Remote-Controlled DJ

By

Jie Du

Ningkai Wu

Yifei Teng

Final Report for ECE 445, Senior Design, Spring 2018

TA: Jacob Bryan

2nd May 2018

Project No.12

Abstract

The purpose of the project is to design and build a device to control music remotely with hand gestures. It offers a portable and cheap alternative to a classical DJ station. In short, Remote-Controlled DJ is a portable DJ station that hooked up to the user's wrist. By using various sensor data, the device can detect user's gesture, manipulate playlist and produce sound effects accordingly.

Keywords: DJ Station, remotely, portable, cheap.

Contents

Introduction 1
1.1 Background and Purpose 1
1.2 Gesture Logic 1
1.3 Subsystem 1
1.4 Functionalities 3
Design 3
2.1 Gesture Detection Design 4
2.1.1 Kalman Filtering 4
2.1.2 Gesture Detection 5
2.2 Physical Design 5
2.3 Schematics 7
2.4 Design Alternatives 9
2.5 Risk Analysis 10
3. Design Verification 11
3.1 Orientation Accuracy 11
3.2 One-Shot Gestures 13
3.3 Barometer 14
4. Costs 15
4.1 Parts 15
4.2 Labor Cost 16
4.3 Schedule 16
5. Conclusion 17
5.1 Accomplishments 17
5.2 Uncertainties 17
5.3 Ethical Considerations 18
5.4 Future Work 18
Citations 19
Appendix A. Requirement and Verification Table 20

1. Introduction

1.1 Background and Purpose

In a party, the DJ is responsible for supplying everyone an endless stream of entertaining music. But we all know he deeply wants to join the party! So we will build a remote gesture controlled DJ console that every DJ can take into the action. Due to the incorporation of physical turntables and hardware implementation of signal processing functions, DJ stations often cost upwards of $1000 dollars [7], and are very heavy and cumbersome to transport. We want to miniaturize the DJ for the average user, and at the same time add creative ways to interact with the sound. The primary uses of DJ has been to stream through prepared playlists. Therefore having a phone app provides the reasonable convenience of accessing the user's existing playlists. Through the gesture control offered by our wrist-strapped device, we achieve multiple dimensions of interaction that will be more intuitive and direct than the myriad of buttons and sliders on a DJ station.

1.2 Gesture Logic

In order for readers to have a better understand how the device works, we have all the detailed gesture commands here. We have three different modes in total which map to different sound effects. The modes can be switched via buttons on our device. Also, we have some general gestures that can be used in all modes such as navigating the playlist.

All modes: Turning of wrist maps to texture change. (Like turning the texture knob on a guitar amp). Slicing your hand horizontally outwards through air moves to the next song. Slicing inwards goes to the previous song.

Mode 1: Height maps to pitch. Up/down rotation of palm maps to reverb.

Mode 2: Sudden upwards rotation of palm maps to activating looping, sudden downwards rotation maps to stopping the looping.

Mode 3: Height maps to wah-wah effect.

1.3 Subsystem

Our system comprises two parts:

1. **A compact device** that straps to one's wrist and collects gesture information. The gesture can be used to navigate a playlist, change various effects, manipulate voice recorded from a microphone etc. It will also sample the onboard inertial measurement units at a rate of 500Hz, estimate orientation from the measurements, and send over the processed result at a lower rate of 20Hz.

2. **A phone app** that implements the various signal processing functions and outputs the music. The app is driven by gesture data from the embedded device.

Specifically, the device is composed of a power module, a control unit and a bluetooth module as shown below by the block diagram. The power module is used to charge our device. At the meantime, it also includes a charging circuit that is used to charge the Li-Po Battery. The control unit collects all sensor data, for example, the data from accelerometers, gyroscopes, magnetometers and barometer. The microcontroller mounted on the device fuses sensor data (using Kalman filters) to estimate the pose of the hand, in the form of orientation and change in height. We defined a custom protocol to stream these events along with continuously changing gesture data to the phone, which will make use of these data to perform signal processing tasks. A flowchart with various signal processing steps on our phone app is also provided.

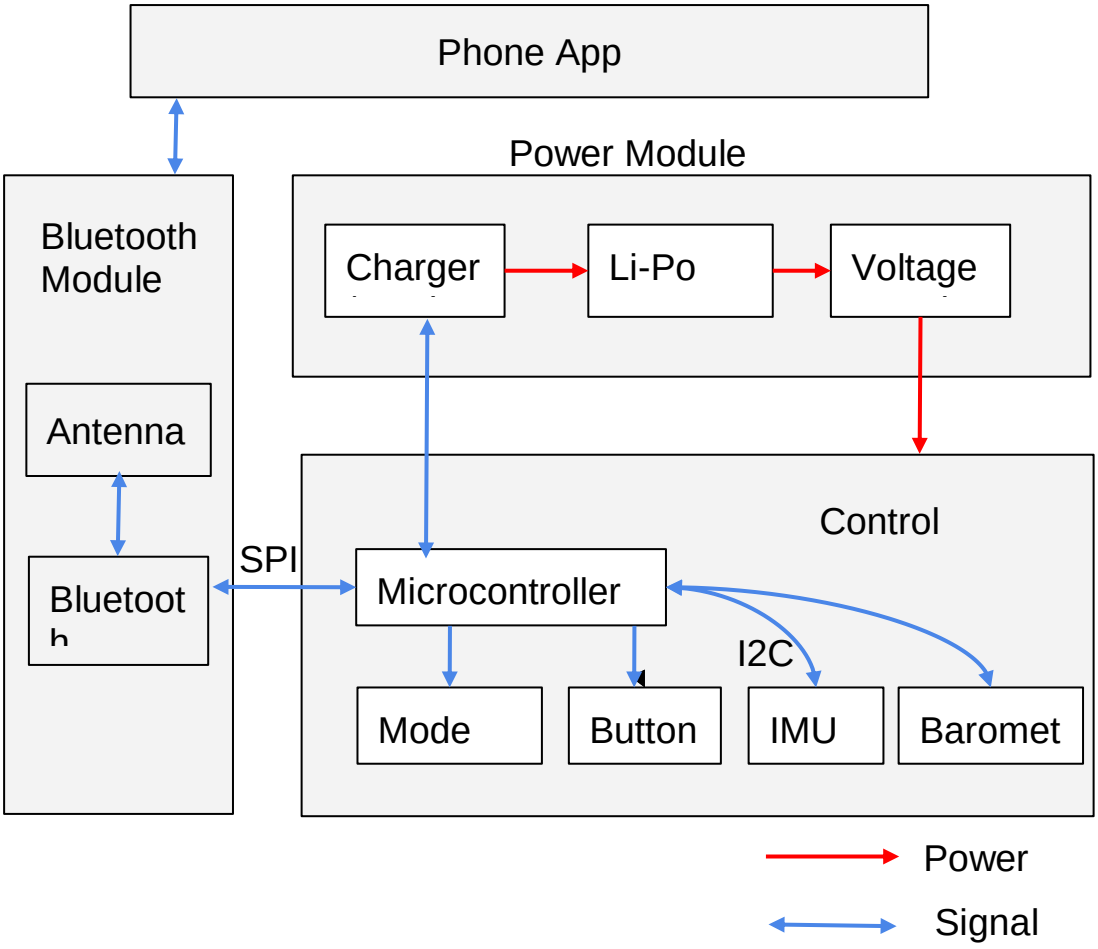

Fig 1. Block Diagram

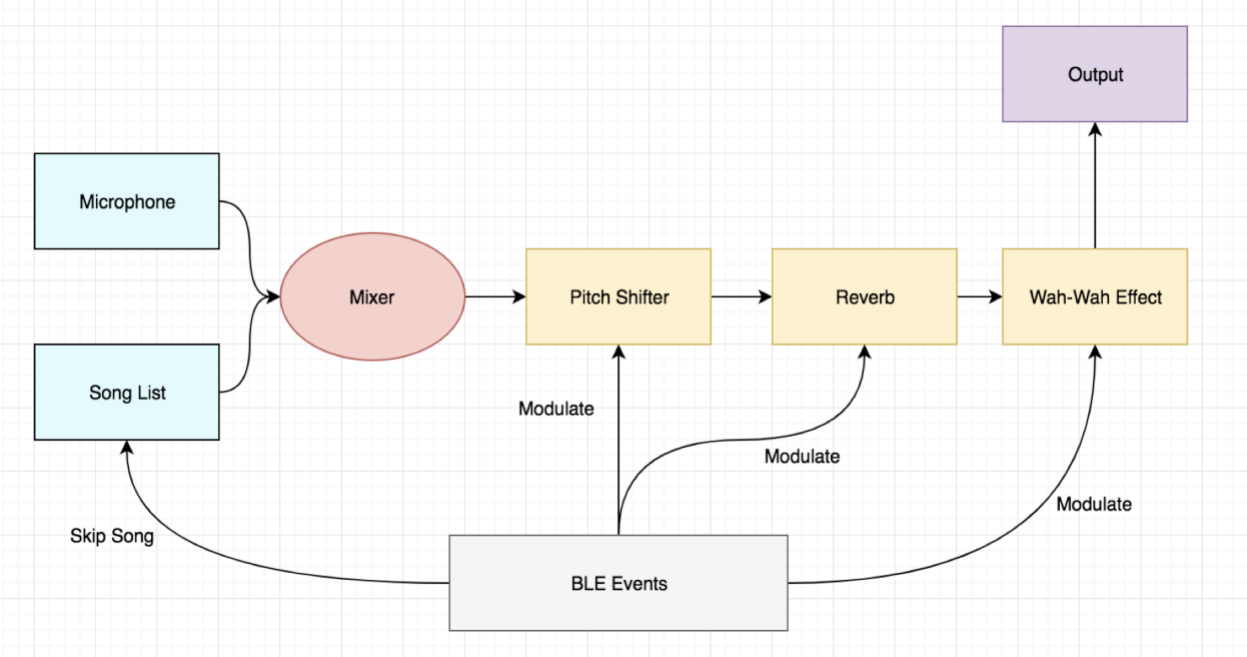

Fig 2. Audio Processing Architecture

1.4 Functionalities

- Requirement 1: The device will detect the orientation of the hand with an accuracy of +/- 10 degrees.
- Requirement 2: The device will detect “one-shot” gesture events e.g. skipping songs by horizontally throwing the wrist outwards with a success rate greater than 90% and less than two false positives in ten minutes.
- Requirement 3: The device will measure the relative change in height with an accuracy of +/- 20 centimeters.

In order for our device to detect gestures correctly, the first thing we need to ensure is that the orientation from our IMU is correct. For example, the orientation in roll, pitch and yaw angles are within the +/-10 degrees bound. Secondly, we need to map these orientations and other acceleration values to our gestures correctly. We want to know the approximate range of the orientation and acceleration we need for specific gestures. That’s how we come up with requirement 2. Thirdly, we need the relative height to be within +/- 20 centimeters bound in order for some gestures related to height to work properly.

2. Design

A modular breakdown explanation is provided for each of our components.

2.1 Gesture Detection Design

Gesture detection is performed in two stages. First the raw values of accelerometers, gyroscopes, and magnetometers are fused to determine an estimation of an orientation quaternion. Then, both the sensor raw values and the change in orientation angles are used to identify a specific gesture.

2.1.1 Kalman Filtering

For sensor fusion, we implemented an adaptive extended Kalman filter outlined in [12]. The algorithm iteratively estimates the error in each of the sensors, and runs 500 times per second. The algorithm iteration can be summarized as follows:

1. We start with a unit quaternion representation of the current orientation:

$$\hat{q}_{est,t0} = |q_1\ q_2\ q_3\ q_4|$$

2. Given normalized magnetometer reading and the quaternion, compute the reference direction of Earth's magnetic field, and use it to estimate the magnetic field

$${}^{E}\hat{\boldsymbol{h}}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^{S}\hat{\boldsymbol{m}}_t \otimes {}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{est,t-1}$$

$${}^{E}\hat{\boldsymbol{b}}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix}$$

$$\frac{1}{2}\vec{w} = b_x \begin{pmatrix} 0.5 - q_3q_3 + q_4q_4 \\ q_2q_3 - q_1q_4 \\ q_1q_3 + q_2q_4 \end{pmatrix} + b_z \begin{pmatrix} q_2q_4 - q_1q_3 \\ q_1q_2 + q_3q_4 \\ 0.5 - q_2q_2 + q_3q_3 \end{pmatrix}$$

3. Compute the error in magnetometer reading

$$E_h = \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix} \otimes \vec{w}$$

4. Estimate direction of gravity

$$\frac{1}{2}\vec{v} = \begin{pmatrix} q_2q_4 - q_1q_3 \\ q_1q_2 + q_3q_4 \\ -0.5 + q_1q_1 + q_4q_4 \end{pmatrix}$$

5. Accumulate errors in accelerometer reading

$$E = E_h + \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \otimes \vec{v}$$

6. Update quaternion using compensated gyroscope readings

$$Integral\ Feedback_t = Integral\ Feedback_{t-1} + K_i\ E\ \Delta t$$

$$\tilde{g} = g + K_p\, E + Integral\ Feedback_t$$

$$\Delta g = \tilde{g}\, \Delta t$$

$$\hat{q}_{est,t} = \begin{vmatrix} -q_2\, \Delta g_x - q_3 \Delta g_y - q_4 \Delta g_z \\ q_1 \Delta g_x + q_3 \Delta g_z - q_4 \Delta g_y \\ q_1 \Delta g_y - q_2 \Delta g_z + q_4 \Delta g_x \\ q_1 \Delta g_z + q_2 \Delta g_y - q_3 \Delta g_x \end{vmatrix}$$

7. And finally just normalize the new estimated quaternion.

2.1.2 Gesture Detection

For gesture detection, we are basing our detection on the yaw/pitch/roll angles converted from the quaternions, in addition to low-pass filtered accelerometer and gyroscope readings. Our detection logic is run every 20 milliseconds. To register a “next song” event, we require the following conditions to be met:

1. Change in yaw is lower than -60 degrees compared to 10 runs ago.
2. Z-axis angular velocity is lower than -200 at least 2 times out of the last 10 checks.
3. None of the 10 checks indicates Z-axis angular velocity is greater than 20.
4. Absolute values of both pitch and roll are less than 30 degrees throughout.

We considered more advanced detection mechanisms such as linear regression and Bayesian inference before deciding that simple conditions will suffice.

2.2 Physical Design

The device should comfortably strap onto the user’s wrist. We will design a casing for our electronics that partially matches the shape of a mobile phone, so it fits in one of the off-the-shelf exercise phone holder wristbands (Fig. 3). The design consists of two halves, each of which will be 3D printed and pieced together to form the casing.

Referring to Fig. 3, we plan to place the battery in between the two M3 nut holders, glued to the bottom part of the casing. The PCB will be mounted on top of the M3 holders using screws. Holes will be carved out of the front panel as needed to expose buttons and LEDs.

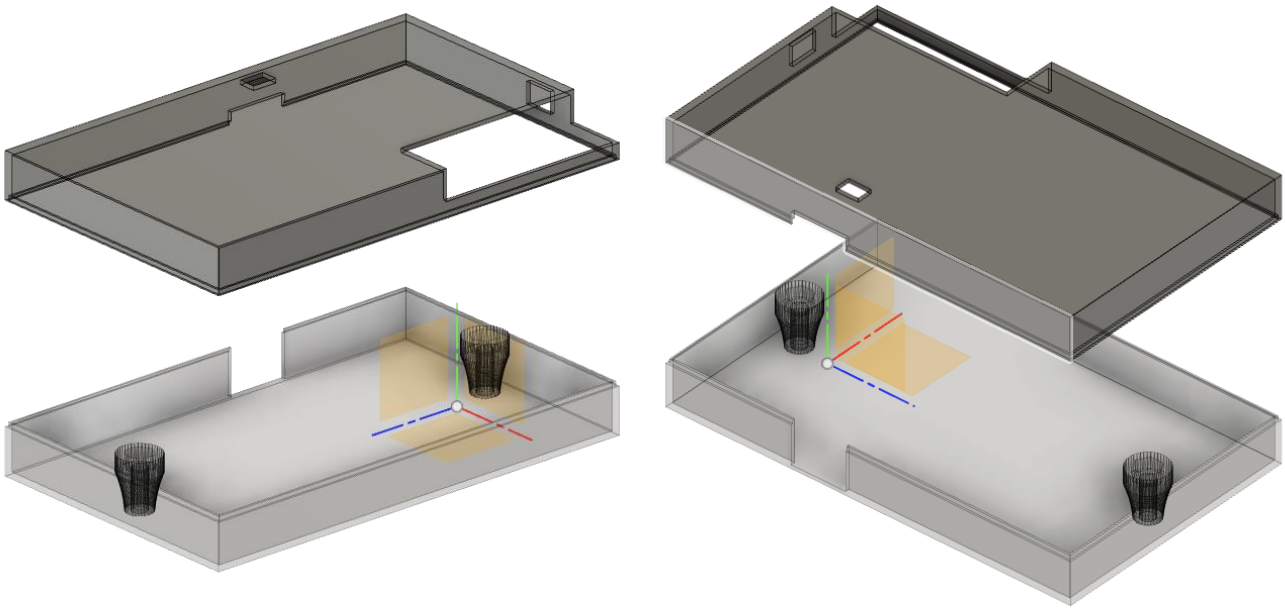

Fig 3. 3D External View of Physical Design, Two Perspectives

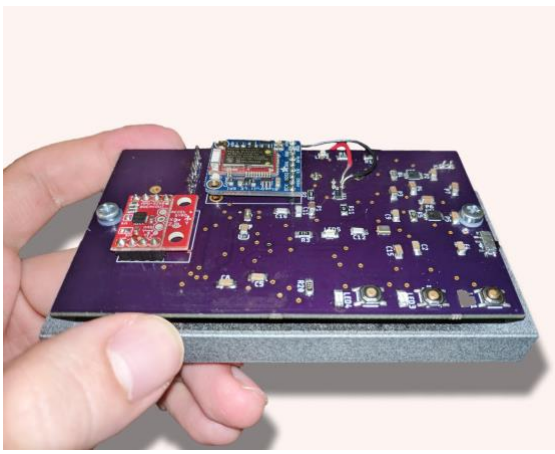

Figure 4.Front View of PCB

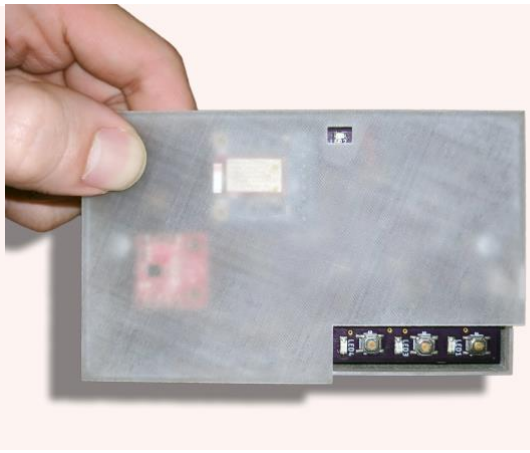

Figure 5. Front View of Case with PCB fixed inside

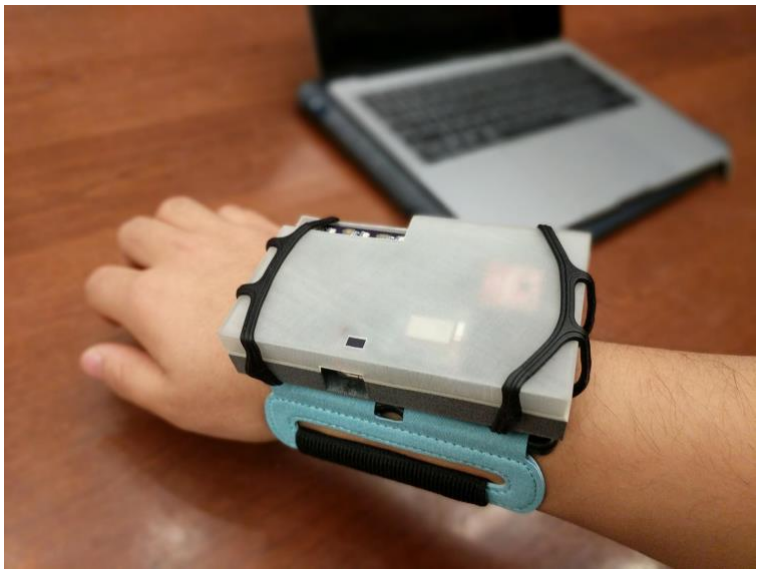

Figure 6. Final product with wrist band

2.3 Schematics

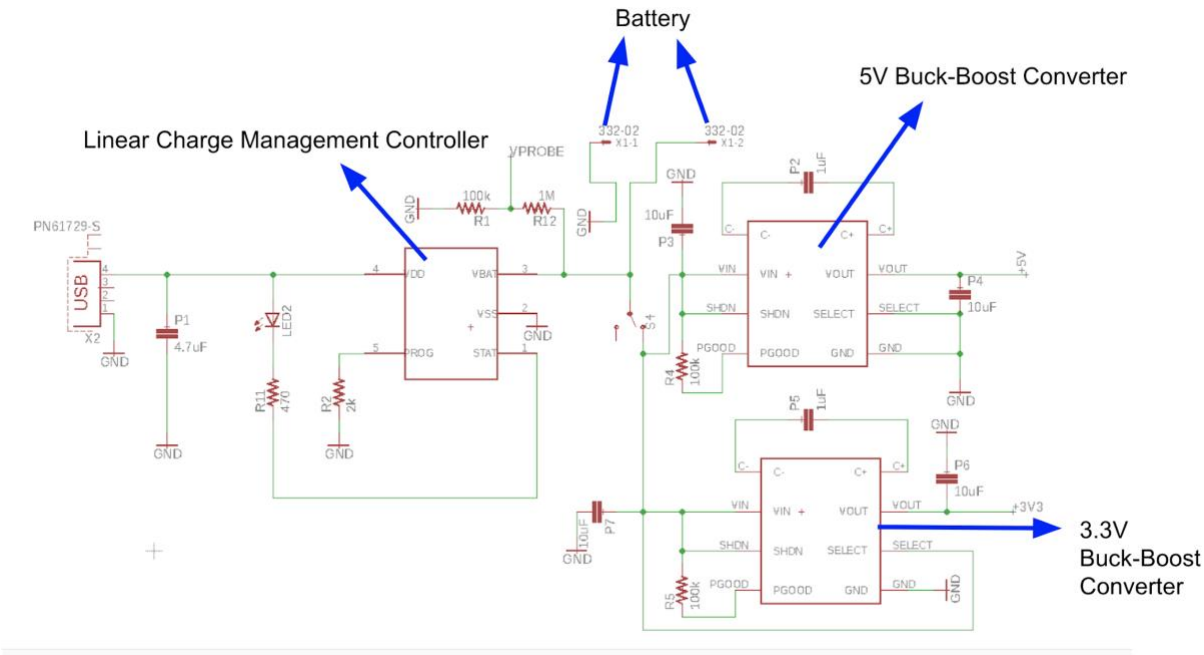

Fig 7: Power Module

Power module includes battery charger and monitor, Li-Po battery, and two voltage regulators. Li-Po battery will be used as a power source to keep the communication network up continually, and it is boosted to 5V by a boost converter to power barometer and bluetooth. And the regulated voltage 3.3V will be used to power microcontroller, IMU, and other components in control unit. We are using MCP1252 for 3.3V and 5V converter since it has high efficiency and

deliver up to 120mA output current, which is more than we our device needs. We can charge Li-Po battery from a USB port(5V) and monitor the voltage of battery continuously through a voltage divider circuit while charging and discharging. A linear charge management controller is used to supply constant current to charge the battery. We also have a switch that allows user to switch between charging mode and operating mode.

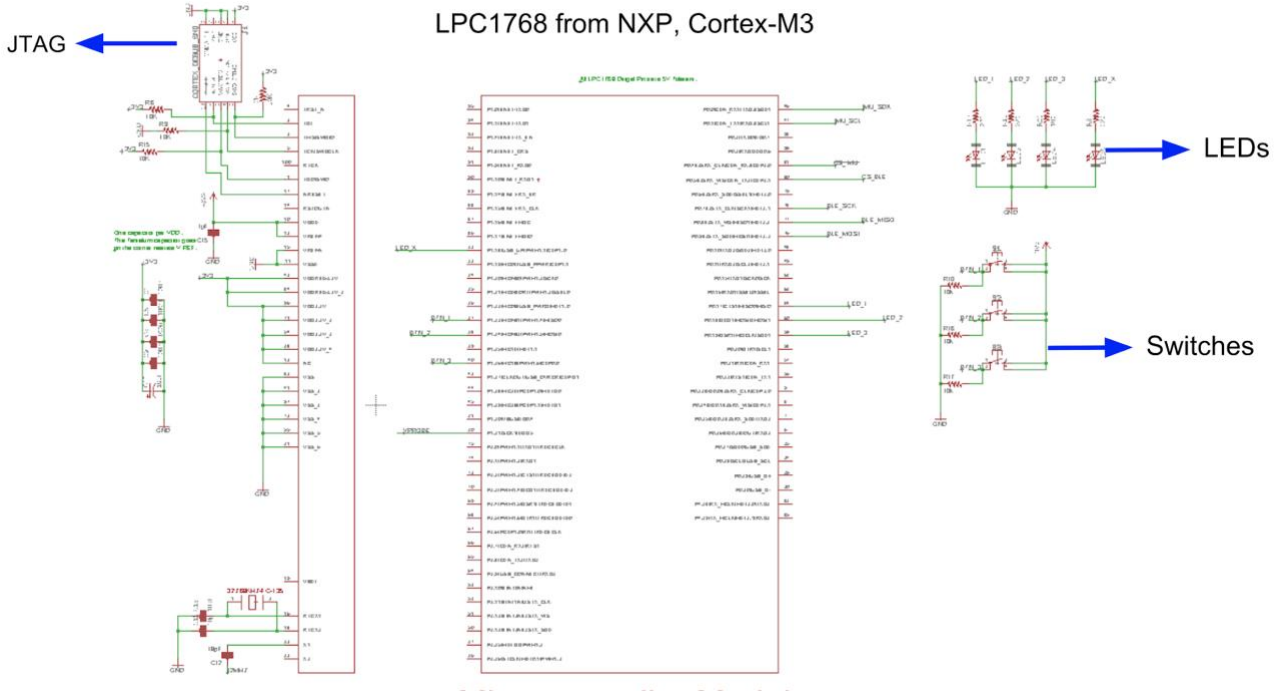

Fig 8: Microcontroller Module

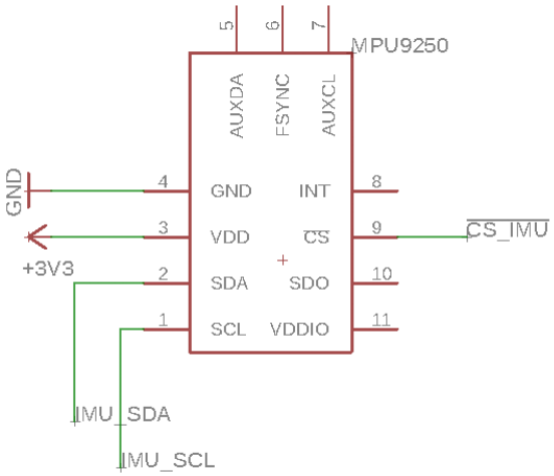

Fig 9: Inertial Measurement Unit

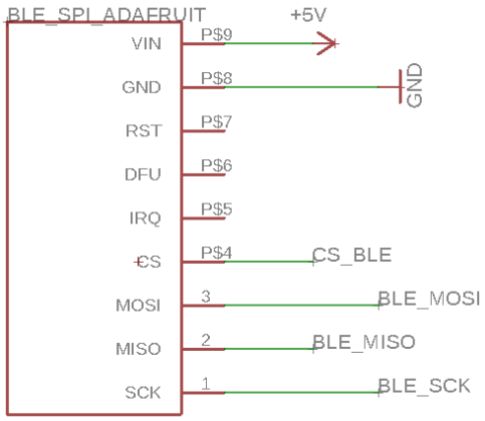

Fig 10: Bluetooth Unit

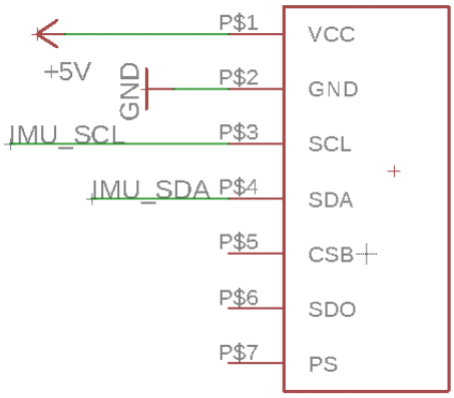

Fig11: Barometer Unit

2.4 Design Alternatives

There are a number of alternatives we pursued under the overall goal of expressive music control, and their respective trade-offs are listed here:

Gesture-detection glove: This will provide unmatched gesture detection results by attaching flex/position sensors to each finger and calculating the finger movements. However, this area has been explored by previous projects extensively, and we felt that the extra innovation contribution here will be low. Furthermore, having your hand limited to a few positions can be unnatural for prolonged use, and the tendency of false positive is likely much higher than a wrist-strapped solution where it's much easier for the wearer to ignore the presence of the device.

Microphone to detect finger snapping for mode switching: Our design used three buttons to indicate the current mode of control. An alternative discussed is using a microphone to analyze the sounds coming from the hand, and detect the particular sound of the user snapping fingers, thus obviating the need for buttons and a second hand. After some evaluation we decided that it

will be very hard to do in a noisy environment. Successful implementation of this feature will also depend largely on the type of microphone, an area we are not familiar with.

Audio processing on the microcontroller: An alternative to offloading the audio processing to the phone would be to do everything on the microcontroller, just like an integrated DJ station. Our choice of microcontroller should be powerful enough for the extra processing. But in practice, most users would find the iPhone solution much more practical and easy to get started with. Doing audio processing on the phone also opens up opportunities to connect with music streaming services such as Spotify and SoundCloud, providing a huge selection of songs instantly.

2.5 Risk Analysis

Our project integrates both IMU and barometer/altimeter sensor values to determine the movement of one hand. Therefore, the greatest risk in our project is whether our sensors are able to provide precise values, e.g. hand orientation and its change in height. At the meantime, these sensors need to respond fast enough if user moves hand fast. If the movement is determined correctly, then the rest of design is straightforward.

2.6 Tolerance Analysis

There are many factors which affect the operation of our system. The most critical one is the accuracy of sensor fusion algorithm responsible for calculating the orientation of the arm. We will analyze backwards from the design requirement to obtain a first order estimate that can serve as minimum tolerances for the sensors. Referring to high level requirement 3, we need to estimate the absolute height of the device using a noisy barometer and a noisy accelerometer. By investigating the Kalman filtering behavior, we will work out the desired accelerometer and barometer accuracy. The state of the system can be characterized as:

$$\hat{s} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

Where $\hat{S}$is the estimate of state and$x$is the vertical location.
Given

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

$$B = \begin{pmatrix} \frac{1}{2}(\Delta t)^2 \\ \Delta t \end{pmatrix}$$

The state-space system mode is given by

$$s_k = \begin{pmatrix} x_k \\ \dot{x}_k \end{pmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} s_{k-1} + \begin{pmatrix} \frac{1}{2}(\Delta t)^2 \\ \Delta t \end{pmatrix} \ddot{m}$$

$$s_k = A s_{k-1} + B\ddot{m}$$

Where $\ddot{m}$ is the accelerometer reading.
The state estimate error covariance is

$$P_k = AP_{k-1}A^T + Q_k$$

Where Q_k is the process noise covariance matrix, modelling any external noise in the measurements.

Assuming the accelerometer reading is perturbed by some additive gaussian noise w_k such that $\ddot{m}_k = a_k + w_k$, and the barometer reading is perturbed by some additive gaussian noise n_k such that $m_k = h_k + n_k$, it can be shown that

$$Q_k = BB^T {\sigma_{a,k}}^2 + \frac{{\sigma_{x,k}}^2}{2}$$

Where $\sigma_{a,k}^2$ is the variance in the accelerometer reading, and $\sigma_{x,k}^2$ is the variance in the barometer reading.
We would like to know the variance in height, assuming the system is able to converge to a steady state, and that would be

$${\sigma_{\hat{x}}}^2 = \frac{\Delta t^4 {\sigma_{a,k}}^2}{4} + \frac{{\sigma_{x,k}}^2}{2}$$

Assume we want the height estimate error to be less than 8 cm with 99% certainty, that requires ${\sigma_{\hat{x}}}^2$ to be less than $(8/2.4)^2$= 11.1. Under a short enough sample interval, we observe that the primary contributing factor to the variance is the barometer measurement, and it needs to be less than 22.2. The barometer needs to have a reported accuracy of +/-11 cm, which is above our design accuracy of +/-20 cm.

3. Design Verification

In order to satisfy our three high-level requirements and let our device work properly, we did following verifications.

3.1 Orientation Accuracy

Requirement : The orientation estimation algorithm must be stable within 10 degrees after randomly rotating for 10 seconds.

In order to test this requirement, we connect only the IMU and the microcontroller on a breadboard, and configured the sensor fusion code to print out the estimated yaw, pitch, and roll values. The breadboard is rotated by roughly 45 degrees at 10 second intervals, and repeated for all three axes. We collected print out and plotted the orientation data, and verified that the output indeed quickly converged to our expectation. See below for some plots of the sensor fusion outputs. Each plot contains multiple rotations.

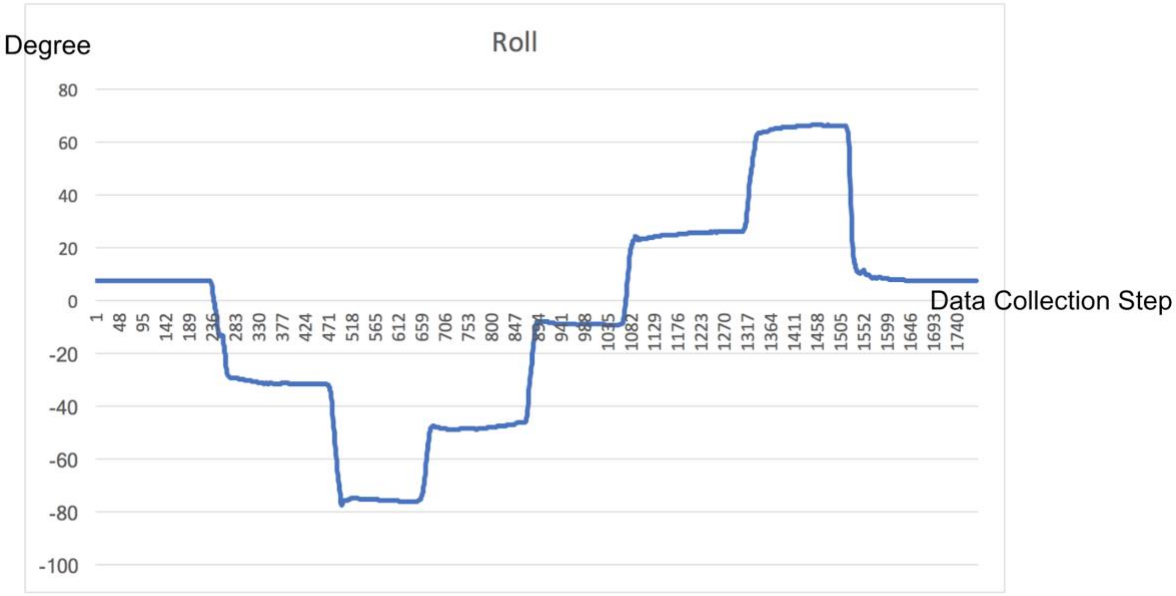

Fig 12. Roll against time-steps as measured from holding the breadboard at different angles

The x-axis is a time-step in the logging mechanism, and corresponds to roughly 30 milliseconds of wall time. Each sharp change indicates a change in orientation of the board. As can be seen in the figure above, the algorithm converges quickly in the roll axis. The initial segment and the final segment are completely flat, since that is the result of placing the breadboard on the table. The middle portions had slight variations which are only due to the instability of the hand. The breadboard had metallic legs on one side which resulted the non-zero roll on the table.

Fig. 13 is the plot of pitch changes:

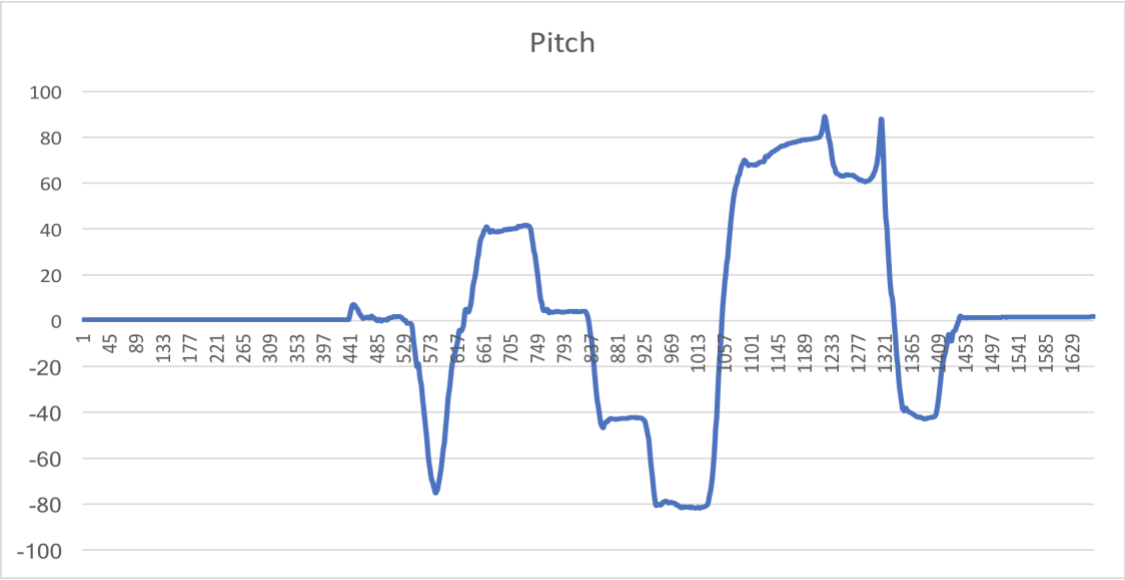

Fig 13. Pitch against time-steps as measured from holding the breadboard at different angles

The valid portions are from time-step 661 to time-step 1057, plus the starting and ending flat portions. The pitch axis is also stable, quickly converging to the expected value under far less than 5 seconds.

Note that there is a sudden drop from close to 90 degrees to 60 degrees at around time-step 1210. This is due to the estimated angle overshooting 90 degrees and wrapping around it. We will need to account for these kind of sudden changes in the gesture detection code in the future. Fig. 14 is the plot of yaw changes:

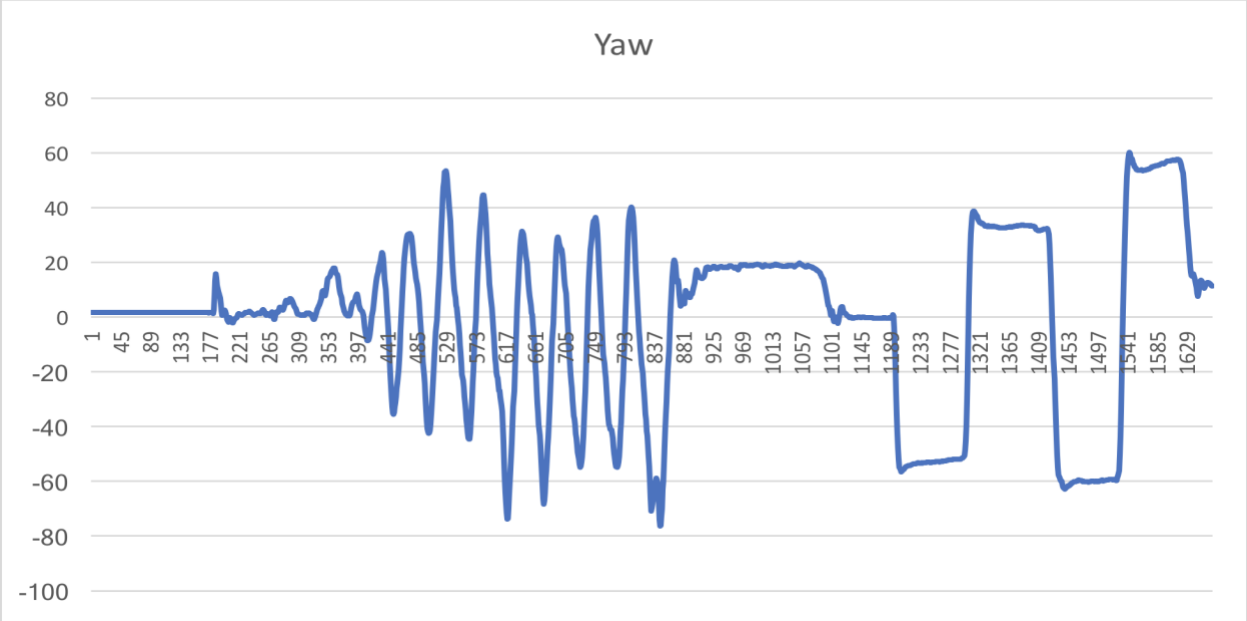

Fig 14. Yaw against time-steps as measured from holding the breadboard at different angles

3.2 One-Shot Gestures

Requirement : The device will detect “one-shot” gesture events e.g. skipping songs by horizontally throwing the wrist outwards with a success rate greater than 90% and less than two false positives in ten minutes.

In order to satisfy this requirement, we perform one-shot reliability tests for 50 times in 183 second, and we have results as following:

Table 1. One-shot reliability tests

Test / Duration	No. Identified	No. Failed	No. False Positive
50 / 183 seconds	46	4	0

Success Rate = 46/50 = 92%

Our test results have success rate greater than 90% and 0 false positive, so it satisfies our requirement.

3.3 Barometer

Requirement: The device will measure the relative change in height with an accuracy of +/- 20 centimeters.

Verification:

1. Set up barometer and microcontroller on a breadboard. Continuously log the height values to disk via UART.
2. Fix the breadboard on ground and stabilize for thirty seconds.
3. Move the breadboard 2.0 meters above for another thirty seconds.
4. Compare the difference in height values by inspecting log. If the difference is within 0.2m then the requirement is met.
5. Repeat 1 to 4 for head to ground.

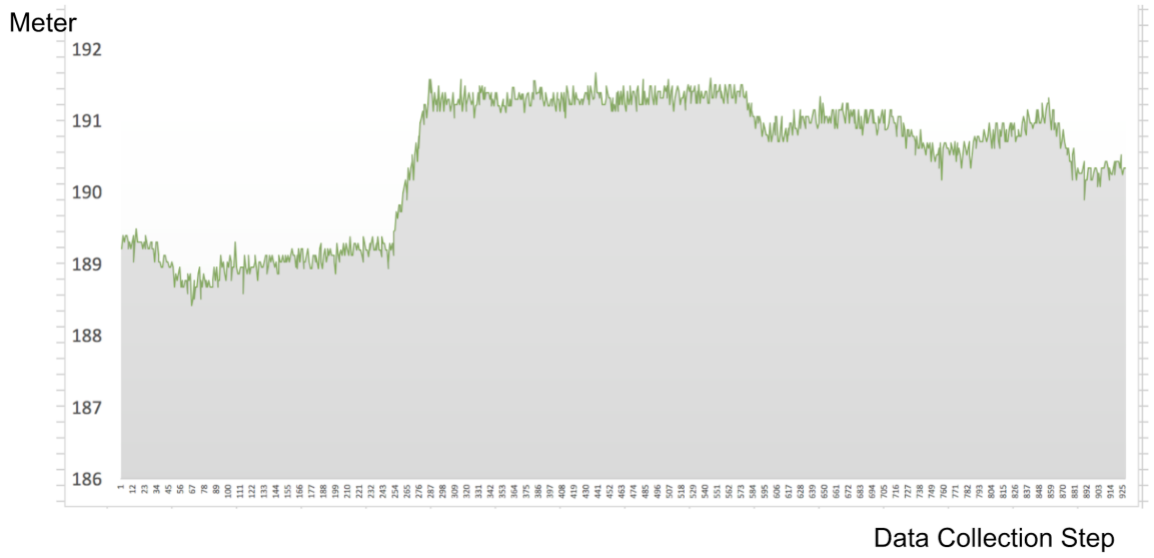

Fig 15. Absolute height against time-steps

Originally, we have the height of our barometer stabilized around 189m starting from time-step 50 to time-step 250. After time-step 250, we can observe the sharp curve indicating the upward movement. Starting from time-step 300, the height again stabilized around 191m. Therefore, the +/- 20 centimeters requirement is satisfied.

4. Costs

4.1 Parts

Table 2 : Parts List

#	Description	Part Number	Manufacturer	Quantity	Cost($)
1	Bluefruit LE SPI Friend - Bluetooth Low Energy (BLE)	2633	Adafruit	1	17.5
2	3.3V Buck-Boost Converter	MCP1252-33X50I/MS	Mouser	2	3.3
3	Lithium Ion Polymer Battery - 3.7v 500mAh	LP503035	Adafruit	3	7.95
4	5.0V Buck-Boost Converter	MCP1252-33X50I/MS	Mouser	2	3.3
5	Battery Management	MCP73831T-2ATI/OT	Mouser	5	3.4
6	Microcontroller	NXP1768	Sparkfun	1	54.95
7	IMU	MPU9250	Sparkfun	1	14.95
8	Altimeter Sensor	MS5611	Amazon	1	13.99
9	Mini Pushbutton Switch	COM-08720	SparkFun	6	5.7
10	SMD Right Angle Switch	COM-10860	SparkFun	3	2.85
11	JST Right-Angle Connector	PRT-09749	SparkFun	2	1.9
12	Yellow LED	PRT-12622	SparkFun	25	4.95
13	Iphone Wristband	None	Amazon	1	14.99
14	First 3D Printed Case	None	Shapeways	1	16.07
15	Second 3D Printed Case	None	Shapeways	1	75.62
16	First Iteration PCB	None	PCBWay	10	(through class)
17	Second Iteration PCB	None	OSH Park	3	95.1

18	SMD Resistors, Capacitors, and Inductors	None	Digikey	185	42.52
Total Cost: $379.04					

4.2 Labor Cost

Assume the labor cost is 45$/hr for each person, and total hours for 3 people to complete the project is calculated below:

Table 3. Labor Cost

Circuit/ PCB Design	Circuit Test	PCB soldering	Data Fusion code	Phone App Design	Integration	Debug	Total Hour
60hr	25hr	28hr	55hr	55hr	45hr	40hr	308hr

Therefore, Total Cost = 308hr * 45$/hr =13860$
GRAND TOTAL = LABOR + PARTS = $13860 + $379.04 = $14239.04

4.3 Schedule

Table 4. Schedule

Week	Jie	Ningkai	Yifei
2/19	1: Order parts for power module 2: Work on schematics and calculations	1: Order parts for bluetooth and microcontroller module 2. Microcontroller schematics	1: Design Physical Housing
2/26	1: Design PCB layout, 2: Doing power module circuit tests and modify the schematics if needed	1: Unit testing barometer 2: Write code for bluetooth module on App end	1: Prototype iOS App 2: Unit testing inertial measurement devices
3/5	1: Order first PCB 2: Help debugging bluetooth code	1: Write code for bluetooth module on Microcontroller end	1: Validate sensor fusion design on breadboard
3/12	1: PCB soldering 2: Reorder parts that are needed 3: Test our circuits	1: Unit testing bluetooth module	1: Validate iPhone integration on breadboard 2: Refine physical design and order second iteration
3/19	Spring Break	Spring Break	Spring Break

3/26	1: Make some modifications on first PCB layout and order second PCB iteration	1: PCB soldering 2: Make modifications on schematics in needed	1: Integration testing on PCB
4/2	1: Work on gesture detection Mode 1 code	1: Work on gesture detection Mode 2 code 2: Unit test LED and button	1: Work on gesture detection Mode 3 code
4/9	1: Second round soldering	1: Combine three modes with other functions	1: Fine-tune gesture detection
4/16	1: Final testing and debugging 2: Adding more functions of our device if we have time	1: Unit test microcontroller	1: Finalize product
4/23	1: Prepare for final presentation slides	1: Rehearse final demo sequences	1: Rehearse final demo sequences
4/30	1: Work on final report	1: Polish final presentation slides 2: Work on final report	1: Work on final report 2: Last-minute iPhone app debugging

5. Conclusion

5.1 Accomplishments

There are two aspects to our biggest accomplishments. Our technical accomplishments are centered around integration and algorithms. We started out with the goal of building a great product, a remote DJ controller that is easy and intuitive to use. With that in mind, the entire design must be streamlined, and just work with the press of a button. We managed to hide most of the complexities about orientation calculation and gesture detection. The microcontroller calibrates all the sensors automatically upon startup, and the phone app automatically searches and connects to the microcontroller before streaming the orientation and gesture events in the background. In terms of the experience our design is much closer to a finished product. The other accomplishment is that we implemented and fine-tuned the Kalman filter and the gesture detection code to reasonable accuracy. It passes our own requirements, and is entertaining not annoying to control when hooked up to the audio processing pipeline.

5.2 Uncertainties

The biggest source of uncertainties come from hardware and PCB. We undertook significant risk when we decided to put a powerful microcontroller with complicated supporting circuitry on the PCB. We need to design our power supplies from scratch and layout the PCB with space constraints. Our first choice of 5V regulator did not solder well as the contacts are hidden under the chip, and our first choice of 3.3V regulator had an unexpectedly large dropout voltage of 1V when tested such that even under a 4.2V Lipo supply, the output voltage was only 3.2V. So we

had to change our PCB design last minute to use two MSOP-package SMD regulator chips (MCP1252) [11]. We soldered three PCBs in total: the first one was scrapped because of power unit insufficiencies; the second one was scrapped because the JTAG header unexpectedly peeled off during testing taking that corresponding area of PCB with it. Needless to say, these resulted significant surge in budget and extended hours, which we could foresee and manage better in the future.

5.3 Ethical Considerations

We did a lot of circuit and performance tests during the project, and we are honest with our test results. When we have incorrect test results, we wrote notes and did tests again. Based on #3 of IEEE Code of Ethics,'' to be honest and realistic in stating claims or estimates based on available data''[1]. We didn't use unreal data to make our project seem working well.

We did many research on the project, and there are some brilliant ideas from a paper and websites. We respect ideas of others and cite them in our project report. Based on #1.6 of AMC Code of Ethics, it is very important to" Give proper credit for intellectual property" [2].

5.4 Future Work

In the future, we can redesign our PCB to eliminate some air wires. We can also make our device smaller by choosing smaller chips and rearrange PCB layout. More music effects could be added to make our device more interesting.

Citations

[1] Ieee.org. (2018). *IEEE IEEE Code of Ethics*. [online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 9 Feb. 2018].
[2] Acm.org. (2018). *ACM Code of Ethics and Professional Conduct*. [online] Available at: https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct [Accessed 9 Feb. 2018].
[3] Manager, S., 1Ah, L., 110mAh, L., 850mAh, L. and 400mAh, P. (2018). *Battery Babysitter Hookup Guide - learn.sparkfun.com*. [online] Learn.sparkfun.com. Available at: https://learn.sparkfun.com/tutorials/battery-babysitter-hookup-guide?_ga=2.93141789.1476657923.1517777758-793387797.1516935255#lipo-fuel-gauge [Accessed 9 Feb. 2018].
[4] MPU-9250, S. (2018). *MPU-9250 Hookup Guide - learn.sparkfun.com*. [online] Learn.sparkfun.com. Available at: https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide?_ga=2.53729448.1893352856.1518141768-793387797.1516935255 [Accessed 9 Feb. 2018].
[5] Sensor, M. and Sensor, B. (2018). *Barometric Pressure Sensor - TE Connectivity | DigiKey*. [online] Digikey.com. Available at: https://www.digikey.com/en/product-highlight/t/te-connectivity-measurement-specialties/ms5611-01ba-barometric-pressure-sensor [Accessed 9 Feb. 2018].
[6] Industries, A. (2018). *Adafruit Bluefruit LE SPI Friend - Bluetooth Low Energy (BLE)*. [online] Adafruit.com. Available at: https://www.adafruit.com/product/2633 [Accessed 9 Feb. 2018].
[7] American Musical. *Pioneer DDJSX2 Professional DJ Controller.* [online] AmericanMusical.com. Available at: https://www.americanmusical.com/Item--i-PIO-DDJSX2?src=Y0802G00SRCHCAPN&adpos=1o1&scid=scplpPIO+DDJSX2&sc_intid=PIO+DDJSX2&gclid=Cj0KCQiAh_DTBRCTARIsABlT9MaC9EYL5T6RwqgdbdXrabY-VJQHX-LJpFrHlEd2KquUM5-4JevxkTUaAjNIEALw_wcB [Retrieved Feb 8, 2018].
[8] SparkFun IMU Breakout - MPU-9250. Available at: https://www.sparkfun.com/products/13762.
[9] Digikey MS5611-01BA Barometric Pressure Sensor. Available at: https://www.digikey.com/en/product-highlight/t/te-connectivity-measurement-specialties/ms5611-01ba-barometric-pressure-sensor. [Retrieved April 30, 2018]
[10] Adafruit Bluefruit LE SPI Friend - Bluetooth Low Energy (BLE). Available at: https://www.adafruit.com/product/2633 [Retrieved April 30, 2018]
[11] MCP1252 voltage regulator. Available at: http://www.microchip.com/wwwproducts/en/MCP1252 [Retrieved May 2, 2018]
[12] S. O. H. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," Bristol University, UK, 2010.

Appendix A. Requirement and Verification Table

Table 5: System Requirements and Verifications

Requirement	Verification	Status
Power Module: Charger/Monitor		
1: When the battery is fully charged from a USB port(5V), the voltage of battery should be between 4.15V and 4.2V.	1: When the LED of MCP73831 is OFF, measure the voltage of battery by voltmeter and ensure it is within our desired range.	Yes
2: The voltage monitoring should be accurate to within 2%.	2: Compare the voltage of battery and monitored reading from microcontroller, and check whether two values are differed less than 2%.	Yes
Lipo Battery		
1: The battery must be able to store enough charge to provide at 3.3-4.2V for current draw 150mA at least five hours continuously.	1: A: Fully charging the battery. B: Discharging the battery at 150mA for five hours. C. Measure the voltage of battery by voltmeter and make sure it is above 3.3V.	Yes
3.3V Buck Boost Converter		

1: *The voltage regulator must provide 3.3V +/- 5% from a 3.3-4.2V source, under any current load up to 150mA.*	1: A: Connect battery and voltage regulator, and supply 150mA current. B: Measure the output voltage of regulator and make sure percentage difference between measured value and 3.3V is less than 5%.	Yes
2: *Must maintain thermal stability below 125°C at a peak current draw of 150mA.*	2: Measure the temperature of voltage regulator by IR thermometer with 150mA current.	Yes
5V Boost Converter		
1: *The boost converter must provide 5V +/- 5% from a 3.3-4.2V source, under any current load up to 150mA.*	1: A: Connect battery and boost converter, and supply 150mA current. B: Measure the output voltage of regulator and make sure percentage difference between measured value and 3.3V is less than 5%.	Yes
2: *Must maintain thermal stability below 125°C at a peak current draw of 150mA.*	2: Measure the temperature of boost converter by IR thermometer with 150mA current.	Yes
Inertial Measurement Unit		

1: *The device will detect the acceleration of the hand with an accuracy of +/- 0.008 g-rms.*	1. A: Set up IMU and microcontroller on a breadboard. B: Hang the breadboard with a string in a windless environment and hold still. C: Read out the acceleration value from microcontroller directly and make sure the magnitude is within 9.8g +/- 0.008 g rms.	Yes
Altimeter / Barometer		
1: *The device will measure the relative change in height with an accuracy of +/- 20 centimeters.*	1: A: Set up barometer and microcontroller on a breadboard. Continuously log the height values to disk via UART. B: Fix the breadboard on ground and stabilize for thirty seconds. C: Move the breadboard 2.0 meters above for another thirty seconds. D: Compare the difference in height values by inspecting log. If the difference is within 0.2m then the requirement is met. E: Repeat B to D for head to ground.	Yes
Microcontroller		
1: *The orientation estimation algorithm must be accurate within 5 degrees after stabilizing for 10 seconds.*	1: A: Set up IMU and microcontroller on a breadboard. Continuously log the orientation values to disk via UART. B: Fix the breadboard on hand and rotate our hand in roll with a degree of 45. C: Compare the difference in roll values by inspecting log and if they are within 5 degrees then the requirement is met. D: Continue B-D with a rotation in Pitch and Yaw with a degree of 45. E: Try out a degree of -45 in Roll, Pitch, Yaw.	Yes
2: *The drift in orientation must be less than 15 degrees after 5 minutes of active use.*	2: A: Set up IMU and microcontroller on a breadboard. Continuously log the orientation values to disk via UART. B: Fix the breadboard on hand and after stabilizing, randomly rotate hand in Roll, Pitch, Yaw for any degrees for 5 minutes. C: Rotate hand to the original stabilized position. D: Compare the difference in Roll, Pitch, Yaw values by inspecting log and if all of them are within 5 degrees then the requirement is met.	Yes

3. The microcontroller must detect gestures as indicated by a particular pattern in the change in orientation, and report to the phone app with a latency less than 1 second.	3: A: Set up IMU, microcontroller and bluetooth module on a breadboard. B: For “next song” function, fix the breadboard on hand and slice hand horizontally outward. C: Make sure the phone can recognize the gesture in 1 second by outputting the time difference between the time of sliding and the time phone receives it. D: Repeat the same process for all gestures mentioned in mode functions.	Yes
4. The microcontroller must detect changes in height with an accuracy better than 15cm.	4: A: Set up IMU, barometer, and microcontroller on a breadboard.Continuously log the height values to disk via UART. B: Fix the breadboard on ground for thirty seconds. C: Move breadboard up to head and stabilize for thirty seconds. D: Compare the difference in height values by inspecting log and if they are within 15 cm then the requirement is met. E: Repeat B to D for head to ground.	Yes
Buttons		
1: *The buttons are required to be robust and persistent with a success rate greater than 90%. Success means our microcontroller is be able to detect it and change modes accordingly if any button is pressed.*	1: A: Connect three buttons with our microcontroller. Log a different message via UART for each button preseed. B: For each button, press the button 100 times and count the number the true positive detections. Check that the number is more than 90 times.	Yes
Mode LED		

1: *The LEDs are required to be able to last for at least five hours.*	1: A: Connect our microcontroller with a single LED and 330 Ohms resistor. If LED is able to be visible for at least five hours continuously, it satisfies the requirement.	Yes
2: *The LEDs are required to be visible to indicate which mode is currently on.*	1: A: Connect our microcontroller with three LEDs, three 330 Ohms resistors, and a button for each LED. Check to see if LED is visible once its button is pressed and we can tell the difference between the ON LED and OFF LED. B: If only one LED is able to be on at one time, it satisfies the requirement.	Yes
Bluetooth Module		
1: *The Bluetooth module is required to be able to send data to phone within 10 meters.*	1: A: First connect bluetooth to phone within a short range, e.g. 1m. B: Increase the distance from the initial distance up to 10m and see whether connection is still stable by reading App status message.	Yes
2: *The Bluetooth module is required to be able to send data to phone within 0.5 second.*	1: A: Connect bluetooth to phone within a short range, e.g. 1m. B: Output time1 from microcontroller before it sends fused data (Roll, Pitch, Yaw and height) to bluetooth. Output time2 from phone after it receives fused data from bluetooth. Make sure the time difference is smaller than 0.5 second. C: Increase the connection range from the initial distance up to 10m and make sure the time difference is still within 0.5 second range.	Yes