Length