Team GLOVE - Wireless Gaming Gloves

EECS 473
Fall 2016

Team members: Nick Bertoldi, Ben Heckathorn, Ryan O'Keefe, Adrian Padin, Tim Schumacher

Introduction

Over the course of the last decade, video game makers have sought to create a more immersive experience through high definition graphics, virtual reality, and augmented reality. However, there are currently only two options for controlling these games: a standard controller, or a keyboard and mouse. The divide between immersive gameplay and a traditional, button-based control scheme has created huge demand for a controller solution that provides a more realistic gaming experience. The focus of our project was to develop a glove-based controller, which would allow a user's hand and finger gestures to be translated into control inputs for a computer game. We were able to accomplish this through the use of touch, motion, and flex sensors to allow for a wide variety of different input methods. These inputs could be used both in existing games, and as new options for game developers in the future.

The result of this project was a pair of wireless gloves that handle communication via Bluetooth Low Energy (BLE). The gloves could then send input to a Windows computer, and behave like a wireless keyboard and mouse. Each glove contains various sensors used to capture different gestures, which are then translated into PC input. The glove inputs can be adjusted through a configuration app to allow customization of controls for specific games. The final product meets all of the fundamental goals of the original proposal with the exception of the heart rate sensor, which was removed early in the development cycle due to constraints in engineering time. During the COE Design Expo, several games, including Portal 2 and Fallout 4, were demonstrated successfully via glove control.

Project Description

The focus of the project was to create a more intuitive, immersive, and unique way to interact with existing computer games. Currently, the main control options for computer games are either a keyboard and mouse combo, or a USB controller. These input methods rely on the use of several buttons, and do not track hand motion or gestures. Therefore, these outdated control methods interrupt the feeling of immersion that computer games aim to provide. There is currently a gap in the market for a controller that feels more natural and does not require the user to hold something and press buttons. This means there is currently a demand for a controller that feels more like an extension of our natural movement. Products like the Nintendo Wii and HTC Vive have attempted to fill this gap to some degree, but they still face technical shortcomings including: inaccurate hand and wrist motion, or reliance on input via button presses. Ideally, a user should be able to perform actions in the real world, and have them translate directly to on-screen actions with a high degree of precision. Currently, precision performance capture gloves, designed to track hand position and gesture, exist, but they are not built for gaming. The integration of this advanced technology into gaming may be years away, so we chose to develop a glove-based controller that worked with existing games in an attempt to bridge the gap between current and future technology.

High-Level Description

Our group decided to create a pair of gloves that could operate in place of a keyboard and mouse to control existing computer games. The gloves allow a user to have their hand and finger gestures translated into standard keyboard and mouse inputs. This is accomplished through the use of touch, motion, and flex sensors allowing for a wide variety of gaming inputs for use in both existing games, and as new options for future game developers. For example, a user can squeeze their finger and the computer could interpret this as “spacebar”, which could then be interpreted as a jump command. A major benefit of this method is that the gloves can mimic any keyboard or mouse input. Therefore, they can be used with any game, and an easy mapping from hand gestures to in-game actions can be provided.

It is important to note that the glove is not meant to be a “virtual-reality” glove. It does not track absolute three-dimensional hand or finger position, and it does not provide any haptic or vibration feedback, which was one of the initial stretch goals. Rather, the focus of this project is to provide a different way to play existing PC games now, and to build an interface that could easily be integrated into future games. This would allow future games to take advantage of the unique inputs that the gloves provide, and make better use of the data provided.

System Layout

The layout of the system is two gloves, one for the left hand and one for the right hand, and a receiver board connected to the computer via USB. Each glove continuously collects data from sensors and then broadcast that sensor data over Bluetooth Low Energy (BLE) to the receiver, which analyzes the data and maps it to standard keyboard and mouse input. This input is sent to the computer using the Human Interface Device (HID) protocol (see Figure 1 below). The sensors on the glove are flex sensors that measure the deflection of each of the four fingers, capacitive touch sensors that function like buttons when the thumb is pressed to the pad of a finger, and an IMU for motion-sensing to determine the absolute pitch, roll, and yaw of each hand (layout in Figure 2).

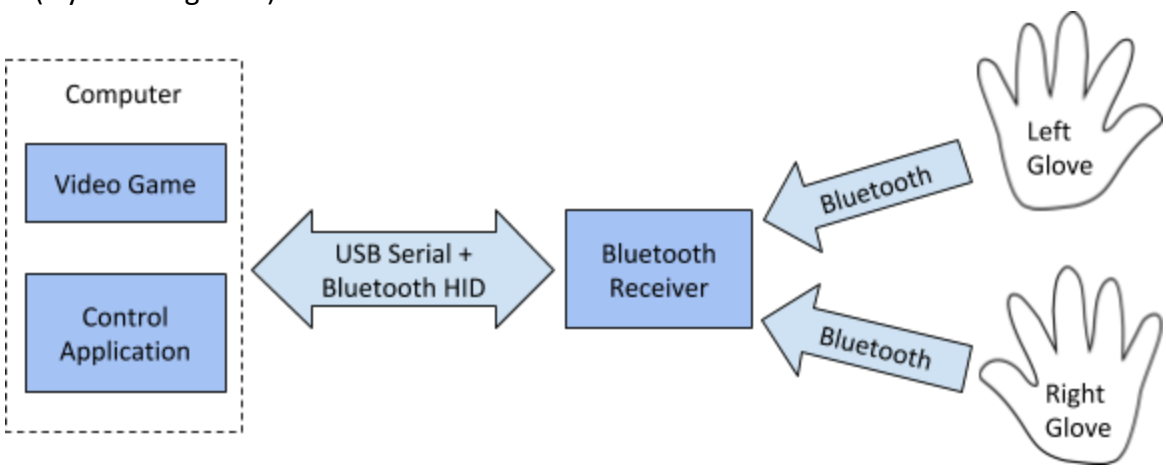

Figure 1: High-level diagram of user interaction with the gloves. Gloves collect sensor data from the user and send it via Bluetooth to the receiver.

Sensors

The gloves use a combination of flex sensors, a capacitive sensor, and an IMU to capture gestures form the user (Figure 2). The flex sensors are made of a thin, flexible material that has a variable resistance, and measure 2.2” long. The sensor’s resistance changes from 30kΩ when straight to 90kΩ when fully bent. By placing the sensors along the middle knuckle of each finger, the bend of the finger is precisely measured. Along with the flex sensors, the AT42QT1070 capacitive touch chip sits on the PCB and connects to the touch sensor pads via wires in the fabric. The touch sensor chip measures the capacitance of the pads, and registers when the value changes. The final on the PCB is the BNO055 IMU. This particular chip combines a standard IMU with a ARM Cortex M0 to provide euclidian position information to the processor at a rate of 100 Hz.

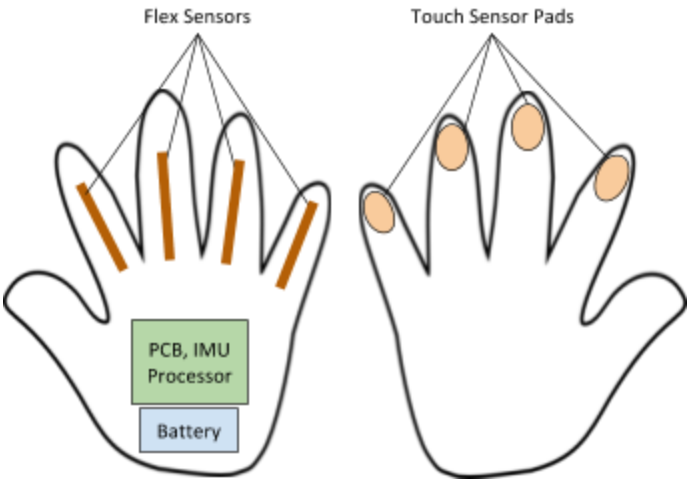

Figure 2: Layout of sensors in the final glove design. Back of the hand (left) and palm of the hand (right).

Glove Construction

The final glove assembly needed to be lightweight, comfortable, self-contained, and reasonably professional-looking (see Figure 3). This was important in order to differentiate the project from previous glove-based projects in the class that were fairly haphazard in their construction, since a poorly-constructed prototype not only appears unfinished and unprofessional, but can develop functional issues with loose or poorly connected wires. Our design strategy, therefore, was to start with a preconstructed base glove, attach sensors to it, cover the sensors to protect them, and then only expose the touch pads and status lights. The electronic components would be hidden

The components that make up the final construction on each base glove include a PCB, four flex sensors, four touch sensor pads, a rechargeable battery, a pair of indicator lights, and wiring to connect it all together. The base gloves are a pair of men’s large, black cycling gloves, with microsuede palms and synthetic materials on the fingers. These gloves provide enough space

for components to be securely attached, and are easy to sew additional covering onto for the components. The PCB attaches to to the glove at the back of the palm with adhesive velcro to hold it in place while allowing it to be taken out as needed during development. Each flex sensor is anchored on the first segment of the finger below the knuckle, with the rest of their length inside a spandex sleeve sewn to the finger. Anchoring at the base end relieves stress from the delicate leads, while allowing the sensor to freely bend along with the finger. On the pad of each finger, copper foil is attached to a wire, that then plugs into headers on the PCB. The copper foil is adhesive and sticks to the glove reasonably well, and is then covered around the edges with sewn leather to give it a more finished look and extend the durability of the copper pads. Just above the wrist, the battery sits inside a band of elastic that is sewn into the glove and plugs into the header on the PCB. Finally, a large piece of leather covers the back of the glove and is secured with two sewn in snaps on the side to allow access to the internal components. Access is also required to get to the on/off switch and the micro-USB header to charge the battery. Built into the cover is the strip of two indicator lights, placed prominently on the gloves to relay status information to the user.

Figure 3: Prototype glove (top left), partially assembled final glove showing PCB, battery, lights and wiring (top right), fully assembled and working gloves (bottom).

PCB Design

A printed circuit board (PCB) was used to secure and arrange the processor, sensor chips, and Bluetooth antenna. This board allowed the final gloves to have a more rigid and compact structure than our prototype. The pcb had no explicit size requirement, aside from the need to fit on the back of an average adult hand; however, the final result is comparatively dense with an area close to 3 in^2. Both gloves share an identical PCB (shown in Figure 4).

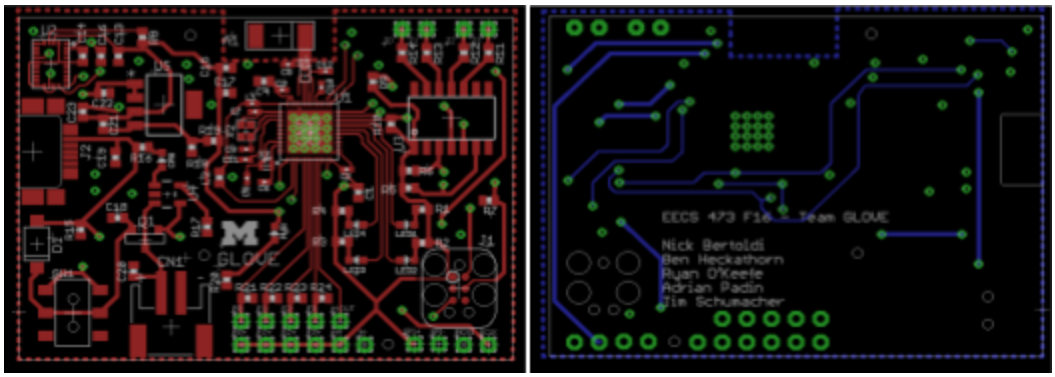

Figure 4: Eagle files screenshots of the PCB design, top (left) and bottom (right).

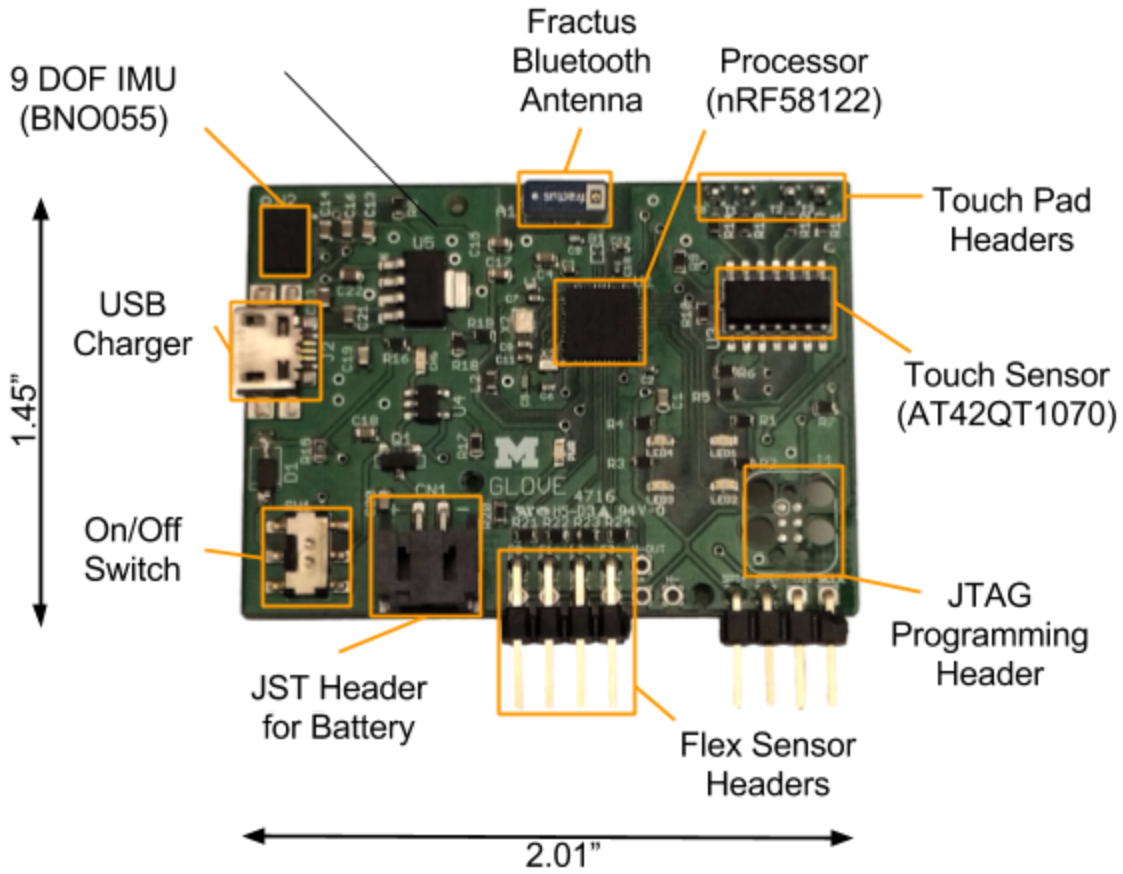

Figure 5: Layout of the major components on the PCB, including the processor and sensors

Configuration App

The mapping of sensor data to keyboard and mouse input can be adjusted through the Glove Configuration App (Figure 6). The Windows application allows the user to change the input associated with each sensor. Any of the fourteen inputs on each hand can be mapped to any standard keyboard key, or mouse input, which includes of cursor movement and button clicks. This allows the user to play a variety of games with the gloves, as well as letting the user customize their gaming experience based on which actions seem more natural.

Due to the adjustable inputs, a user is not required to use both hands at times, or be restrained to using certain fingers for specific actions. Currently, this is a major limitation in games that use a keyboard and mouse because mice are not usually designed to be left-handed, and keyboards can have fixed settings.

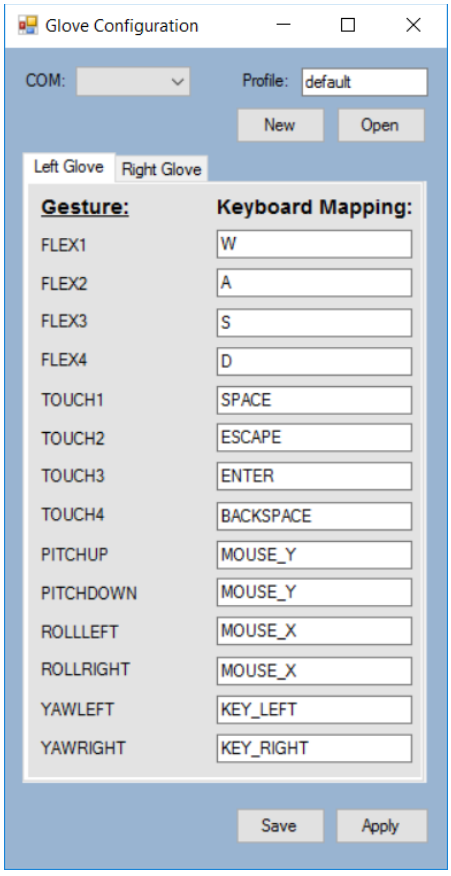

Figure 6: The configuration app

Schedule, Milestones, and Budget

Due to software and hardware delays, the project was consistently behind schedule by roughly two weeks when compared to the proposed schedule. The details of a few important time and budget setbacks are detailed later in this report. To offset the hardware production issues, we decided to focus on software earlier on to minimize testing once we received our PCBs.

The original milestone deliverables consisted of the following items:

Milestone 1

- **Flex and touch sensors working, reading data properly**
- **Bluetooth communication scheme implemented and completed**
- Receiver able to get commands from computer over USB
- Receiver able to send data through a HID profile
- Able to collect IMU data and interpret it somehow
- Starting work on desktop configuration app

Milestone 2

- **PCBs designed and ordered**
- **Prototype completed, usable with at least one game**
- IMU, flex, touch, heart rate sensors all reading data correctly
- Desktop app almost finished, usable with prototype
- Several HID profiles implemented, configurable through desktop app

Overall, the order in which these tasks were completed did not change. Nearly every task started about a week later than planned, and finished about two weeks later than previously anticipated. The original schedule had allowed for two weeks of testing and playing with the final product, so, as a result, that testing time had to be reduced to three days. However, all of the original milestone projections and features were incorporated into the final product.

The budget stayed similar to the proposed budget. Certain features had to be cut out early on, such as the haptic feedback, which was one of the original stretch goals. As a result, the buzzer motors were not purchased and fewer flex sensors were bought. There were also a series of unforeseen expenses. The biggest change to the budget was due to the delay in the original PCB order. The boards were designed and ordered the weekend before the given deadline on November 8th, but due to shipping issues they did not arrive until November 30th, which was a full week after the vendor's advertised worst case delivery time. Once it was clear that the shipment would be delayed, the group made the decision to do a second order from a more reputable site, but it still cost extra for expedited shipping to meet the deadline. Also, several mistakes were made when soldering the boards that meant that various other components also needed to be reordered. However, the budget had a contingency for this particular problem and it did not cause any major issues.

Lessons Learned

There were many mistakes that were made and many lessons that were learned the hard way in this project. The first issues arose with our original PCB design. Since no one in the group had the knowledge nor inclination to make our own component footprints, we at first limited ourselves to only using components for which we could find an existing footprint to use in Eagle, an easy to use PCB design program. We quickly realized that this was not sufficient since some of our critical components such as the motion and touch sensors did not already have footprints. We also realized after we ordered our boards that we should have added several useful debugging features such as a soft reset button, and USB data lines to read debugging information off of our processor. In addition, we should not have ordered our boards from a website which were not experienced with, as it caused delays of nearly a week and a half and ended up being lower quality than the boards we ordered later.

One very difficult issue was the use of HID for our keyboard and mouse input. At first we thought this would be easy because it seemed to be a cross-platform solution with good documentation. However, there ended up being very few hardware devices which supported true USB/HID, and integrating those devices seemed like an immense challenge. One of our group members discovered that we could implement HID over Bluetooth Low Energy and it seemed to work well at first. However, stability issues and connection drops ended up plaguing the project throughout its development. For example, if the device tried to send commands at too fast of a rate, the Bluetooth stack would get overloaded after a few minutes of playing a game and stop sending commands. Another major issue we encountered was when integrating a second glove after getting the first glove working; the two gloves would often overwrite each other's commands and inaccurate data would be sent. These issues were compounded by the use of the ARM mbed toolchain which was used to write our software, which turned out to have bugs and did not support all of the features that it advertised. In the long run, it probably would have been better to use a board such as the PJRC Teensy, which implements USB functionality rather than use HID over BLE.

Despite these issues and several more, our team learned a lot about embedded systems design. A big part of this learning process was having to start our project from scratch, picking a processor and designing a PCB based on our requirements. We learned about how Bluetooth Low Energy communication works and how to balance tasks which require long amounts of time to run, such as scanning and sending data over Bluetooth. We also learned that prototyping is critical to determine exactly what is feasible and how best to implement ideas in order to catch mistakes early on.

Contributions

Nick Bertoldi (20%) - Nick contributed to the PCB design and testing. He performed research into the communication scheme and wrote code and tests around HID over GATT communication and GAP advertisements. He also designed and implemented the backend to the configuration application and designed serial communication code to interface with the receiver board.

Ben Heckathorn (20%) - Ben contributed primarily to software design and integration. He also helped develop the high-level communication scheme between the gloves, receiver, and CPU. He helped write the software that translates glove data to HID input, and developed it to be completely configurable so that any sensor can map to any HID input. Ben also wrote portions of the receiver code by integrating various pieces of software into the final version. In addition, Ben also designed the interface for the glove configuration application.

Ryan O'Keefe (20%) - Ryan contributed to the PCB design layout. He also soldered together the PCBs when they came in. He fixed the touch sensor breakout board that we acquired for prototyping. He performed performance checking and debugging of the PCBs and fixed any soldering issues associated. Ryan also helped write the software that translates the data coming from the sensors into a specific kind of input - HID. He helped make the software configurable so that any arbitrary sensor could be mapped to any arbitrary HID input.

Adrian Padin (20%) - Adrian contributed primarily to the Bluetooth Low Energy and HID software. He developed drivers for using BLE to send keyboard and mouse data and wrote the portions of the receiver code that scan for glove data and send HID data to the computer. Adrian also helped write a good portion of the Milestone and Final reports and created figures for the final report showing the high-level diagram and glove layout.

Tim Schumacher (20%) - Tim started by prototyping the flex, touch, and orientation sensors by breadboarding them in the lab and writing the first drivers to collect sensor data. He continued to refine the sensor drivers and also developed the rest of the glove firmware that collects data from the sensors, sends BLE advertisements with the sensor data, and controls the indicator lights on the glove. In addition to the glove firmware, he designed and built the prototype and final gloves, with much of the sewing assembly done in conjunction with his mom, Marci Schumacher.

Cost to Manufacture

We estimate that a reasonable quantity to manufacture this product as a consumer electronic device is 10,000 units. This number is based on the assumption that this is an experimental device, but one that has a reasonably large pool of potential buyers. In this instance, the potential buyers are anyone who does PC gaming, and is reduced by the number of people who are willing to try the gloves and who can afford them.

The cost per board at this quantity would be approximately $22.50, which includes fabricating the boards, purchasing components, and assembling the boards. This does not include the cost of purchasing the materials for the glove and sewing them, but it otherwise gives a pretty reasonable expectation of manufacturing costs, though perhaps a bit high if quantity discounts were considered.

Final Parts List and Budget

The following two pages contain a spreadsheet of the complete part list of every component used to build this project and the total list of expenses. The part list covers all of the parts needed to make four boards, one set of complete gloves, and an extra glove as backup. The parts list is divided into the following sections:

Board components are all of the passive and active components that had to be soldered to the board. This includes all of the passive components, such as resistors, capacitors, LEDs, oscillators, and transistors, and the active components such as the touch sensors, IMU, and processor.

System components are parts that are needed to make any number of gloves, such as the leather and PCBs. Whereas the board components can be scaled down by a factor of four to make a single glove, these system parts are needed to make any number of boards.

Extra parts are parts that were used by the team in the development of this project but are not necessary to construct the final product. This includes items such as development kits, resistor kits, and spare components. Any parts which were used to make the prototype glove such as sensor breakout boards and programming materials are also included.

Part Type	Part	Quantity	Unit Price	Total Price
Board Components	Fractus Antenna	4	$1.11	$4.44
	Balun BAL-NRF01D3	4	$0.77	$3.08
	B130LAW Diode	4	$0.35	$1.40
	Micro USB Connector	4	$1.08	$4.32
	Green 0603 LED	16	$0.31	$4.96
	Yellow 0603 LED	4	$0.29	$1.16
	Blue 0603 LED	4	$0.52	$2.08
	DMP1045U Transistor	4	$0.51	$2.04
	EG1390 DPDT Switch	4	$2.50	$10.00
	NRF51822 Processor	4	$5.20	$20.80
	BNO055 Orientation Sensor	4	$10.99	$43.96
	AT42QT1070 Touch Sensor	4	$1.90	$7.60
	MCP73831 LiPo Charger	4	$0.56	$2.24
	LP3855EMP-ADJ LDO	4	$4.55	$18.20
	32.768 KHz Oscillator	4	$1.46	$5.84
	16 MHz Oscillator	4	$0.86	$3.44
	0603 Capacitor 4.7 uF	12	$0.11	$1.32
	0402 Capacitor 100 nF	4	$0.10	$0.40
	0402 Capacitor 1 nF	8	$0.10	$0.80
	0603 Capacitor 1 uF	8	$0.10	$0.80
	0402 Capacitor 12 pF	16	$0.10	$1.60
	0402 Capacitor 0.8 pF	4	$0.10	$0.40
	0402 Capacitor 47 nF	4	$0.10	$0.40
	0402 Capacitor 2.2 nF	4	$0.10	$0.40
	0603 Capacitor 6.8 nF	4	$0.10	$0.40
	0603 Capacitor 120 nF	4	$0.10	$0.40
	0603 Capacitor 100 nF	12	$0.10	$1.20
	0603 Capacitor 8 nF	4	$0.10	$0.40
	0603 Capacitor 10 uF	8	$0.41	$3.28
	0603 Resistor 1 kΩ	24	$0.11	$2.64
	0603 Resistor 12 kΩ	4	$0.11	$0.44
	0603 Resistor 10 kΩ	24	$0.11	$2.64
	0603 Resistor 100 kΩ	4	$0.10	$0.40
	0603 Resistor 470 Ω	4	$0.10	$0.40
	0603 Resistor 2 kΩ	4	$0.10	$0.40
	0603 Resistor 5.6 kΩ	4	$0.10	$0.40
	0603 Resistor 250 kΩ	4	$0.11	$0.44
	0603 Resistor 20 kΩ	16	$0.10	$1.60
	0402 Inductor 15 nH	4	$0.10	$0.40
	0603 Inductor 10 uH	4	$0.14	$0.56
System Components	2 Layer Glove PCB	4	$12.50	$50.00
	DotStar LED strips	1	$19.95	$19.95
	2.2" Flex Sensor	8	$7.95	$63.60
	22AWG 17/34 Flexible Silicone Wire 5m	1	$8.99	$8.99
	30AWG Flexible Silicone Wire 20m	1	$8.99	$8.99
	LiPo Battery - 3.7v 500mAh	1	$7.95	$7.95
	Glove Leather	1	$25.43	$25.43
	Glove Fabric / Glue	1	$23.90	$23.90

	Right angle 0.1" male header, 1x40 pin	3	$0.99	$2.97
	0.1" female crimp pins x100	2	$5.95	$11.90
	0.1" crimp connector housings, 1x2	2	$0.69	$1.38
	0.1" crimp connector housings, 1x4	2	$0.59	$1.18
	0.1" crimp connector housings, 1x1	6	$0.59	$3.54
Extra Parts	AT42QT1070 Cap Sensor Breakout	2	$7.50	$15.00
	BNO055 9DOF Sensor Breakout	2	$34.96	$69.92
	Raspberry PI Zero	2	$5.00	$10.00
	Micro USB Connector	3	$0.87	$2.61
	nRF51-DK Dev Kit	2	$39.00	$78.00
	Heart Rate Sensor	1	$24.95	$24.95
	Resistor Kit	1	$7.95	$7.95
	RedBearLab Nano BLE Dev Kit	1	$32.95	$32.95
	Micro USB Charging Board	2	$7.95	$15.90
	4.5" Flex Sensor	1	$12.95	$12.95
	2.2" Flex Sensor	5	$7.95	$39.75
	Lithium Polymer Battery - 3.7v 2000mA	2	$12.50	$25.00
	Lithium Ion Polymer Battery - 3.7v 500n	3	$7.95	$31.80
	2 Layer Glove PCB	10	$6.49	$64.90
	0.1" crimp connector housings, 2x4	2	$0.79	$1.58
	Balun Signal Conditioning Chip	11	$0.62	$6.82
	nrf51822 Processor	2	$4.95	$9.90
	32.678 kHz Oscillator	1	$1.60	$1.60
Total Spending:				$839.04

Note that the list does not cover shipping costs, which put the final expense value much closer to the maximum provided budget of $1,000. The final total cost was $999.91.

References and Citations

All code was developed using the ARM mbed toolchain and compiled using the ARM-GCC compiler. More information about ARM mbed can be found at there website here: https://developer.mbed.org/.

Several hardware and software design choices were based on similar designs from Lab 11. Credit goes to the Lab 11 crew for helping with these designs, especially the charging circuit and the Bluetooth antenna setup.

Special thanks goes to Branden Ghena, one of the Lab 11 staff who helped us a lot with both the final PCB design review, and bluetooth advertisement debugging and optimization.

The 473 GSI Seth Goldstein helped us design parts of our PCB as well. He provided sample code from his EECS 473 project that helped us write the code to scan for bluetooth advertisements and receive data from the computer over USB.

For more information about our project and to see our code for yourself, visit our Github repository: https://github.com/apadin1/Team-GLOVE.

Appendix A: Completed PCB