High 6

Brian Troili
Laura Rubio-Perez
Ali Mizan
Kirk Chan

EEL 4914
Group 6

TABLE OF CONTENTS

1. Executive Summary 5
2. Project Definition 7
2.1 Personnel 7
2.2 Motivation 7
2.3 Goals and Objectives 8
2.4 Requirements and Specifications 8
2.4.1 Overall Functional Requirements 9
2.4.2 Mobile Application 10
2.3.3 Hand Gesture Detection 11
3. Research 12
3.1 Previous Projects 12
3.1.1 Microsoft Imagine Cup 12
3.1.2 Glove Sense 14
3.1.3 GRID 14
3.1.4 California Polytech 15
3.2 Implementation Methodology 15
3.2.1 Project Management 15
3.2.2 Research Methodology 16
3.2.3 Software Design Methodology 17
3.3 Development Boards 20
3.3.1 Overall 21
3.3.2 Advantages 21
3.3.3 Limitations 21
3.3.4 ATMEL 22
3.3.5 ATMEGA 23
3.3.6 MIPS 24
3.3.7 ATMEGA VS MSP430 25
3.4 Sensors 26
3.4.1 Flex Sensors 27
3.4.2 Pressure Sensors 28
3.4.3 Inertial Measurement Unit 29
3.4.4 I^2C 30
3.5 Wireless Communications 31
3.5.1 Wi-Fi 31
3.5.2 Near Field Communications 32
3.5.3 Bluetooth 32
3.6 AVD converter 37
3.6.1 ADC081C021 37
3.6.2 ADS7828 38
3.7 PCB 39
3.7.1 Express PCB 39
3.7.2 Sunstone 39
3.7.3 Ultimate PCB 40
3.7.4 PCB4Less 40

3.8 FPGA 40
3.8.1 FPGA Architecture 41
3.8.2 FPGA Programming Technologies 42
3.8.3 Programming Technologies 42
3.8.3.1 Summary of Programming Technology 44
3.8.4 Manufacturers 45
3.8.4.1 Xilinx 45
3.3.4.2 Altera 47
3.8.5 FPGA Development Software 47
3.8.5.1 Eagle PCB Software 47
3.9 Power Source 48
3.9.1 Batteries 49
3.9.2 Voltage Regulator 51
3.9.2.1 Series Voltage Regulator 52
3.9.2.2 Shunt voltage regulator 52
3.9.2.3 Switching voltage regulator 52
3.9.2.4 Comparing regulators 53
3.9.3 Battery Charger 54
3.9.3.1 NiCd and NiMH Charger 54
3.9.3.2 Li-ion Charger 55
3.10 Software Components 56
3.10.1 Mobile App 56
3.10.2 Mobile Platforms 57
3.10.2.1 Android 57
3.10.2.2 iOS 59
3.10.3 Programming Languages 60
3.10.4 Machine Learning 62
3.10.4.1 Effect of Probability on Machine Learning 63
3.10.4.2 Machine Learning Types 65
3.10.4.2.1 Classification 65
3.10.4.2.2 Regression 66
3.10.4.2.3 Clustering 67
3.10.4.3 Hidden Markov Method 67
3.10.4.4 Libraries and Technology 73
3.10.4.4.1 Weka3 74
3.10.4.4.2 Octave 74
3.10.4.4.3 PyML 74
3.11 Unit Testing 75
3.11.1 JUnit 75
4 Design 76
4.1 Hardware Component Selection 79
4.1.1 Bluetooth Low Energy 79
4.1.2 Sensors 84
4.1.3 I^2C 84
4.2 Software Design 85
4.2.1 Compatibility 86

4.2.2 System State Machine 87
4.2.3 Android Application 88
4.2.3.1 Bluetooth 88
4.2.3.1.1 Basic Bluetooth Terminology 88
4.2.3.1.2 Giving BLE Permissions 88
4.2.3.1.3 Finding Bluetooth Device 89
4.2.3.1.4 Auto connect Bluetooth Devices 90
4.2.2.1.5 Connecting to Gatt Server 90
4.2.3.1.6 Receiving BLE Data 93
4.2.3.1.7 Translating BLE Data 94
4.2.3.2 Gesture Unit 95
4.2.3.3 Application Manager 95
4.2.3.3.1 Graphical User Interface 96
4.2.3.3.2 Menu Navigation System 99
4.2.3.3.3 Library Gesture Editor 100
4.2.3.3.4 Translator 103
4.5 FPGA 104
4.5.1 Development Boards 104
4.5.2 FPGA Components 105
4.6 Glove Aesthetics 106
5 Prototype 107
5.1 Test Environment 107
5.1.1 Temperature 107
5.1.2 Weather Conditions and Limitations 108
5.1.3 Electrostatic Discharge 108
5.2 Hardware Specific Prototyping 109
5.2.

Appendices
Appendix A - Permissions
Appendix B - References
Appendix C - Table of Tables
Appendix D - Table of Figures

1 | Executive Summary

The concept of High 6 opens the scope of mobile devices to becoming mediums of translating hand gestures through a glove peripheral. Paired up with a modern day smartphone, this product would improve sign language speakers' ability to readily communicate to those unfamiliar with the language. Users would use the language glove, capable of detecting the hand gestures, to make the appropriate gestures for the message. This glove would automatically forward the information wirelessly to the speaker's mobile device. With the power of today's smartphone, High 6 conveniently recognizes and translates the hand signs to an audible and readable message for their respective audience.

The project's overall system architecture consists of three main subsystems. These subsystems are unified hand gesture detector, the Bluetooth communication, and the Android translation application. Each subsystem displayed in Figure 1 defines the overall High 6 design.

Figure 1: Top Level System Architecture

The system starts with the unified hand gesture detector. The design focus surrounds the idea of the glove requiring low power consumption for everyday use, being lightweight for portability, and wireless for a clean look. The unified hand gesture detector, or language glove, captures the variable data of each hand sign. The factors considered into each gesture include the orientation of the hand, position of the fingers, and the motion of the entire hand.

As language glove attains the information of the hand gesture, the glove subsystem prepares to forward the data to the Bluetooth. Through the microcontroller, the signals from the sensors are synchronized before the data is sent wirelessly. Consistent with our low-power design focus, the Bluetooth module uses the one of the new Bluetooth standards known as the Bluetooth low energy, or Bluetooth Smart. As the name denotes, the Bluetooth module uses less power than its predecessor. The use of this module proves helpful when considering the battery of the mobile phone and the language glove. Smartphones today may be challenged with a long battery life through normal use. Limited to being lightweight, the glove subsystem uses two lithium-ion batteries to power itself. Hence, the smaller power consumption of Bluetooth

Smart promotes longevity of the battery life for the unified hand gesture detector and the Android phone.

The next core subsystem is the Bluetooth communication subsystem using Bluetooth Smart. BLE serves as the bridge for the wireless data transmission between the Android application and the language glove. To establish this connection, the Android application transmits a request signal when in use. Meanwhile, the glove subsystem waits in standby until it hears the request. Then, the language glove would confirm the connection request. From there, the Android application will establish the connection upon receiving the approval from the glove. Now, the connection setup is complete, and the two subsystems may send packages of data.

The last subsystem, the Android translation application, is the primary language tool, focusing on the management of the gesture library and the visual display of the translation. The default library supports the American Sign Language (ASL) alphabet, numbers zero to nine, backspace, and other basic gestures. The application is also extensible through gesture training. The application is capable of "learning" from a set of the new hand sign. Through machine learning algorithms, the Android system may recognize the pattern after being exposed to a set of repeated motions in a training set. Then, the application may infer the appropriate response when the instance of the hand sign comes again. These machine learning algorithms serve as a black box to users and should not concern them. Instead, they interact only interact with the user interface of the application. Figure 3 illustrates the interactions of the subsystems within the High 6 system.

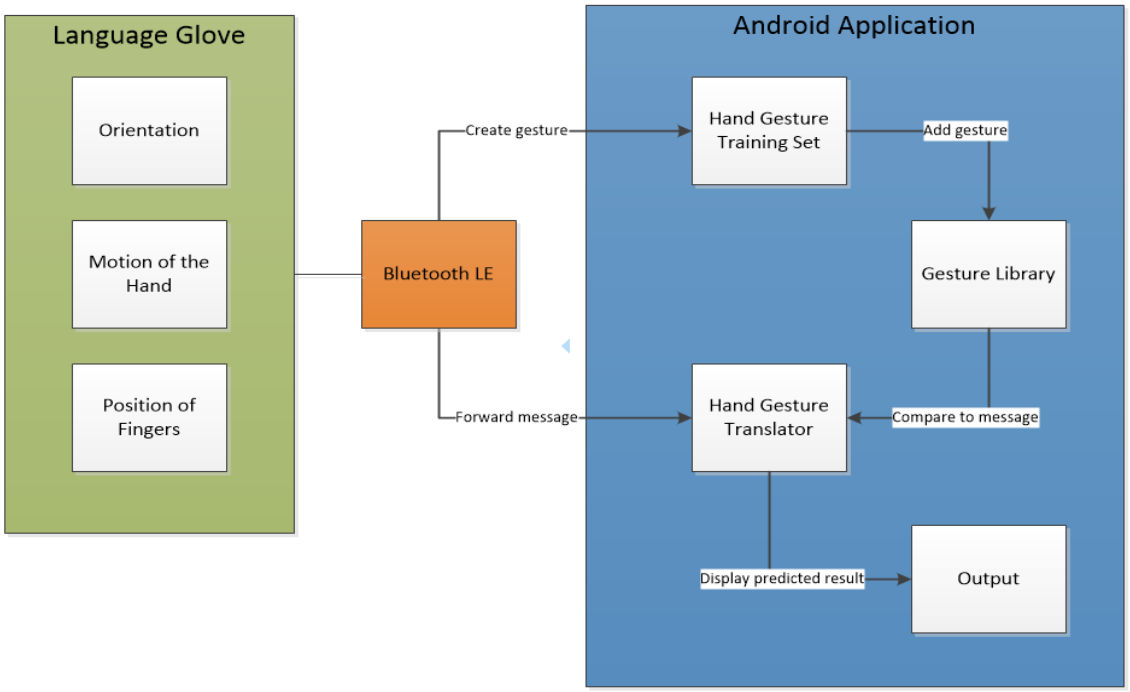


Figure 2: High Level System Architecture

To prove the practicality of the concept, the project limits the design to one glove and a select gesture library and is compliant with the performance requirements. The choice of building one glove is a budget-conscious decision. With the system recognizing only one hand's motions, it impacts the scope of American Sign Language users may work with.

ASL is an expressive language where fingerspellers speak with both the left and right hand. With the system capable of capturing one hands' motion and hand shape, the word bank available is extremely limited. Thus, the default library provides the alphanumeric characters along with the feature of adding, editing, and removing gestures.

The High 6 glove system, including the application, meets our non-functional specifications. In particular, the Android application needs to provide an accurate response to the given gesture within 1.5 seconds. To provide the convenience of this translation tool, the app must reflect this requirement in a timely fashion. Achieving its project goals and specifications, High 6 proves to be a practical and convenient tool for the users of sign language.

2 | PROJECT DEFINITION

2.1 | PERSONNEL

Brian Troili
braintroili@knights.ucf.edu

Laura Rubio-Perez
lalis.rubiete@knights.ucf.edu

Ali Mizan
ali.mizan@gmail.com

Kirk Chan
chan.kirk@knights.ucf.edu

2.2 | MOTIVATION

Group 6 expressed an interest to develop a product with purpose. We decided to search for a project idea where we could find compromise between our budget and interests since our group is interested in both hardware and software. At the same time, the group members wanted to work on a project that would allow them to gain knowledge in microcontrollers and machine learning. From our research came the idea where we develop a glove with the ability to translate hand gestures into audio and text.

The 2012 Microsoft Imagine Cup and previous senior design projects such as the “GRID” project, designed a glove to aid first-response personnel and sign language speakers as well as a way to interface with a computer. This project peaks our interest, it was fresh to the UCF community and our group is interested in the technologies. Together, Group 6 wishes to design a glove equipped in translating hand gestures through an Android mobile.

2.3 | GOALS & OBJECTIVES

Our project design aims to develop a user friendly, low-powered, wireless glove to translate hand gestures via an Android device. The glove system will interpret the data from a variety of sensors to capture the four parameters of the American Sign Language. The four parameters that distinguish sign language gestures are hand shape, orientation of the palm, location of hand, and movement. Through a Bluetooth module on the glove, the data will be forwarded to the Android mobile device. Using hand recognition algorithms, the gestures will be translated from sign language letters to words. Then, the Android system will display and pronounce the message made.

With this glove, we are mainly aiming to help people with hearing and speech impairment. By taking into consideration the four parameters of the American Sign Language, we are making sure that this glove paired with the android device will fulfill the purpose of allowing and improving the communication of this community. This glove will contain two types of sensors to detect the different position of the fingers and hand shape, as a well as a gyro/accelerometer to detect the hand movement and orientation.

Our main objective is to be able to translate the 26 letters of the American Sign Language’s alphabet. Once this is working properly, we would like to take the challenge to improve our glove so it can translate the hand gestures that are mostly used in sign language. As second objective, our glove will have a user friendly app, in which the user will have the capability to select a specific language. This feature will allow the translation of hand gestures into different speaking languages, making it very diverse. As final and optional goal, we are considering the possibility to create two compatible gloves that will work simultaneously with the android app, only if we can achieve the previous goals first.

2.4 | REQUIREMENTS & SPECIFICATIONS

The Sign Language glove system will be split up into several subsystems. In the following sections, we will discuss the requirements and specifications for the overall system, mobile application, and hand gesture detection.

2.4.1 | OVERALL FUNCTIONAL REQUIREMENTS

Table 1 shows the overall functional requirements the sign language glove should meet. At the same time, these requirements provide an overview of the overall system such as functional requirements, data requirements, and quality requirements.

Overall Objectives
1. The sign language glove must be weight less than two pounds
2. The sign language glove must use a low-powered microcontroller unit
3. The sign language glove must use a Bluetooth transmitter
4. The sign language glove must communicate with an android device
5. The sign language glove must use an gyro/accelerometer to handle orientation and detect movement
6. The sign language glove must have two flex sensors per finger to detect partial flexes
7. The sign language glove must have one flex sensor across the palm
8. The sign language glove must handle up to 2-G movements

Table 1: Overall System Objectives

The following diagram, figure 1, outlines the overall structure of our sign language glove:

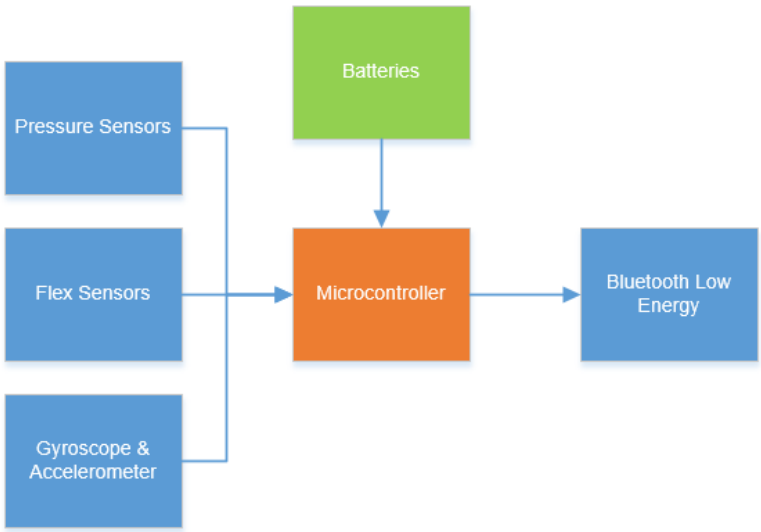


Figure 3: Language Glove System Architecture

2.4.2 | MOBILE APPLICATION

The following table, table 2, shows the non-functional requirements the sign language glove's mobile application shall meet.

Non-functional Objectives
1. The mobile application shall be able to display and pronounce the words in multiple languages
2. The mobile application shall be easy to use
3. The mobile application shall display readable text and provide customization of font styling
4. The mobile application shall display each letter of the alphabet in less than 1.5 seconds
5. The mobile application shall pronounce each word in less than 1.5 seconds
6. The mobile application shall be created using the android application framework
7. The mobile application shall interact with the Bluetooth module
8. The mobile application shall not drain the battery of the android device
9. The mobile application shall be available to the users in the android market
10. The mobile application shall be free to the users

Table 2: Non-Functional Objectives

The following diagram outlines the basic system structure to our mobile application:

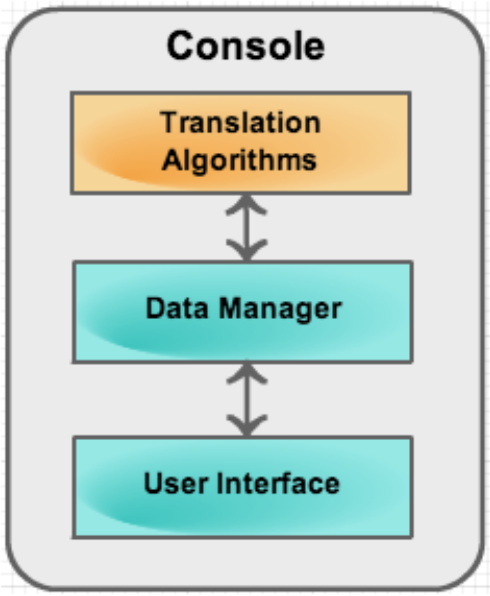


Figure 4: Mobile Application System Architecture

2.3.3 | HAND GESTURE DETECTION

Table 3 shows the requirements needed on a hand gesture algorithm in order to interpret each hand movement and translate it into the 26 letters of the alphabet.

Hand Gesture Detection Algorithm
1. The algorithm shall recognize the 26 different hand postures of the alphabet
2. The algorithm shall recognize the orientation of the hand and detect movement in general
3. The algorithm shall create a library with the 26 hand gestures for easy detection
4. The algorithm shall detect and identify a hand gesture in a timely manner
5. The algorithm shall take up to 1.5 seconds to detect multiple hand gestures

Table 3: Detecting Algorithm Features

3 | RESEARCH

3.1 | Previous Projects

The idea of a smart glove is gaining popularity with the possibility of three dimensional interactions with a computer interface. The many variations developed in recent years are showing just how versatile the technology can be. Therefore a wealth of information is available and ready to be referenced. Projects include designs of a smart glove for Graphical User Interface (GUI) interaction, to other gloves capable of sign language translation. Despite the idea already being successful, there is still plenty of room for improvement with these “Sign Language Gloves”.

3.1.1 | Microsoft Imagine Cup 2012

The Microsoft Imagine Cup is an international student technology competition running for the past 10 years. In 2012, the winner of the Software Design Competition was Team QuadSquad from Ukraine– the developer of Enable Talk. Enable Talk is a glove which translates sign language into audio via smart phone. This technology can potentially connect the 70 million people with hearing and speech impairment with those who do not know sign language. The community for the hearing and speech impaired remains a very exclusive group; the language rarely extends beyond the impaired and their families.

The smart glove designed used a total of 15 flex sensors to measure the radius of curvature of the finger when bent. The smaller flex sensors were used so flexion could be detected at two separate points on the fingers and thumb. This is due to the fact that the flex sensors can only detect flex at the point closest to the sensors connection leads. Also due to the subtle nature of hand gestures in sign language, pressure sensors are used to detect small differences between alphabet characters that only differ by contact at specific points.

A system of accelerometers and gyroscopes interpret the motions made by the user’s hand in three dimensions. The accelerometers and gyroscopes are a very complicated system and usually come as a combination set. The device has six degrees of detection so as to differentiate between motion vectors of the three dimensions of space of the two separate measurements, acceleration and orientation.

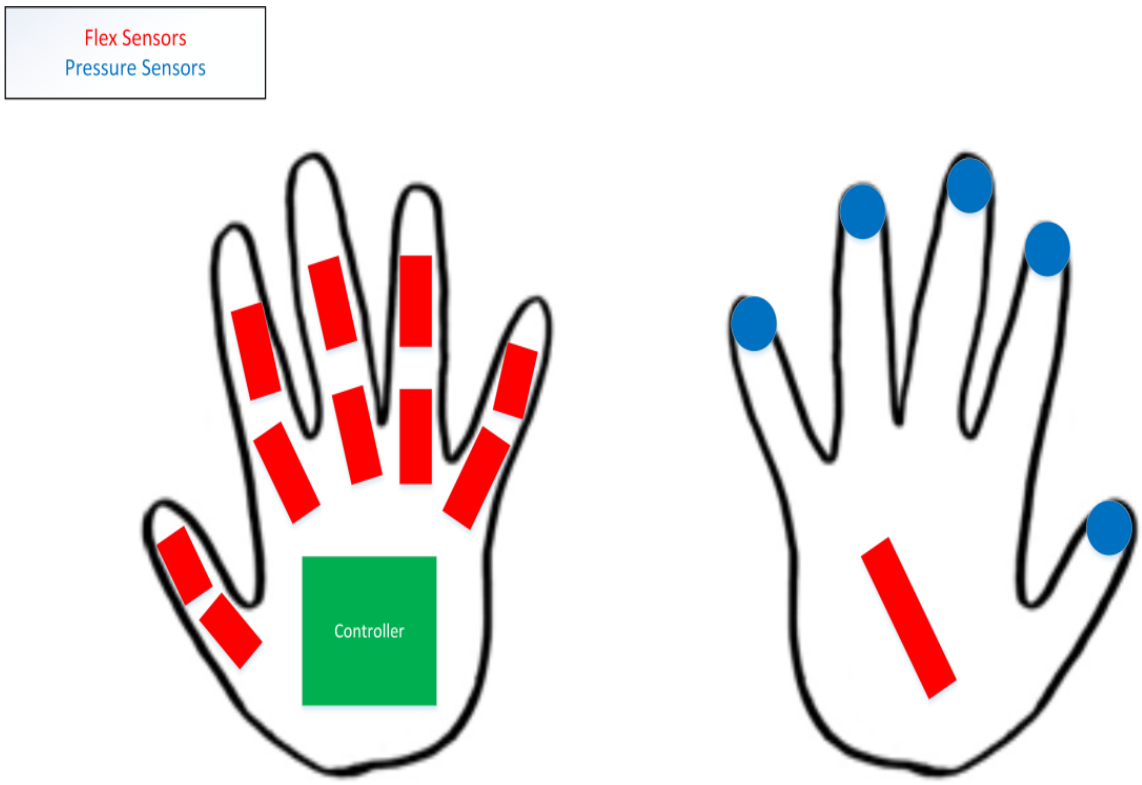


Figure 5: Placement of sensors for Enable Talk from top and bottom view

This results in a set of information which can be understood by the hardware and sent via Bluetooth to a smart phone. There the data is translated by the algorithms for machine learning in the mobile application and use APIs (Application Programming Interface) provided by Microsoft Speech. /the application then translates the data into an audio message and text on an LCD display. The application runs on the Windows Phone 7 mobile platform, but QuadSquad is currently working with other developers to be available on Android and iOS.

Despite the success of Enable Talk, there are still draw backs. By the time of the competition, Enable Talk could translate a limited number of phrases. This can be improved by creating a larger library of hand gestures. Also, due to its processing speed, conversation speed was hindered. The accuracy of translations was approximately 90%, leaving the user to attempt to edit or retry the gestures. These problems leave plenty of room for improvement in design.

The technology for Enable Talk also extends to a much more broad area of development: Wearable Computers. One major example is the ability to interact with a PC's GUI (Graphical User Interface). A previous project from the University of Central Florida has already been developed, as we will see in a later section. It is an important reference piece for the High 6 project.

3.1.2 | Glove Sense

Luke Anderson & Company are the members of Team GloveSense – a group which developed a smart glove that communicates using hand gestures. Allowing emergency personnel to communicate nonverbally is suggested by the US department of Homeland Security to be of great importance. The current inability to effectively communicate has a negative result and contributes to on job injuries of emergency personnel. Team GloveSense took the approach of a smart glove to help with nonverbal communication on location.

The glove developed acts much in the same way as the previously discussed Enable Talk. A collection of sensors work with an MCU (Microcontroller Unit) to process signals and sends the data wirelessly. The part separating it from Enable Talk is a larger gesture library which is also expandable to encompass a larger vocabulary list. The larger vocabulary is key to a successful end product since the user will want chose a product capable of communicating at a comfortable pace. This is an issue that will be later addressed by Machine Learning.

3.1.3 | G.R.I.D.

The Gesture Recognition Interface Device (GRID) is a project developed by a group of students from the University of Central Florida. The device measures hand gestures via three sensors: a camera, accelerometer, and a gyroscope. The camera records at 100 frames per second as a balance between accurate measurement and power consumption. All information collected from the sensors are sent via Bluetooth (or USB when charging) to the computer.

The wireless device requires a 3.7 V rechargeable battery for the power supply which will last up to 14 hours for the main power supply. A second battery is used for the camera; it requires much more power for the constant feed of data being supplied. For this reason a “Play and Charge” feature was implemented to allow the user to charge the glove while still having the ability to interact with the personal computer.

The LEDs implemented are tracked by the camera for the location of the cursor. The camera decided upon was a webcam modified to view only near infrared light at a frame rate of 60 Hz. Through the set of sensors and data collected a mouse pointer can be moved around on screen and “clicking” actions can be performed. Also, through touch sensors on the fingers, zooming in and out can be performed.

3.1.4 | California Polytechnic

Nicholas Born, an Electrical Engineering student at California Polytechnic State University, is the developer of a sign language glove. The device used a PS/2 port to behave as a keyboard for a personal computer. The design implemented used a collection of flex sensors, pressure sensors, and an optical sensor (camera). The documentation had several important flaws listed about the design to take note of.

There was one drawback of the design being a startup signal was not used by the glove; therefore the computer had to be turned on with a regular keyboard already plugged in only to be swapped out. Also, after enough use the resistance of the flex sensors began to change. Another problem arose with the sensor output voltage only changing 10 percent from its maximum value, thus requiring increased sensitivity due to subtle differences in some of the letters in American Sign Language.

3.2 | IMPLEMENTATION METHODOLOGY

3.2.1 | PROJECT MANAGEMENT

Team High 6 actively collaborates to meet the demands of Senior Design. From the early weeks of the semester, the group quickly recognized each other's skill set and focus.

Member	Major	Interest
Brian	Electrical Engineering	Embedded Systems
Laura	Computer/Electrical Engineering	System Architecture
Ali	Computer Engineering	Algorithms in software
Kirk	Computer Engineering	Software architecture & design

Table 4: Team Member Focus and Interest

From the common interests of the group, the team identified the focus of their contribution to the project, primarily electrical circuit design or software design. Each member played an equal role in documentation. With the active communication and the time constraints of the project, the group organized themselves in an agile project management environment.

The agile management is an iterative and incremental approach to manage the design and build engineering projects. It composes of three parts: plan, collaborate, and deliver. Planning is a collection of the tasks to complete in the scope of the project. Then, taking one of these tasks, it undergoes development and review and repeats until the task complete in the collaborate stage. Once the current task is complete, it enters delivery to become part of the final product or another task. Through a repetition of these stages, there is a fine granularity within the development of the product and is useful for our project.

Group 6 requires an original design to the glove system and the Android application. With the development of these two systems, there is a degree of uncertainty when approaching the design. In the face of a design oversight or fault, there needs to be a fast response to the issue and issue a redesign. Constrained to four months to create the solution, agile management allows the group to adapt to changes rapidly as the group tackles tasks incrementally.

3.2.2 | RESEARCH METHODOLOGY

The first question that comes to mind when thinking of an algorithm for interpreting sign language gestures is how to compensate for the various ranges of motion that the users hand can have. An analogy for this is human handwriting: it comes in various shapes, spacing, sizes, legibility, etc. A human can easily recognize a letter ‘a’ when they see it in front of them, but what about a computer? How can a computer be taught to recognize patterns the way that a human can? The answer is through machine learning.

How does a human know that what they’re seeing is the letter ‘a’ as opposed to the letter ‘o’? The reason is because the person has seen the letter’s ‘a’, and ‘o’ all throughout their lives, and when they see another instance of the letter ‘a’, they can go ahead and recall what the general pattern of what an ‘a’ looks like based on all the examples they’ve seen before. This is how a person recognizes a pattern. Doing a Google search on pattern recognitions, one would learn that Machine Learning uses the same approach to teach a computer to recognize patterns the way humans do. And thus, it was decided that machine learning would be the approach to creating an algorithm to interpret sign language input.

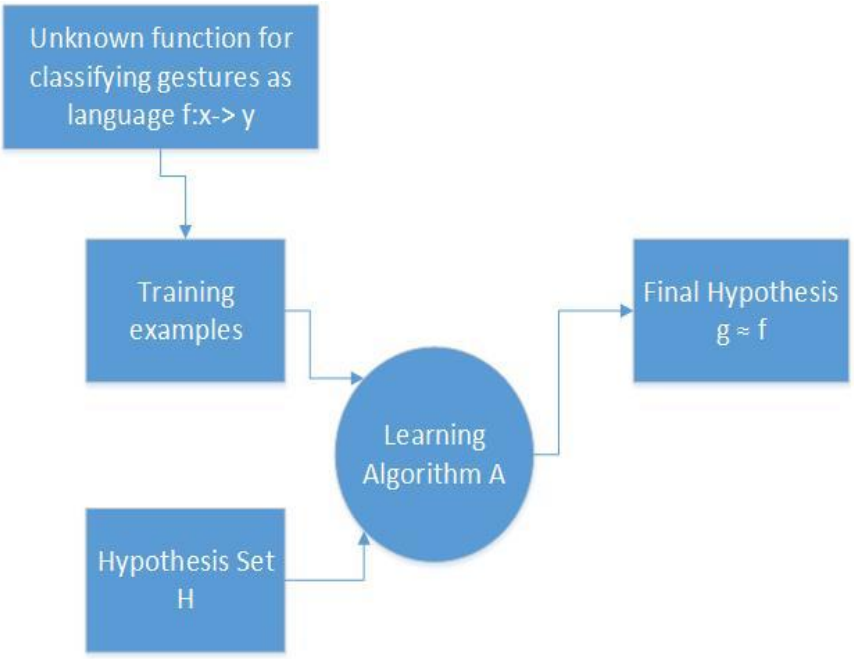


Figure 6: Learning Algorithm Theory

The above diagram shows the general approach that was researched. Basically, in real life, there's some type of extremely complex and unknown target function that classifies some input X (which is the set of all sign language gestures made by a person) as letters Y (the set of all letters that sign language gestures represent). This algorithm is too complex to be discovered exactly. But there's plenty of data on it: we can have people basically go through the entire sign language alphabet from A to Z, and then the glove would have a training example of what all the letters look like. But just 1 sample of each letter isn't enough. The glove would need multiple examples, so that the learning algorithm can go ahead and see what the general pattern is between all the gestures for each letter. At that point, the algorithm will pick an algorithm from the hypothesis set that most closely matches the unknown target function. The function picked from the hypothesis set depends on the training data.

3.2.3 | SOFTWARE DESIGN METHODOLOGIES

This section talks about the software design method used to create the android application that will be processing data from the sign language glove. The 3 main methods that were considered were: the waterfall method, incremental development method, the agile method.

For the waterfall method, each sub-problem goes through 5 stages which are shown in the image below.

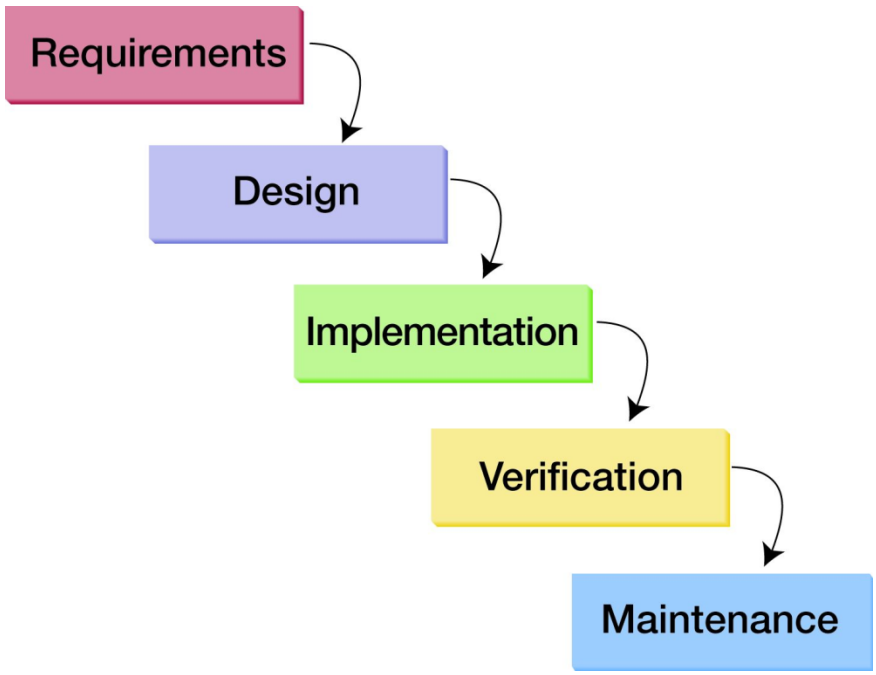


Figure 7: Waterfall Model

First, the requirements are stated. So for instance, if the application needs to take in a sign language gesture from a human hand and then spit the 8 bit ASCII character onscreen of the android application, then that's one of the requirements that need to be listed. Not only that, but it is noted that the human hand fluctuates vary greatly when making sign language gestures, just like how people's handwritings vary from person to person. Anything that makes the problem harder to solve should be kept in mind when going onto the next step: design. In the design step, a plan is made to meet all the requirements. For the requirement stated earlier, it can be decided that a statistical model such as the Hidden Markov method will be used to recognize human hand gestures. After a design has been created, it is then implemented. Afterwards, the only way to know for sure whether or not it worked is to go ahead and thoroughly test it. If something failed, go back to the design stage and re-implement it. After the solution has been verified to meet all the requirements, all that needs to be done is maintenance throughout the rest of the product life cycle. That means if Android decides to come out with a new operating system, then that means you better make sure your product is compatible with that new latest and greatest operating system. Or if it turns out there's some type of weird bug that you didn't catch earlier, then go ahead and implement a fix for it.

The iterative model takes a different approach. The problem isn't solved as a whole by itself at first: the problem is solved via a divide-and-conquer technique. In incremental development method, the problem of trying to create an application that can use pattern recognition techniques to recognize input from the sign language glove will be broken down into small, simple problems. Problems include: getting an android application to install on the phone, making sure that the android application is receiving data from the glove, making sure

that the data that the glove is receiving isn't being interfered with, figuring out a way to implement the Hidden Markov method to do pattern recognition on the gloves signals, etc. For each and every little problem the waterfall method is applied to go ahead and solve a problem, and the go onto the next problem until all the sub-problems are solved and there's an overall solution that meets all the requirements.

All in all, the incremental method completely beats the waterfall method. With the incremental method, a developer doesn't really have to assume anything. In the waterfall method, it may not be known what format the data from the glove is in before the android application goes ahead and tries to use it. So assumptions must be made about it in the design phase. However, in the incremental method, the android application won't be developed until the glove is ready to send data to the phone because that's the next stage in the development. At that point, the format of the data will be known before algorithms that handle this data are being developed.

Agile methodology is a highly collaborative method based on iterative and incremental development. First, a closer look into iterative and incremental development is needed. This method is a combination of iterative and incremental. It starts with simple uptake on a subsystem of software and repeat. Through the repetition of developing a piece of the code, the piece of software improves with the experience learned before and new features are added. Over time, more and more of each task becomes completed until the end of the project. Now that there is a basic understanding of the iterative and incremental method, there may be a deeper dive into agile.

Agile development works in a familiar fashion but finishes a task in one iteration. The project divides the work into smaller tasks with less planning and long-term planning compared to the other method types discussed. In exchange for the brevity in planning, there is more interface with the business user and more iterations in short time frames of development. These periods could vary from one to four weeks. By the end of each iteration, meetings with the investors are held, allowing for the product to be cross checked with the system requirements. Through continuous review of the project, the agile method reduces risk in the development process. A design flaw or mistake in the implementation could prove fatal to a project taking a more rigid design methodology. Agile promotes adaptability to setbacks in the design as the developing solution is under more exposure.

Within the agile method, there are underlying team mechanics that projects build around. First, it starts with a motivated team who may be trusted to quickly develop good code and adaptive to change. Because how short the iterations are, the members must be able to comply with the continuous needs of the ongoing tasks. Each member must also be self-organizing as an individual and with the team. This goes hand-and-hand with the typical approaches in an agile

environment to maintain order in an agile team: stand-up meetings and storyboards. Stand-up meetings, or the daily scrum, open the floor to a brief session where team members give insight into each others' progress yesterday, today, and plans for tomorrow. A storyboard is visual tool to outline the status and priority of all tasks for the entirety of the project. As a task progresses through different stages, the developer responsible for the task will update the storyboard. Figure displays a sample storyboard.

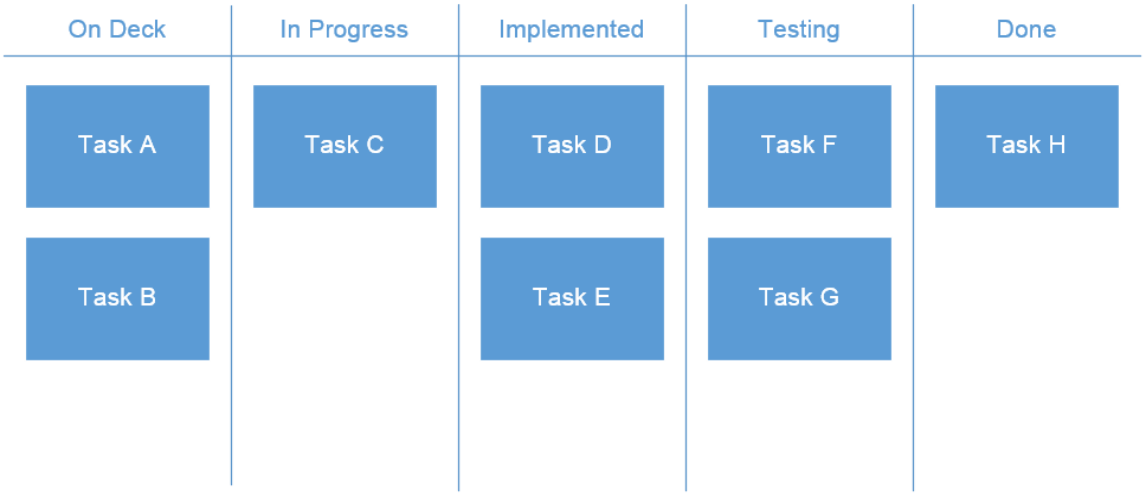


Figure 8: Storyboard used in agile software development

Through the high levels of communication, agile teams are able to achieve sustainable development within short periods of time.

Under the conditions of the project, agile may prove to be a useful approach for the project design process. Given one semester to build the design researched earlier, the characteristics of agile development favor a small team. The risk of facing a design issue would be detrimental to the project. These conditions of a smaller project could not afford a more traditional model, such as waterfall, or a redesign of the software architecture with prototyping. An aggressive stance in programming would allow the user to address changes more quickly. Hence, an agile lifecycle through dividing tasks and regularly checking the system against the project requirements would ensure the success of the software component.

3.3 | DEVELOPMENT BOARDS

In this section, we will present our research on the various microcontrollers we considered for the development and execution of the sign language glove. First, we will provide background information on microcontrollers; then, we will discuss their advantages and disadvantages. Finally, we will talk about the different microcontrollers available on the market that may be a good fit for the requirements of this project.

3.3.1 | OVERALL

Microcontrollers are integrated chips, commonly they are composed of a processing unit, memory devices such as read only memory(ROM) and random access memory(RAM), timers, I/O pins, and registers; whose main purpose is to control some processes; in other words, these tiny devices act just like a microcomputer with our the digital part. Moreover, these integrated chips are used in embedded systems, such as digital cameras, cell phones, automobiles and any specialized device. Their price is really affordable, they have a limited, and they are not versatile.

Microcontrollers execute bit operations called “instructions” in order to perform a specific task. They take data as input and by following a certain program in memory, it produces a control signal as output. In the market there is a wide variety of these devices available. They are classified by architecture, size of bus width, programming language, and memory. The most recognized and used are made by Texas Instruments, Atmel, and Microchip.

3.3.2 | ADVANTAGES

Since microprocessors are composed of various components, they have a faster speed of execution. At the same time, these devices have a low cost and a higher level of integration, since most of its resources are included on it. Due to their low cost, multiple microcontrollers are found on intensive applications. These devices are designed for a determined purpose, for example to certain information in a LED; they do not require additional or external devices to help the functionality of microcontrollers and they are small, they fit inside other appliances.

With this in mind, microcontrollers are more flexible to reprogram to change its functionality, if needed. This re-programmability makes microcontrollers for prototyping. Moreover, microcontrollers are easy to use, especially to program using assembly or high level languages; for instance, programs can be compiled using C programming languages. The compilation of C codes or any other high level language, allows simulations to be performed in microcontrollers, which permits to check for system requirements and to guarantee correctness in the code. In conclusion, microcontrollers are dependable, reliable, cost efficient and have low power.

3.3.3 | LIMITATIONS

Even though microcontrollers have great advantages that have helped improve our day to day lives, they have limitations that are worth considering. The biggest limitation that microcontrollers have is the limited number of executions they can perform simultaneously. Furthermore, these devices have a limitation on speed;

for instance, when the amount of data to be processed increases the microcontroller is not able to interpret the given data fast enough.

The memory size of microcontrollers is another limitation, the lack of memory size might result on time delay while waiting for data to be ready for the microcontroller to use. Not to mention, microcontrollers cannot pair up directly with high power devices. The second biggest drawback to consider is the programming complexity. Since microcontrollers need to be programmed in order to fulfill their purpose, it is required to have strong programming skills or have a high level of training in the desired programming language that microcontrollers use.

Since there is variety of programming languages that microcontrollers can use, this can become troublesome when trying to merge two different systems to one control system. Other important limitation is the electrostatic sensitivity that microcontrollers have; these devices are highly sensible to electrostatic since they are composed of metal-oxide and they can be damaged very easily with only a static charge.

3.3.4 | ATMEL

Atmel offers microcontrollers that are optimized for low power, high-speed connectivity, optimal data bandwidth, and interface support. They also support integration of capacitive touch and they deliver wireless and security support. Atmel has four different types of microcontrollers: Atmel AVR 8- and 32-bit microcontrollers, ARM-based microcontrollers, MCU microcontrollers, and 8051 architecture microcontrollers.

The Atmel AVR 8- and 32-bit microcontroller has low power consumption and a high level of integration. In other words they have high performance, provide power efficiency and they offer design flexibility. It also has up to 2MB of flash memory and 160KB of SRAM; this microcontroller has the lowest power consumption out of the four types available. Their next microcontroller, the ARM-based, does not have flash memory but it has a single integrated development platform, which provides time saving code sources, access to simulations, debuggers, and extensions. Its performance is up to 850DMIPS at 536MHz with a floating-point unit. It also has a robust security since it has a secure boot and a hardware crypto engine.

As stated on Atmel's website, their MCU wireless offer a complete line of IEEE 802.15.4 certified wireless solutions. These are based on RF transceivers as well as Atmel AVR, ARM-based and single-chip wireless microcontrollers. Atmel's 8-bit microcontrollers a based on the 8051 instruction set as well as low-power single cycle. At the same time, these devices offer advance flash technologies; they provide binary code level compatibility, and pin-to-pin compatibility with other devices. Other optional features Atmel offer are digital signal processing

extensions, advance timer/pulse width modulation peripherals, and CAN and USB interfaces.

3.3.5 | ATMEGA

ATMega is a single-board microcontroller whose software, programming environment, and hardware are open source. The ATMega‘s software is cross-platform which allows programmers to work on different operating systems; this software was design for users that have no experience programming. The open source software is based on C++ libraries and the open source hardware is based on two microcontrollers from Atmel, ATMEGA8 and ATMEGA168.

Since this microprocessor’s software and hardware are both open source, it is very important to mention that there is a huge community that provides GUIdance and help if needed with any ATMega microcontroller. The most important aspect of ATMega is the way in which its connectors are laid, these connectors allow the board to connect with a variety of modules known as shields. These shields are boards that as mention before can connect to the ATMega PCB in order to extend its capabilities. If we decided to use this microprocessor for our project, we will be using the Bluetooth shield to forward the data to the Android device.

From the diversity of boards that ATMega has, ATMega-Leonardo has called our attention. This microcontroller is based on the ATMEGA32u4. As shown in figure 1, ATMega-Leonardo is an low power 8-bit microcontroller, it uses a RISC architecture, it has a microbus connection and has built a USB communication which eliminates the need of other processors, this means that this microcontroller has a processor that acts as both: a main processor for the board and it can communicate with the computer directly through the USB protocol.

This infers that the ATMega-Leonardo is capable of running sketches and having a USB communication with a computer at the same time. Furthermore, this UBS feature allows the microcontroller to operate as a USB device and simulate as a joystick, mouse or keyboard. This microcontroller is available in two flavors one with headers and one without them.

Summary

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz

Figure 9: ATMega-Leonardo Summary

Through the micro USB connection or an external power supply this microcontroller can be powered; 6-20 volts are the voltages that the ATMega-Leonardo can operate. As shown in figure 1, this microcontroller has 20 digital I/O pins, every single one can operate at 5 volts with a maximum current of 40mA. This microcontroller allows the communication between itself and a computer, itself and other ATMega or other microcontroller; showing that this device is not only cross-platform but it is diverse enough to allow inter-microcontrollers communications.

Furthermore, this microcontroller gives the protection from shorts or over currents to the computers, where the microcontroller it is being programmed, through a resettable fuse. This fuse provides an extra layer of protection; for instance, if more than 500mA are applied to the USB port, the resettable fuse will break the connection automatically under the overload gets removed. It is important to notice the ATMega-Leonardo's physical dimensions considering that if our group selects this microprocessor it will be embedded in the sign language glove. The PBC of the ATMega-Leonardo is 2.7 inches by 2.1 inches which if selected it will fit perfectly in the glove our group will use.

3.3.6 | MSP430

The MSP430 is another 16-bit low power microcontroller well-known in the industry, created by Texas Instruments. It is one of the most flexible microcontrollers available; it has multiple low-power modes, direct memory access, instant wake up, real time clock, and autonomous peripherals. This microcontroller is capable of not only enable but also disable diverse clocks and

oscillators, allowing the user to utilize different low-power modules. At the same time, the MSP430 has a designed memory access controller which allows data transfer from memory without the CPU.

As mention above, the MPS430 has a real time clock; this clock permits the wake up of the microcontroller at a determined time. Most of the time, this microcontroller is on standby mode unless it gets woken up to perform a determined action. Moreover this device has different analog and digital peripherals that run independent from each other. The diversity of peripherals includes radio frequency, USB, LCD controllers and much more. Not only the ATMega has a RISC architecture but the MSP430 has it as well. All operations are interpreted as register operations, each of which is divided into source and destination operands. The CPU is composed of 16 registers, four of which are dedicated to the program counter, status register, constant generator, and stack pointer and the rest of the registers are for general purpose. The instruction set of this microcontroller contains a total of 51 instructions, this allows instructions to operate as bits, bytes or word data. The combination of the addressing modes, the reduced instruction set, and the data bus, allows this microcontroller to have a shorter programming code that has a fast execution.

3.3.7 | ATMEGA VS MSP430

During the research of microcontrollers, we found major differences between them which will help us decide what the best solution for our project is. In this section, we will summarize the main differences are between the two microprocessors: MSP430 and the ATMega.

- The ATMega has cross platforms environments; programs can be developed in Linux, Windows, and OSX while on the MSP430 programs can be developed only on Windows environment.
- The MSP430 has a very low consumption of power. It has six different low power modes some of which are important for devices that are powered through batteries.
- The ATMega has wide amount of documentation available for its users since, as it was stated before, its software and hardware are open source. This is also shown in the community support the ATMega has. It also has shields that can be added to the ATMega's PCB to expand any capabilities.
- The MSP430 has code composer studio, which it can be seen as the documentation package of the MSP430 but is not as broad as the community the ATMega has
- The ATMega has a bootloader already installed which helps ease the sketching part by allowing the update of new code without the use of an external hardware.
- The MSP430 is more affordable and more powerful. In terms of hardware components, the following tables present the differences:

	MSP430 Launchpad	ATMEGA-Leonardo
Microcontroller	TI M430G2553	ATmega32u4
Data Bus	16 bit	8 bit
Speed	16 MHz	16 MHz
Storage	16 KB	32 KB
RAM	512 B	2.5 KB
Digital I/O	8 channels	20 channels
Analog I/O	8 channels	12 channels
Cost	$4.30 @ TI.com	$24.99 @makershed.com

Table 5: Hardware comparison of MAP430 Launchpad vs. ATMega-Leonardo

3.4 | SENSORS

By definition, a sensor is a device whose main purpose is to sense any presence or absence of objects. A sensor is a device that converts a physical parameter such as temperature or pressure into an electrical signal, either voltage or current. Sensors are classified into different categories: Temperature, Infrared, Ultraviolet, Touch, and Proximity. These categories are the most commonly used in the industry. They will be explained in this section.

1. Temperature sensors: This type of sensors measure temperature to convert it into information that is easier to understand; for instance, a thermometer is a temperature sensor that gathers or senses temperature and displays it to the user in an understandable and clear way. Contact and non-contact sensors are the two basic types of temperature sensors. Contact sensors require physical contact with the media or device in order to be able to sense its temperature. Non-contact sensors do not need any type of physical contact, which make them good on sensing non-reflective solids and liquids.
2. Infrared (IR) sensors: These sensors sense or measure infrared light which is not detectable to the human eye. This type of sensors detect specific light wavelength through a light sensor. Even though infrared sensors are good for interfacing with other devices, they get easily disturbed by other radiations or lights. Thermography, heating, and meteorology are some of the application where infrared sensors are most used.

3. Ultraviolet (UV) sensors: UV sensors measure ultraviolet radiation which is a form of electromagnetic radiation with wavelengths shorter than visible radiation. Through these sensors is possible to discover any exposure of an environment to ultraviolet radiation. Some of the applications where UV sensors are used include ultraviolet light detector, automobiles, and measurement of the UV spectrum for detecting sunburns in human skin.
4. Touch sensors: Touch sensors are switches that act as a variable resistor when touched. These types of sensors are durable and cost effective, they are mostly used in applications such as fluid level sensors, washer and dryer machines, and transportation. There are three different types of touch sensors: Capacitance, resistance, and piezo. Capacitance sensors change capacitance when touch; resistance touch switch are turn on and off by touch and they last longer than normal switches since they do not have any mechanical parts; finally, piezo touch switch is made of piezo-ceramic, it functions as an actuator and allows touch interfaces between different materials.
5. Proximity sensors: As the name says it, these sensors feel the proximity of objects. These devices have a long functional life, making them highly reliable. Inductive proximity sensors, capacitive proximity sensors, ultrasonic proximity sensors, and photoelectric sensors are they different types of proximity sensors available in the market. They are mostly used in the machine vibration monitoring, window alarms, and automation in production systems.

3.4.1 | Flex Sensors

The most common sensor on our device is the flex sensor. The flex sensor is able to detect changes in bend/flex. It does so by changing its resistance at several points along the device. When a current is applied to this it creates a voltage divider. Since the amount of flex is directly related to the voltage we have an analog signal we can calculate.

Afterwards the development board needs to be able to interpret the signal. The MCU deals with digital signals represented in binary signals. The analog must be treated in a way that its magnitude is treated as a set of digital signals. This is where we need an Analog to Digital Converter (ADC). The ATMega Leonardo already has several pins for analog input. This is a convenience for simulation; we will need to implement only select parts of the Leonardo for the Printed Circuit Board (PCB). For now the Leonardo will be used for simplicity. Make sure the sensor is not bent at the base near the leads; this can damage the sensor and does not affect the output voltage as found in the flexing range of the sensor.

The resistor is to create a voltage divider. If we know what voltage it is originally then we can compare it to the present voltage and infer how much flexion the sensor is experiencing.

3.4.2 | Pressure Sensors

While making sign language gestures there are several points of interest in which contact is made. Subtle differences in hand flexion would make an appropriate interpretation difficult. Therefore if we add pressure sensors to these points we can make a much more accurate interpretation of the hand gesture made. Since the pressure sensor creates a change in voltage with the variable resistance we have a direct relation between the two and data we can measure. This signal is sent out as an analog signal. Below is the schematic for the A201-1 pressure sensor.

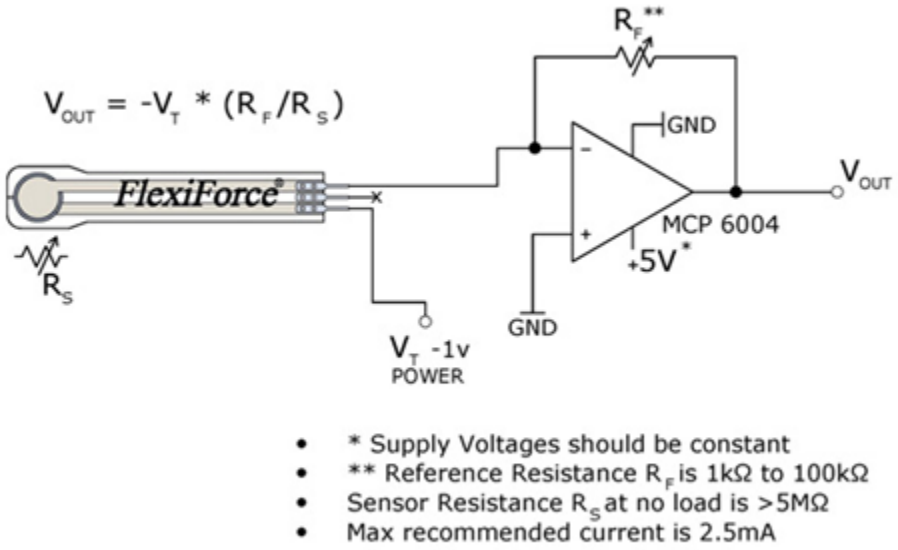


Figure 10: Schematic for Pressure Sensor

The MCU deals with digital signals represented in binary signals. The analog must be treated in a way that its magnitude is treated as a set of digital signals. As with the flex sensor, the pressure sensor must be changed from an analog to a digital signal via an ADC. This is so the MCU can make sense of the voltage changes in terms of binary signals. In this part however, we will use the ATMega analog inputs for convenience. A complete method will be listed later in the Design section while designing the Printed Circuit Board (PCB).

Notice that like the flex sensor the pressure sensor also has a voltage divider at the sensor's output. This is so we have a reference voltage to compare it to. Therefore there is a way to know how much pressure the sensor is under (within its 1 lb limit).

When connecting this via the Atmel processor, we will have to run the analog output to an ADC so we have a digital signal. This will be sent to the General Purpose Input/Output (GPIO) pins.

3.4.3 | Inertial Measurement Unit

Gestures made in sign language are not only dependent upon the flexion of certain fingers, but also the orientation of the hand. To detect position of the hand in 3-D space we need a number of gyroscopes and accelerometers. This capability is very desirable and yet complicated to construct. Therefore the Inertial Measurement Unit (IMU) is sold as a whole part to simplify the work. Below is the bloc diagram for the IC-MPU 6050.

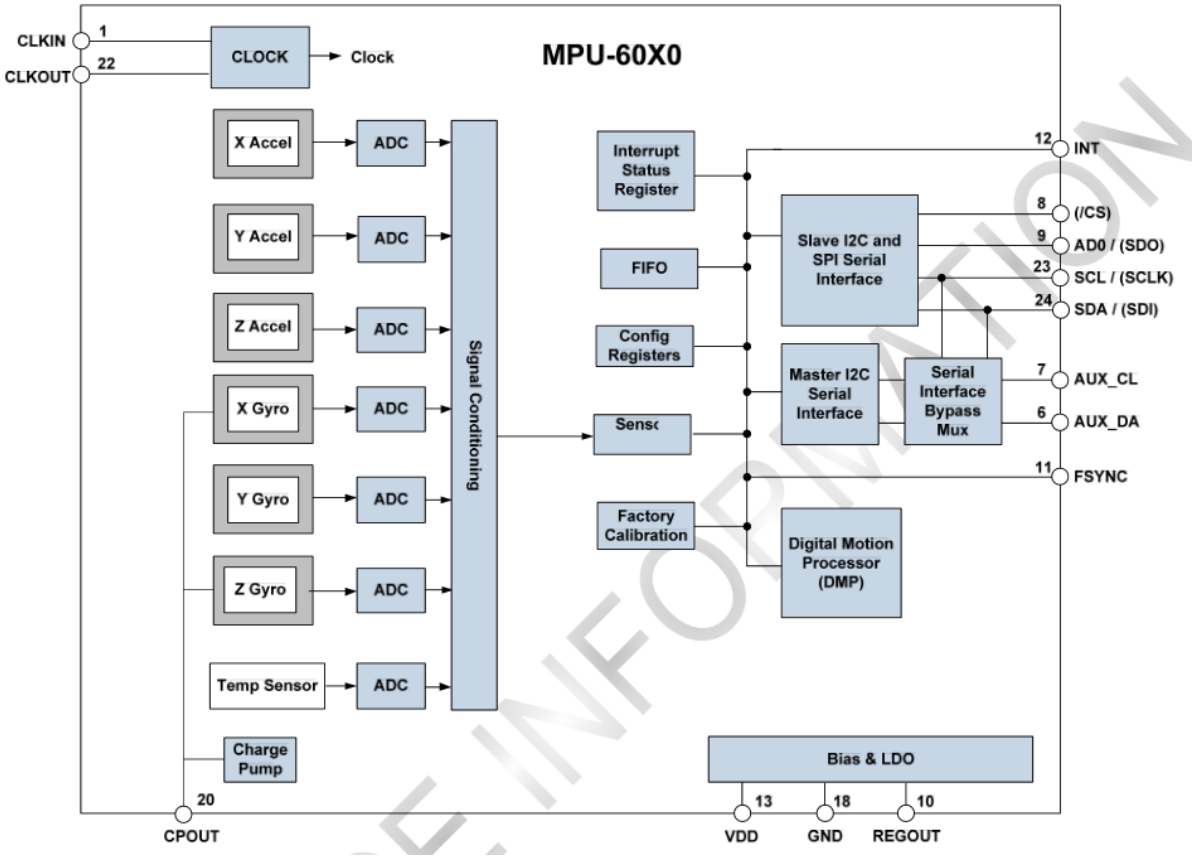


Figure 11: Block diagram of the IC-MPU 6050

From the above figure it is clear that an IMU is a project all of its own. Focus of the project is not the IMU but the data collected from it, therefore it should be purchased instead of running the risk of error. There is no need for an ADC to connect to the ATMega, the IC-MPU 6050 uses the I^2C bus to communicate with the ATMega. The I^2C bus is a serial protocol that utilizes one line that goes in both directions. In order to use the bus, you need to include the Wire.h library

that comes standard with the ATMega. You can then implement the I^2C protocol from the datasheet to get the values needed.

After we send the collected data to the MCU it is then up to the Bluetooth Module to send the information away to the Android device for pattern recognition and let the Machine Learning Algorithms interpret each users various ways of making similar motions.

3.4.4 | I^2C

Inter-Integrated Circuit, also known as I^2C, is a multi-master serial bus interface invented by the Dutch engineering conglomerate Philips. I^2C is a two wire serial bus interface which connects between the master and slave devices. These devices can be a motherboard, cell phone, or a multitude of embedded systems used as a low speed peripheral. The slave devices can then either send or receive information via the two wire interface. A master device starts communication and has the clock, and a slave responds. They come with either a 7 or 10 bit addresses of which some devices can have programmable or predetermined addresses.

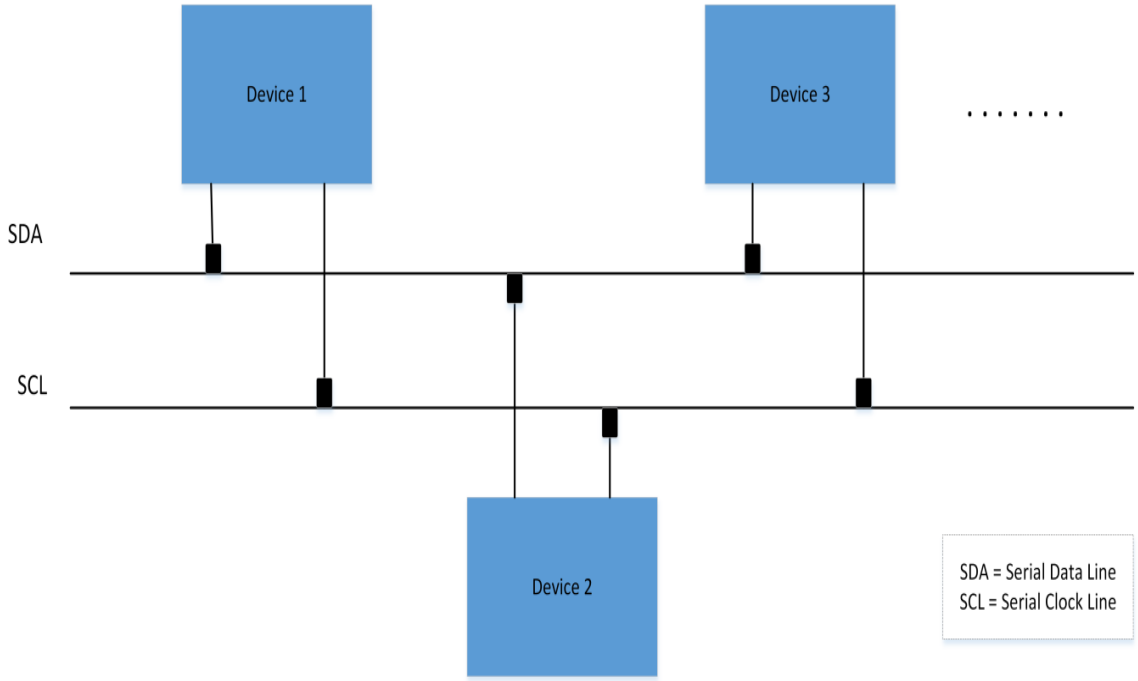


Figure 12: I^2C Serial Clock & Data Line

The bus design allows for which ever number of master devices to be connected. An interesting feature is that I^2C allows for the role swapping of master and slave devices with the appropriate message. This can prove useful when working with certain Bluetooth modules that use I^2C to communicate with the microcontroller.
To start the Master transmits to the desired slave. Since all of the devices are on the same SDA line, the slave with the appropriate address will activate and

respond with and acknowledge the master. The last bit sent indicates whether or not the master will read or write from the slave device. The master then enters write/read mode (whichever is specified) and then the slave transmits/receives data. The master will then either end with a stop bit or will send another start signal to keep control of the bus. Afterward it can start another data transfer with any of the slave devices connected. If not, any of the other devices can become the new master device.
With a limited number of analog inputs and the need for Analog/Digital converters, the I^2C interface proves a valuable tool. Utilizing the feature with the sensors is of a high priority to make the design efficient. The less preferred and greatly discouraged alternative is a 54 GPIO Atmel microcontroller with which half of the pins would be in use. The benefit of I^2C also begs the question of using a smaller microcontroller than the Atmega32u4 since most of the sensors would be moved to the serial interface. The decision was made however to keep the Atmega32u4 microcontroller in favor of using a BLE module without the I^2C capability. This would prove to be too many GPIOs lost to also encompass the remaining pressure and flex sensors.

3.5 | Wireless Communication

3.5.1 | Wi-Fi

Wi-Fi is a wireless network technology which allows an electronic device to exchange data. It is defined as a wireless local area network (WLAN). The design project specifies that wireless communication is desired. Wi-Fi has a few advantages over other methods of wireless communication. Security is generally not an issue as long as the user defines a strong password. Wi-Fi is backwards compatible which would be useful for broadening the target demographic.
Limitations for Wi-Fi set it back though. With a higher frequency it therefore consumes more power. Using Wi-Fi also requires more hardware than other methods. This design, being portable, would be encumbered if it needs to connect to additional hardware. The wireless router is stationary since it needs a much more demanding power supply than other methods. If implemented the device would only be able to communicate if within range of an unprotected Wi-Fi router or one with given permission. This prevents use in new locations in which the users may find themselves away from an available Wi-Fi connection.
One of the most crucial aspects of this design is portability. Other methods of communication used by Android mobile devices are much more suitable in this regard; for this reason we cannot use Wi-Fi.

3.5.2 | Near Field Communication

Near Field Communication is a method of wireless communication which is growing incredibly quick in recent years. It communicates to other devices via radio communication and uses minimal power of the three mentioned methods. In fact “tags” don’t have a power supply; the induced current from the EMF powers it. Many smart phones available have NFC capabilities. Sony devices alone number over 150. One can simply check their cell phone software to see if it is NFC compatible or simply remove the battery door and check to see if the NFC device is there.

Figure 13: NFC on HTC Amaze (grey rectangle in center)

Yet due to the lack of security for NFCs application development has been slow. Many applications involving transactions have been on wait until the issue is resolved. The draw backs to the NFCs build quickly. Low computing power and a much smaller effective range far outweigh the benefit of minimal energy demands. For this reason alone NFC technology cannot be implemented for the glove design.

3.5.3 | Bluetooth

Bluetooth is a wireless communication method for sending data over short distances. After pairing, the devices use a master-slave approach in which said devices can switch roles. Since Bluetooth is practically a mobile device standard at this point in time, interfacing with a cell phone is simplified since only one transceiver is needed.

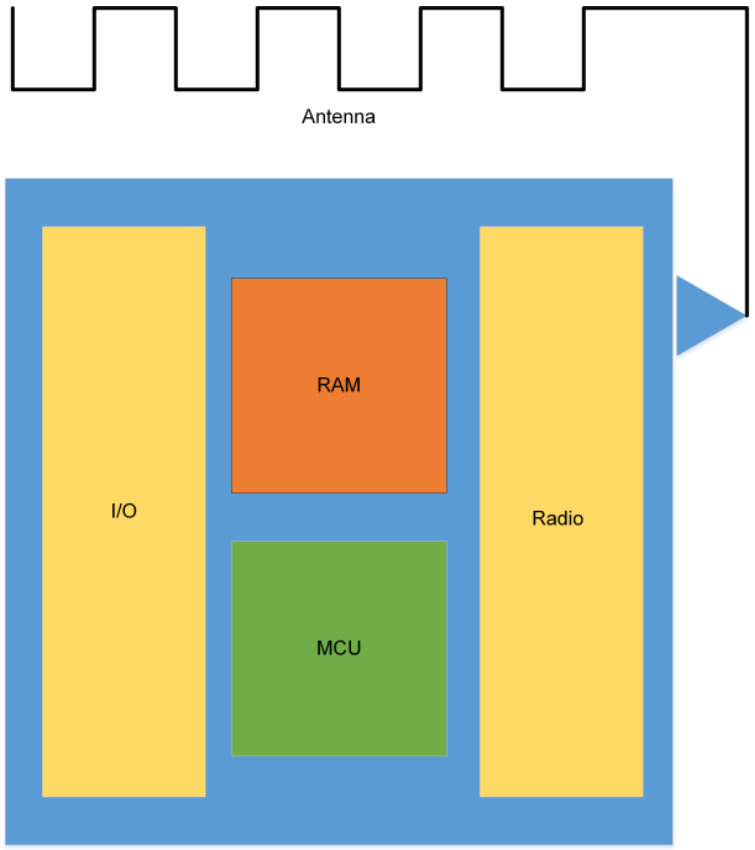


Figure 14: Bluetooth Schematic

There was large problem experienced by many using classical Bluetooth. They may have noticed the large demands it makes on the battery. Often something as simple as pairing a Bluetooth device to play music through a car stereo would require a car charger. In contrast the new BLE technology typically operates at only one tenth of the previous Classic Bluetooth.

Part of the debate over using BLE for the High 6 glove was that the relatively new technology is not particularly well documented in the online community. In contrast, Bluetooth v3.0 modules have been available to the market for quite some time. As a result, the online community has put forth a wealth of information on many scenarios and cases for programming embedded systems with Classic Bluetooth capabilities.

For the sake of efficiency, this new technology is an incredibly valuable feature to the High 6 glove. Therefore, the team decided upon Bluetooth Low Energy (BLE) to be implemented for the wireless communication with the appropriate smart phone with Bluetooth v4.0. Since this project is about device feasibility and application prototyping rather than a production ready device, the group decided to use the new Bluetooth v4.0 Low Energy technology.

The main reason why Bluetooth is the best choice for our project is mobility. Unlike Wi-Fi, which requires a demanding configuration of hardware, Bluetooth is

on mobile phones and uses less power to conserve battery life. Wi-Fi operates over a larger range, yet it also consumes more power as a result. The larger range is not necessary and power management is critical for battery life. Here it seems Wi-Fi would be too much in some regards and too little in others. On the other end of the spectrum is Near Field Communications (NFC) which uses minimal energy. Unfortunately it also has less computing power and a significantly smaller range. Most NFC technologies have an effective range of only several inches.

We want a communication with low energy demands, a large enough range to not interfere with signing gestures, and not be dependent on stationary hardware. For practicality BLE (Bluetooth Low Energy) is to be implemented for this particular design.

	Classic Bluetooth Technology	**Bluetooth low energy Technology**
Distance / Range	100 m (330 ft)	50 m (160 ft)
Over the air data rate	1–3 Mbit/s	1 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.27 Mbit/s
Active slaves	7	Implementation dependent
Latency (from a non-connected state)	Typically 100 ms	6 ms
Total time to send data	100 ms	3 ms, <3 ms
Service discovery	Yes	Yes
Primary use cases	Mobile phones, gaming, headsets, stereo audio streaming, automotive, PCs, security, proximity, healthcare, sports & fitness, etc.	Mobile phones, gaming, PCs, watches, sports and fitness, healthcare, security & proximity, automotive, home electronics, automation, Industrial, etc.

Table 6: Differences between Classic and BLE

BLE is relatively new and has a range half that of Classical Bluetooth. As a result, BLE has a power consumption of about 1/100 to 1/2 relative to classic Bluetooth. Portability is essential to the glove and the more power we need results in a larger power supply. An enlarging battery therefore will eventually over encumber the user.

	NFC	**Bluetooth**	**Bluetooth Low Energy**
Cryptography	No	available	available
Range	< 0.2 m	~100 m (class 1)	~50 m
Frequency	13.56 MHz	2.4–2.5 GHz	2.4–2.5 GHz
Bit rate	424 kbit/s	2.1 Mbit/s	~1.0 Mbit/s
Set-up time	< 0.1 s	< 6 s	< 0.006 s
Power consumption	< 15mA (read)	Varies	< 15 mA (read and transmit)

Table 7: Comparisons between NFC, Bluetooth, and BLE

With a range of 50 meters BLE is more than sufficient for the design requirements. BLE is not backwards compatible to Classic Bluetooth though, so a newer mobile device with Bluetooth v4.0 or higher is needed to pair with the glove. A list of all current mobile devices with Bluetooth v4.0 is shown below.

Windows Phone
• Nokia Lumia
Android
• Samsung (Galaxy S3, Galaxy S4, Galaxy S4 Mini, Note 2 and Note 3)
• LG (Nexus 4, Nexus 5, Optimus G, 4X, G2 and up)
• HTC (One, One Mini, One Max, Desire 300, Desire 601, Desire 500, Butterfly S)

Table 8: Mobile Devices with Bluetooth Low Energy

Apple
• iPhone 4s and later
• iPad 3 and later
• iPod Touch 5
• iPad mini
BlackBerry
• BlackBerry Z10
• BlackBerry Q5, Q10
• BlackBerry Z30

Table 9: Mobile Devices with Bluetooth Low Energy (cont.)

With this list the decision was made to use the Android platform as it is Open Source and permission is not needed to develop the application. Android has a much larger device selection and is more commonly purchased than any other choice of mobile Operating System with BLE. Bluetooth v4.0 is supported on Android v4.3 and later.

Next is the Bluetooth shield, a device which allows the Bluetooth modules to perform serial communication (sending data one bit a time). A Bluetooth shield for BLE is needed. We need a shield that is compatible with our development board, the ATMega Leonardo. The Bluetooth Shield v2.0 from MakerSHED is sufficient for our needs and compatible with the ATMega Leonardo.

A feature added to the 2.0 version is added test points for power consumption measurement of the nRF8001 chip. Added circuitry and test pin (J9 NC) is to be used for measurement of instant working current.

Next we must connect the Bluetooth module to test its connection to the mobile device. To connect the Bluetooth module we will wire it as shown in the diagram on the next page. Add in the resistor to the LED to avoid shorting it out.

The MCU must be programmed via a serial cable. The device cannot be programmed via Bluetooth. Using the Android Software Development Kit (SDK) install the code from Figure 3.5.2.6 which makes the device discoverable but does not exchange information.

3.6 | A/D Converter

One major thing to consider in the design is the number of General Purpose Input/Output (GPIO) pins available on the selected microcontroller. The Atmel32u4 has a total of 20 GPIOs with 12 of those being analog input capable. With the 21 sensor pins required between the accelerometers, gyroscopes, flex and pressure sensors there is a lack of pins available for the Bluetooth module. Therefore an Analog to Digital (A/D) Converter is desired, meaning I^2C will be implemented in the design.
The features available change between model and are discussed below. Though there is a large range of A/D converters on the market the two below are believed to suit the High 6 glove design best.

3.6.1 | ADC081C021

The ADC081C021 is a low power A/D converter which operates at 2.7 to 5.5V. The device is an 8 bit converter, features an alert function, and can take inputs of 11MHz. Since power consumption is a large focus of the overall glove design, having a typical power consumption of 0.26 and 0.78mW while operating at 3 and 5V (respectively) is acceptable. The alert function is turned on when the analog input exceeds the upper and lower limits. Two address lines, ADR0 and ADR1, allow for the device to have programmable address ability for multiple I^2C devices to be implemented.

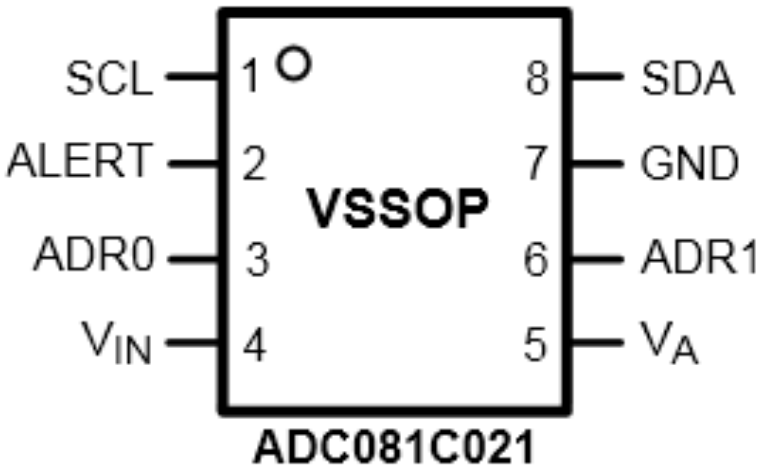


Figure 15: Pin layout for the ADC081C021

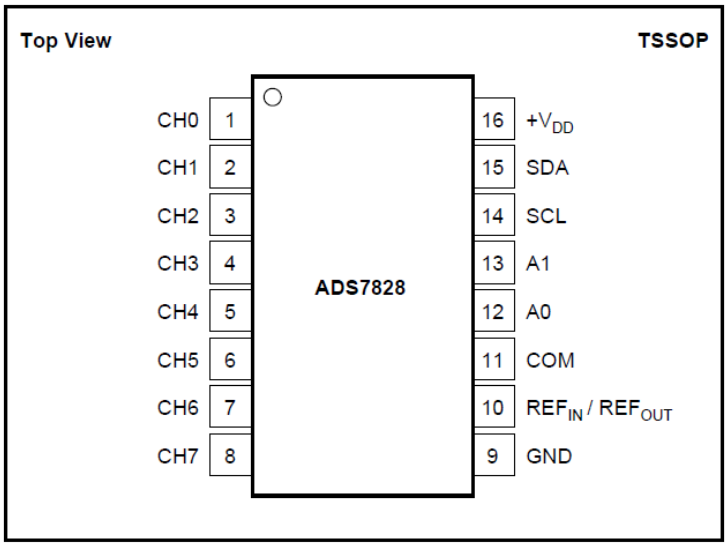
However a drawback is the device has only one analog input for this particular model. Therefore, several of these would be required for the number of sensors. The resolution being 8 bits is also a concern seeing as there are only subtle differences between several hand gestures and the alphabet of American Sign

Language. For this reason a higher resolution for sensor output is desired if possible

3.6.2 | ADS7828

The ADS7828 is an A/D converter with 8 channels, I^2C capability, and 12 bit resolution. It too is a low power device and operates between 2.7 and 5.5V. The higher resolution is desirable as well as the larger number of channels. The A0 and A1 lines indicate the device can be programmed for the addresses of multiple A/D converters of this particular model.

PIN CONFIGURATION



PIN DESCRIPTIONS

PIN	NAME	DESCRIPTION
1	CH0	Analog Input Channel 0
2	CH1	Analog Input Channel 1
3	CH2	Analog Input Channel 2
4	CH3	Analog Input Channel 3
5	CH4	Analog Input Channel 4
6	CH5	Analog Input Channel 5
7	CH6	Analog Input Channel 6
8	CH7	Analog Input Channel 7
9	GND	Analog Ground
10	REF_{IN} / REF_{OUT}	Internal +2.5V Reference, External Reference Input
11	COM	Common to Analog Input Channel
12	A0	Slave Address Bit 0
13	A1	Slave Address Bit 1
14	SCL	Serial Clock
15	SDA	Serial Data
16	$+V_{DD}$	Power Supply, 3.3V Nominal

Figure 16: Pin layout and description of the ADS7828.

Despite the fact that we can only use 4 of these devices, since there are only two address lines, the 8 channels are more than enough to compensate for this. Simply based on the number of channels alone the ADS7828 can be justified for implementation in High 6's glove design.

3.7 | PCB

When it comes to the Printed Circuit Board, there are a number of companies listed that are available to order from. To decide which to go with the team agreed upon a set of specifications. The largest factors being considered are the project budget and the delivery time for the PCB. As rushing delivering a PCB increases the cost, it is advisable to have the printed circuit board ordered early. This also allows for testing with said PCB since part of the design may have been done erroneously.

Due to limits in time and budget, we need to have a printed circuit board designed, submitted, and shipped several weeks before the deadline. This is due to the large increases in cost for expedited shipping. Also, for the sake of aesthetics and functionality, High 6 aims to keep the printed circuit board under a size of 12 square inches. Going beyond the recommended 12 sq. in. runs the risk of compromising wrist movement and therefore threatens the user's ability to make proper American Sign Language gestures.

3.7.1 | Express PCB

Express PCB is an online manufacturer of printed circuit boards and offers free CAD software for drawing schematics and circuit board layout. Since the design needs to be small a 4 layer design is desired to save as much space as possible. The finished product can be shipped in 3 business days and pricing follows a method depending on one of two methods. The preferred proto version starts at a fixed price of $195 and must be 21 square inches or smaller. Our cap on the size of 12 square inches ends up costing an approximate $204 plus shipping. Unfortunately, a more accurate price is not offered until the PCB design is sent to Express PCB first.

3.7.2 | Sunstone

Sunstone provides free CAD software for schematics and circuit board layout. Sunstone's printed circuit board comes in 2 layers but its pricing is much more reasonable. Sunstone's prices for a PCB start at 25 dollars automatically and an additional 3 dollars per square inch. Therefore a desired size of no more than 12 square inches would cost $61.00 at most. This comes at a sacrifice though; the price does not include expedited shipping. Therefore the design must be ordered well ahead of the scheduled deadline. Otherwise it increases to a $134.00 limit

for the expedited option. The considerable cost difference makes the Sunstone fabricator much more favorable over Express PCB.

3.7.3 | Ultimate PCB

After coming across Ultimate PCB we went to find a quote and discovered it too had a relatively high turnout rate. The website had projected a shipping delay of only one business week. This too was a manufacturer that did small orders and customizations. Unfortunately the pricing was an issue; at $165 before shipping and handling expenses it was higher than Sunstone. Yet a positive quality to the company is an 8 layered Flex PCB and Rigid-Flex PCB; this capability may prove to be important for a mobile device which can come across more physically damaging circumstances than other smart glove environments.

3.7.4 | PCB4Less

PCB4Less is a website that offers capabilities in printed circuit board capabilities of up to 30 layers, also in Rigid-Flex Circuit Board, and up to 8 layers of Flex Circuit Board. An interesting service provided is that PCB4Less will test the Circuit Board for quality assurance. This is a great service provided since if anything in the PCB was done incorrectly they will redo the order at that time. The design would even be fabricated and shipped all in one business week. Despite all of these great capabilities and services, a price quote is not offered at the time without the design.

Bearing all of these qualities and capabilities in mind from several different companies, High 6 decided upon Sunstone for PCB fabrication. The limit to two-layer PCB was far outweighed by the large reduction in cost.

3.8 | FPGA

FPGA stands for Field Programmable Gate Arrays. It is an integrated circuit, which is configurable. Its main programmable components are the logic blocks and interconnect resources. The hardware of an FPGA is composed of programmable logic devices, logic gates, RAM and other hardware components such as clock managers. Basically, the FPGA contains the layout of a unit, which is repeated in a matrix form. This infers that the function of each logic block and I/O ports can be configured as well as the interconnection between them. In this section we are going to explain the FPGA architecture, the different FPGA programming technologies, and the different manufactures.

3.8.1 | FPGA ARCHITECTURE

As mentioned before, the basic architecture of an FPGA consists of combinational logic blocks, programmable input/ output ports, and interconnections that are also programmable. As shown in figure #. The combinational logic ports provide the efficiency, performance and ease of design to allow any desired implementation. The input/output ports integrate the package pins and the internal signal lines. The interconnections are in charge of proving routes to connect inputs and output from the combinatorial logic ports and input/output ports to the desired networks.

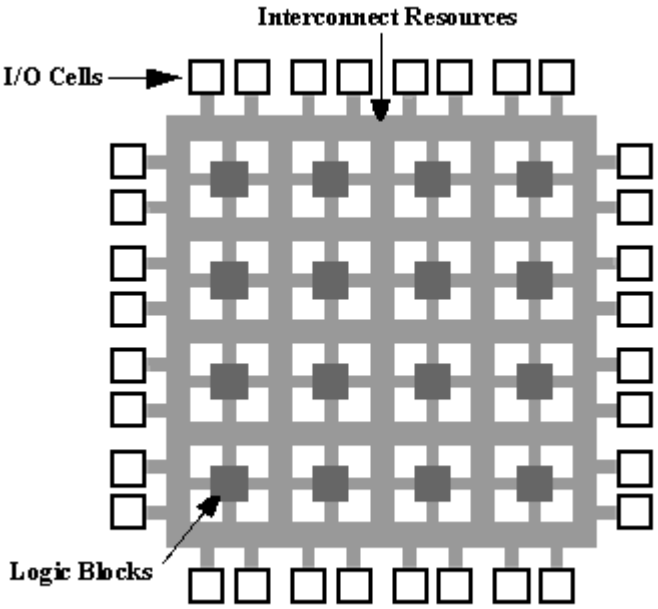
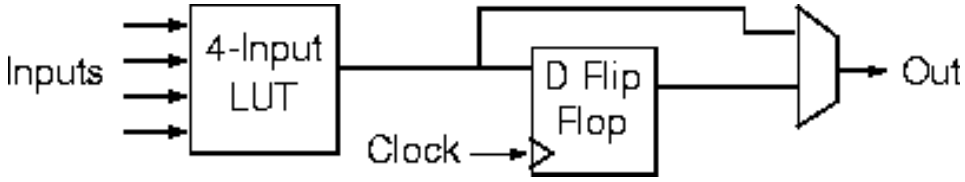


Figure 17: Basic Architecture of FPGA
Figure courtesy of Professor Vaughn Betz

The main purpose of the combinational logic boards is the computation and storage of elements used in digital systems. Transistor pairs, combinational gates, “n” input lookup tables, or multiplexers can implement these logic blocks. For instance, figure # depicts the basic computational logic block which is composed of a 4 input lookup table (LUT), a MUX, and a flip-flop. The main purpose of a LUT is to implement any Boolean function with a set number of inputs.



Figure#. FPGA Logic Block

As mention earlier, the input/output ports or cells connect the external environment with the internal circuits of the FPGA. These ports can the configured as input, output or bidirectional. As shown in the figure above, there are 4 input/output ports that contain logic functionality, which cannot be reassigned. The interconnections, also known as the routing architecture, consist of a variety of wires that have different length intersecting each other at routing switches. These wires' main purpose is to connect combinational logic blocks between each other as well as to connect combinational logic block with input/output ports. There are two types of routing architectures: row based routing and symmetrical routing. On one hand, the row based routing alternates rows of logic blocks and programmable interconnections. On the other hand, the symmetrical routing organizes the combinational logic boards between rows and columns separated one another through the interconnections, these two types of routing architectures are shown in the following figure, figure#.

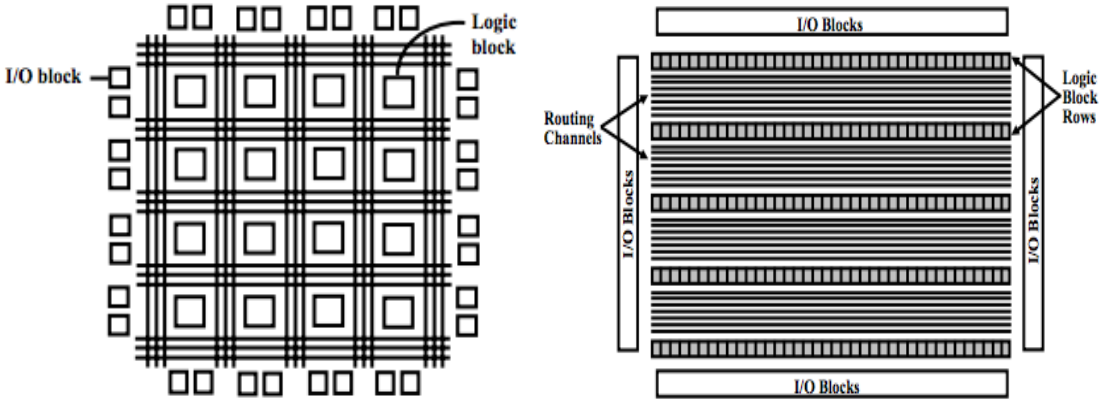


Figure 18: Row Based and Symmetrical Routing Architectures

3.8.2 | FPGA PROGRAMMING TECHNOLOGIES

Programming technologies are used to control the programmable switches that are in the FPGA. There are three different programming technologies available: Antifuse, SRAM, and Flash EPROM

3.8.3 | PROGRAMMING TECHNOLOGIES

1. Antifuse: This technology is based on high resistance under normal circumstances. This means, Antifuse technologies have highest density of interconnections, allowing the efficient usage of smaller logic modules. Since this programming technology does not require silicon area to make connections, it decreases the overhead of programmability, which is its main purpose. Another advantage of this programming technology is the lower resistance and capacitance it has compared to other programming technologies. This last advantage in addition to the low area, allow the

usage of more switches per device. However, this technology is only one time programmable; this means that if any changes are needed, it will not be possible to accomplish them with this technology and therefore the FPGA will not work with the new modifications, this is shown in figure#. This also means that its cost is lower since any additional memory for storing programming information is not needed and it allows the FPGA to be used on operations that need to be powered up immediately.

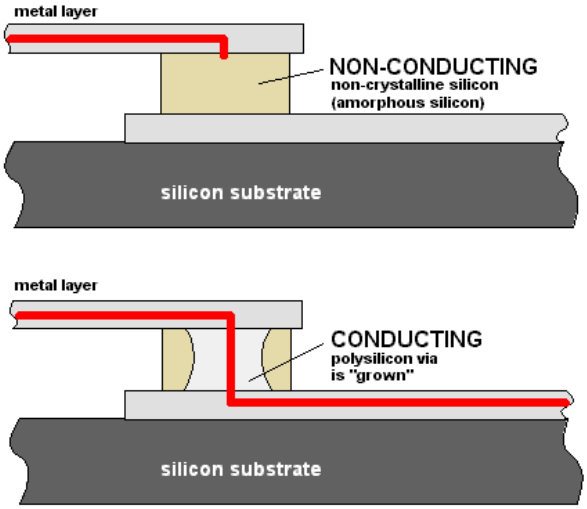


Figure 19: Antifuse Programming Technology

2. SRAM: It is the most common programming technology used for FPGA applications. The main advantage of this type of technologies is its re-programmability. It can also be soldered into the system as well as to change its functionality by modifying only the contents in the PROM. This is possible due to the FPGA circuitry that initializes all the SRAM bits when powering up is occurring. This technology is also used as lookup tables for implementing logic, as well as control to routing and the configuration of switches. Figure#, depicts how the resistance increases between the two wired segments when zero is stored in the SRAM cell. Since this programming technology is re-programmable, its interconnections have high impedance and capacitance which increases the consumption of silicon area compared to other programming technologies. A major drawback is the need to reprogram it when not need it; for instance, every time is turned on. At the same time, an external memory is needed in order to store programs.

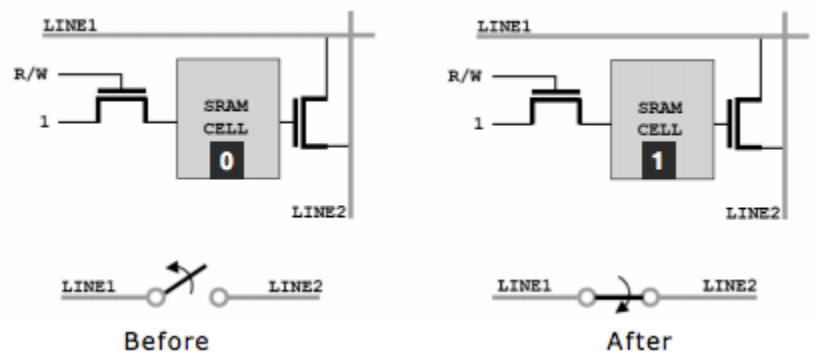


Figure 20: Usage of SRAM Technology

3. EPROM: Since this is a non-volatile programming technology, when powered off it does not lose any information. EPROM programming technology is mostly used for its improved area efficiency. Due to its advantage in non-volatility, there is no need for external resources to store data. This type of device can also function immediately once it's powered up since it does not have to wait for the loading of any configuration data. Moreover, this flash-based technology has more area efficient compared to SRAM. The only downside of this technology compared to SRAM technologies, is the area overhead created due to high and low voltage buffers needed to program the this type of technology. Another disadvantage is that they cannot be reprogrammed infinite multiple times. This occurs because as the charge buildup in the oxide increases it prevents the device from being properly erased and programmed.

3.8.3.1 | SUMMARY OF PROGRAMMING TECHNOLOGIES

The following table illustrates the three different technologies discussed in this section. In the process of selecting a programming technology is important to consider how often and widely is a particular technology that is reprogrammable that uses a CMOS and has a low resistance and capacitance.

Feature	EEPROM	Anti-fuse	SRAM
Reprogrammable	Yes	No	Yes
Reprogramming Speed	Medium		Fast
Volatile	No	No	Yes
External Configuration File	No	No	Yes
Prototyping	Good	Bad	Good
Instant-On	Yes	Yes	No
Security	Good	Good	Poor
Configuration Cell	Medium	Small	Large
Power Consumption	Medium	Low	Medium
Radiation Susceptibility	Low	High	Low

Table 10: Summary of Programming Techniques

3.8.4 | MANUFACTURERS

The graph below, obtained from the article “Top FPGA Companies for 2013” by sourcetech411.com, depicts the FPA manufactures that are dominating the market. By observing the graph is clear that Xilinx, Altera, and Actel are the main FPGA manufactures. Now that is clear which vendors are the best, we will discuss the main leaders and their major differences.

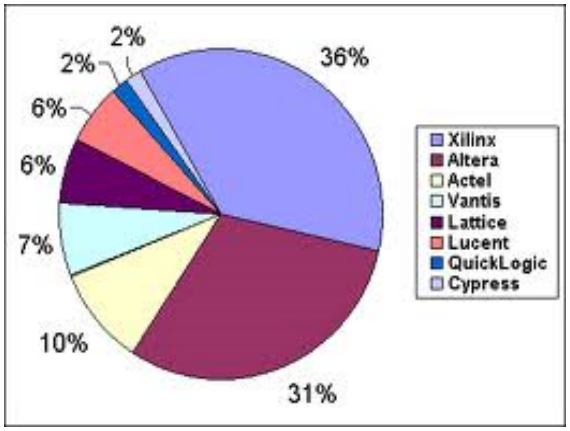


Figure 21: FPGA Manufacturers
Graph courtesy of Source411.com

3.8.4.1 | Xilinx

Being one of the biggest manufacturers in FPGA’s, Xilinx was founded in 1984 and it is the inventor of the FPGA. Nowadays, they control more than half of the

FPGA market. Xilinx have a variety of FPGA families that provide features such as high performance and low power. The table below explains the difference between the families and shows characteristics that should be considered when deciding what type of FPGA best fits our project purposes.

Features	Artix™-7	Kintex™-7	Virtex®-7	Spartan®-6	Virtex-6
Logic Cells	215,000	480,000	2,000,000	150,000	760,000
BlockRAM	13Mb	34Mb	68Mb	4.8Mb	38Mb
DSP Slices	740	1,920	3,600	180	2,016
DSP Performance (symmetric FIR)	930GMACS	2,845GMACS	5,335GMACS	140GMACS	2,419GMACS
Transceiver Count	16	32	96	8	72
Transceiver Speed	6.6Gb/s	12.5Gb/s	28.05Gb/s	3.2Gb/s	11.18Gb/s
Total Transceiver Bandwidth (full duplex)	211Gb/s	800Gb/s	2,784Gb/s	50Gb/s	536Gb/s
Memory Interface (DDR3)	1,066Mb/s	1,866Mb/s	1,866Mb/s	800Mb/s	1,066Mb/s
PCI Express® Interface	x4 Gen2	Gen2x8	Gen3x8	Gen1x1	Gen2x8
Analog Mixed Signal (AMS)/XADC	Yes	Yes	Yes	-	Yes
Configuration AES	Yes	Yes	Yes	Yes	Yes
I/O Pins	500	500	1,200	576	1,200
I/O Voltage	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.5V, 1.8V, 2.5V
EasyPath™ Cost	-	Yes	Yes	-	Yes

Reduction Solution					

Figure 22: Xilinx FPGA Families

Table courtesy of Xilinx, Inc

There are four families in total: The Artix family has the lowest power per logic cell and has a low cost; it has the best performance for DSP and parallel and serial input/output. The next family that offers high DSP ratios is the Kinex family. This family has a 50% reduction in power for half the price. Another family that has high DSP performance and input/output bandwidth is the Virtex family. The Virtex family has a robust serial interface for long lasting channels. The Spartan family is low cost, low power and low risk. According to Xilinx website, this family offers advance power management technology, up to 150K logic cells, integrated PCI Express blocks, advanced memory support and low power transceivers. Paying attention to our project requirements, the Artix family meets all of them.

3.8.4.2 | Altera

Altera offers three different types of FPGA: Cyclone, Arria, and Stratix. Cyclone FPGAs have the lowest cost out the three types. In general, this type has low power and high volume. The next type is Arria, this FPGA provides a balance between performance, power and price needed in any midrange application. Lastly, the Stratix FPGA has the highest bandwidth and density allowing the integration of more functions; this type is ideal for high-end applications. From the observation of each of the types of FPGAs that Altera offers, the Cyclone FPGA will be able to meet out project requirements. There is a total of five generation in this type of FPGA. Each generation has an increased in integration, lower power, and faster time. Out of the five generations, Cyclone III has extended battery life, hot socketing operation support, and routing architecture optimized for design separation.

3.8.5 | FPGA DEVELOPMENT SOFTWARE

Even though all of the FPGA manufacturers provide their own FPGA development software, we would like to look into other development software such as Eagle.

3.8.5.1 | EAGLE PCB SOFTWARE

Eagle is a PCB software that offers a high level functionality of circuitry design. This software can run in Windows, Linux and OS X operating systems. It offers, simulation, as well as data import and export. At the same time, Eagle offers an automated connection to a database to search and find parts from within its own design environment. Furthermore, Eagle enables users to receive quotes and

order prototypes. Eagle's three main modules are: Schematic editor, layout editor, and auto-router.

According to CadSoft, the main features of the schematic editor are:

- Up to 999 sheets per schematic
- Icon preview for sheets
- Sorting sheets with Drag & Drop
- Cross references for nets
- Automatic generation of contact cross references
- Simple copying of parts
- Replace function for parts without loss of consistency between schematic and layout
- Online Forward & Back Annotation between schematic and board
- Automatic generation of supply connections
- Automatic board generation
- Electrical Rule Check (error check in the Schematic and consistency check between Schematic and Layout)
- User Defined Net Classes for Via Size, Wire Width and Clearance

The layout main features are:

- Full SMD support
- Support of Blind and Buried vias
- Rotation of objects in arbitrary angles (0.1degree steps)
- Components can be locked against moving
- Texts can be placed in any orientation
- Dynamic calculation of signal lines while routing the layout
- Magnetic pads function
- Tracks can be drawn with rounded corners in any radius
- Mitering to smooth wire joints
- Design Rule Check for board layouts (checks e.g. overlaps, measures of pads or tracks)
- Copper pouring (ground plains)
- Package variants support
- Differential pair routing
- Meander command for length compensation of signals
- Support of assembly variants
- User definable, free programmable User Language to generate data for mounting machines, test equipments, milling machines or any other data format
- Output of manufacturing data for pen plotters, photo plotters and drilling machines with the CAM Processor

3.9 | POWER SOURCE

As it was established earlier in this document, we want to use a microcontroller that has low power consumption. In this section we will examine the power

components of the ATMega-Leonardo. This microcontroller can be power in two ways: via micro-USB connection or using an external power supply. The external power supply can come from a battery or an AC-DC adapter. If batteries are used, the lead from them can be inserted in the ground and voltage input pins respectively.

It is important to note that if an external power supply is used, the external power supply should not supply less than 7 volts, since the 5V pin used to regulate the power into the microcontroller might regulate less than 5 volts which will make the board behave in an unstable manner. Since we are aiming to develop a low power sign language glove, out of the two ways to power this microcontroller, the usage of batteries is the best option for us.

3.9.1 | BATTERIES

Since the project is portable, it requires the use of batteries to power the microcontroller. We will discuss the different types available in the market that are compatible with the majority of the microcontrollers, the battery's chemistry, and at the end of this section we will show a table that will have the main factors to consider when deciding what battery is the best for our project. The most popular batteries are:

1. **Nickel Cadmium:** These types of rechargeable batteries have low energy density, as well as high charge and discharge cycles. At the same time, they have long life even if stored in a discharged state and they have a good performance in low temperatures. They are mostly used in applications that required long life, and high discharge such as power tools, biomedical equipment, and professional video cameras. Since these types of batteries contain metal that is toxic, they are not environmental friendly. Besides not being environmentally safe, these batteries have a high self-discharge which gives them the need to recharge after storage. They also have a memory effect in which the batteries act as fully charge and stop re-charging while in reality they are not.

2. **Nickel Metal Hydride:** Is a rechargeable battery whose main advantage is its life cycle. Its energy density is high, meaning it can run for a long period of time. Also, NiMH can operate at low level temperatures, as low as -20°C. This type of rechargeable battery is an excellent option for small, lightweight, and portable applications. This battery has 30%-40% more capacity than standard Nickel Cadmium batteries and its second most important advantage is the absence of toxic metals. Unfortunately, it has a limited service life since its discharge reduces the service life. Also, it does not absorb overcharge well and it generates high-load discharge and heat during fast-charge. NiMH biggest drawback is the degradation in performance if it is stored at elevated temperatures.

3. **Lithium Polymer:** These types of batteries are similar to Lithium Ion batteries since they have the same nominal voltage. They differ in the casing, Lithium Polymer batteries do not have the metal casing; instead, they have a flexible material, which encloses its chemicals. Compared to Nickel Metal Hydride batteries, Lithium Polymer batteries are lighter. Moreover, these batteries are more resistant to overcharge and less electrolyte leakage. One serious downside is the required circuit protection to keep current and voltage within safe limits. Furthermore, these types of batteries are expensive to manufacture; they are about 40% higher than Nickel-Cadmium batteries and their metals and chemicals are changing in a continuous basis. Lastly, if they are not store at a low temperature, they age quickly.

4. **Sealed Lead Acid:** The most important advantage of this type of batteries is the ability to mix oxygen and hydrogen in order to create water, as well as the prevention of it. There are different types of Sealed Lead Acid available in the market; the most common are VRLA (valve regulated lead acid), and AGM (absorbent glass mat). These batteries are mostly used in wheelchairs and emergency lighting. The Sealed Lead Acid batteries have a low over voltage potential; this does not allow the batteries to reach their gas generating potential during charge. Since excessive charging produces water depletion, Sealed Lead Acid batteries cannot be charged completely and their charge voltage must be set lower. Moreover, if the temperature of these batteries gets over the allowed temperature threshold, the life of the batteries gets cut in half.

   The following table gather from batteryuniversity.com, table #, summarizes the main factors of each type of battery discussed earlier in this section:

	NiCd	NiMH	Lead Acid	Li-ion	Li-ion polymer
Gravimetric Energy Density (Wh/kg)	45-80	60-120	30-50	110-160	100-130
Internal Resistance (includes peripheral circuits) in mΩ	100 to 200[1] 6V pack	200 to 300[1] 6V pack	<100[1] 12V pack	150 to 250[1] 7.2V pack	200 to 300[1] 7.2V pack
Cycle Life (to 80% of initial capacity)	1500[2]	300 to 500[2,3]	200 to 300[2]	500 to 1000[3]	300 to 500
Fast Charge Time	1h typical	2-4h	8-16h	2-4h	2-4h
Overcharge Tolerance	moderate	low	high	very low	low
Self-discharge / Month (room temperature)	20%[4]	30%[4]	5%	10%[5]	~10%[5]
Cell Voltage (nominal)	1.25V[6]	1.25V[6]	2V	3.6V	3.6V
Load Current - peak - best result	 20C 1C	 5C 0.5C or lower	 5C[7] 0.2C	 >2C 1C or lower	 >2C 1C or lower
Operating Temperature (discharge only)	-40 to 60°C	-20 to 60°C	-20 to 60°C	-20 to 60°C	0 to 60°C
Maintenance Requirement	30 to 60 days	60 to 90 days	3 to 6 months[9]	not req.	not req.
Typical Battery Cost (US$, reference only)	$50 (7.2V)	$60 (7.2V)	$25 (6V)	$100 (7.2V)	$100 (7.2V)
Cost per Cycle (US$)[11]	$0.04	$0.12	$0.10	$0.14	$0.29
Commercial use since	1950	1990	1970	1991	1999

Figure 23: Comparison of most common batteries
Table courtesy of BatteryUniversity.com

3.9.2 | VOLTAGE REGULATOR

A voltage regulator provides a constant value of output voltage from the power supply despite any variations in input voltage or load current. In other words, a voltage regulator minimizes the power consumption, of the microcontroller in our case, and extends the battery life. It is a device that is connected to the output of a power supply to maintain the output voltage at a constant value; if the output voltage varies for any reason, the voltage regulator reacts automatic in order to compensate for change that happened. This regulation and compensation is done through a variable resistor that responds to any changes in the current

passing through it. In general, there are three main types of voltage regulators: series, shunt, and switching voltage regulators.

3.9.2.1 | SERIES VOLTAGE REGULATOR

The series regulator, also known as series pass voltage generator, is a variable resistor between the input and output voltage connected in series with the load. The change of the variable resistor as the voltage across it varies, ensures that the voltage across the load remains the same. The main advantage of this voltage regulator is that the amount of the current drawn is as effective as the one used by the load. It is important to mention that the series voltage regulator does not drawn the current completely even when the load does not require any current.

3.9.2.2 | SHUNT VOLTAGE REGULATOR

With a shunt voltage regulator a load is operated with a resistor in series with the voltage source and the shunt regulator in parallel with the load. The current level is drawn through the series resistor in order to maintain the voltage across the load constant. The load will take some current and the rest will be drain by the shunt voltage regulator. This voltage regulator is designed to not draw any current at a maximum load of current, and at a minimum load of current the voltage regulator passes the current completely; this shows how inefficient the shunt voltage regulator is. Maximum current is drawn from the source regardless of the load current, even when there is no load current.

3.9.2.3 | SWITCHING VOLTAGE REGULATOR

Switching voltage regulators transform power while linear regulators consume power to regulate. Also, these voltage regulators store up energy in a magnetic field and recover the energy when the magnetic field collapses, which make them more efficient than linear voltage regulators. The switching regulator offers higher power conversion efficiency and increase design flexibility such as multiple output voltages, and different polarities can be generated from a single input voltage. One way this voltage regulator controls the average power of a load is by controlling the average voltage to it.

3.9.2.4 | COMPARING REGULATORS

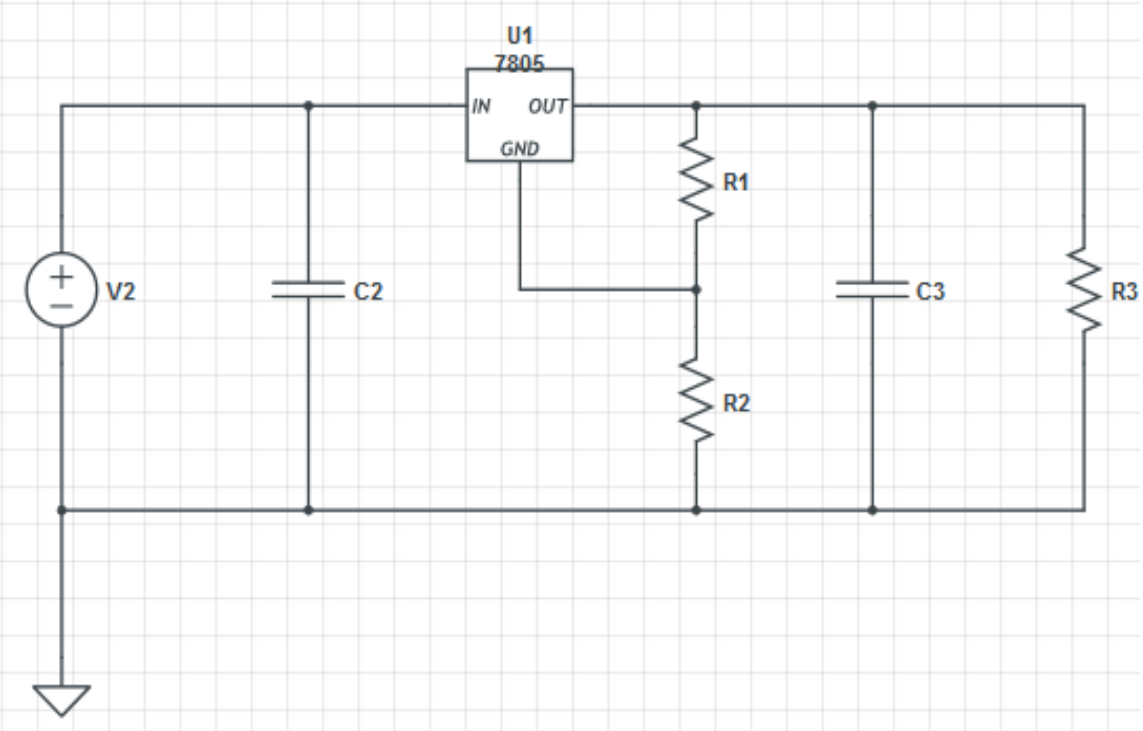


Figure 24: Schematic of a linear adjustable voltage regulator

Due to the linear regulator's inefficiency, and efficiency being a crucial part of a battery powered devices design, the switching regulator is an attractive alternative. The switching regulator achieves this by using the feedback voltage to regulate the Integrated Circuit accepting the input voltage, and large storage elements to allow the output voltage across the load to maintain an acceptable range at the desired value. The LM2576 is a switching regulator that can easily be implemented as a buck (step-down) regulator.

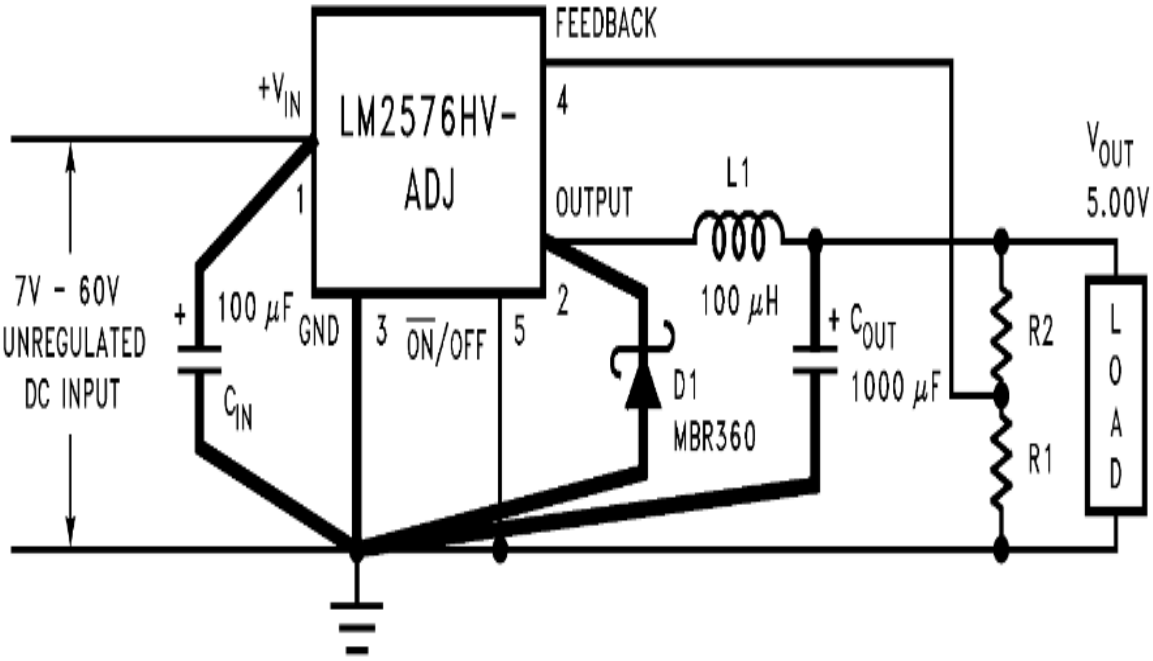


Figure 25: Schematic of the LM2576

A drawback of the switching regulator is the large storage elements used. These values for the elements can only be achieved in a manner that takes a lot of physical space.

3.9.3 | BATTERY CHARGER

For this portable project is very important to take into consideration the circuitry to recharge the batteries that we will use. In this section, we will discuss the charging methodologies available to us and the different charger circuits for the batteries discuss earlier in section 3.8.1.

3.9.3.1 | NiCd and NiMH CHARGER

Since NiCd and NiMH have similar characteristics charging wise, they use the same charging methodology. There are two methods of charging: fast charge and slow charge.

In fast charge, it usually takes about one hour to charge the batteries. This corresponds to a 1.2c charge rate. Fats charge should be only done if the cell temperature is between 10 and 40 °C. If fast charging is done at lower temperatures, it must be done very careful since the pressure of a cold cell rises more quickly during charging, which can cause the releases of gases internally in the battery shorten the life of it. Also, it is important to consider the possible

damage cell damage of fast charging. Due to the creation the gases, the fast charge might be considered hazardous for the user. In the case of NiCd, the gas release is oxygen while for NiMH the gas release is hydrogen, which will burn violently if ignited.

Slow charge is when a charging current can be applied to a charging cell indefinitely without damaging the cell. The main advantage of this method is the charge rate that requires no end-of-charge detection circuitry because it will no damage the battery regardless of how long it is used. Its downside is the long time that takes to charge the cell, which is some cases, might be a negative feature for the user. The NiCd batteries can be recharged in less than six hours without creating any damage to the battery. On the other hand, the NiMH batteries are not tolerant to sustain charging; therefore, when charging is occurring care must be taken in order to not exceed the maximum trickle rate.

3.9.3.2 | Li-ion CHARGER

Li-ion batteries use a constant voltage charging. The constant voltage charging works by sourcing voltage to the battery in an attempt to set the voltage. When the voltage is set, the charge will only source enough current to maintain the voltage in the battery constant. It is important to set the voltage in an accurate manner: if it is too high, the number is charge cycles will be reduced, and this will shorten the battery life. If it is too low, the battery will not charge. The constant changing method takes to segments to produce a full charge. The first segment is the constant current phase; this occurs when the maximum charging current flows into the battery. During this segment, the charge must limit the current to the maximum allow to prevent any damage in the batteries. The following picture depicts the typical charging profile for a Li-ion cell that uses 1c constant voltage charging. This also shows how much current limit and constant voltage is needed for various voltage cells.

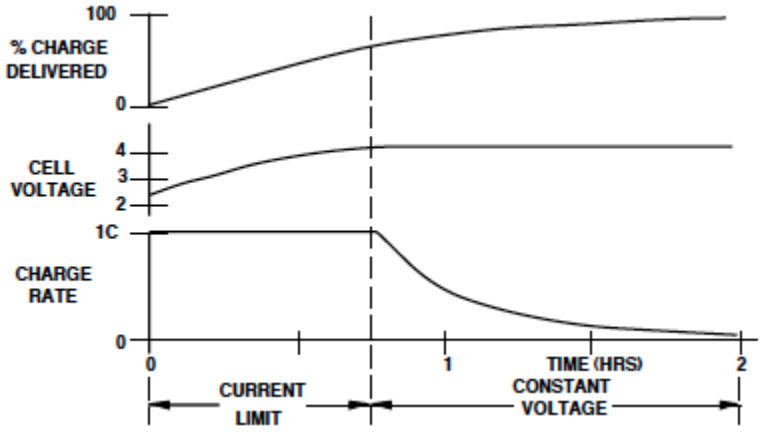


Figure 26: Li-ion Charging Profile

The second segment is the constant voltage phase. This starts when the battery voltage reaches 4.20V when connected to the charger. In this moment, the charger reduces or lowers the charging current and starts to provide the amount needed to keep the battery's voltage constant. This constant decreasing charging current is the main reason why the Li-ion batteries only take about two hours to recharge. Most manufacturers recommend 4.200 ± 50 mV for the set point voltage and 1c as the maximum current that can be used when charging the Li-ion batteries.

3.10 | SOFTWARE COMPONENTS

In the next few sections, all the software components of the project will be discussed. This includes the fact that a mobile app is required, the platforms options that can be used for the mobile applications, and information about machine learning algorithms and methods.

3.10.1 | MOBILE APP

A way to receive data from the sign language glove would be a mobile application. The glove itself will not have any algorithms or processes to determine what letter of the sign language alphabet the user is trying to gesture: the glove itself will detect the hand orientations and positions that the user's hand is in. It will then forward this data over to some platform, and the platform will have the computational power to determine what letters are being represented. Mobile platforms such as Android, iOS, and the windows platforms all have the processing capacity that is needed.

Despite Windows Mobile being an option, we will only look into Android and iOS as mobile platforms. The primary reason is because a critical aspect of this project is that Bluetooth 4.0 is implemented to enable communication between the glove and the phone. Bluetooth 4.0 will allow the glove to have low battery power consumption. The only windows 8 phone to support Bluetooth 4.0 is the Nokia Lumia. Given that this is the only phone to support the critical feature that we need, it means that if there's any other critical feature that this phones happens to not support, then we'd have to redo our project to be implemented onto another mobile platform. At least with the Android OS, and iOS, there are several different phones that support Bluetooth 4.0, giving us more options. This has a much lower risk associated with it. In the following 3 pages, the android mobile platform development options are weighed against the iOS mobile development options.

Also, the native development language for windows mobile applications is Visual C++, which isn't the most ideal programming language when dealing with things that are high level. The reason for this is discussed later on in the paper.

Also, the machine learning algorithm that will get implemented to create this glove will need a sample data set. The bigger the dataset, the better the algorithm is at detecting symbols, because it adaptively learns from new data. We may consider having the mobile phone application collect data anonymously from the phone to give the database more data to learn symbols better.

3.10.2 | MOBILE PLATFORMS

The following 2 sections talks about the 2 mobile platforms that were considered for this project, which were android and iOS.

3.10.2.1 | ANDROID

The Android development platform is a very well supported development platform. Also, Android is written primarily in the Java, object-oriented programming language. All Computer Engineers from UCF are familiar with this programming language since there are 2 courses that use this language, including the algorithms class Computer Science 2. While it is true that android applications use their own custom-made java functions, requiring people who already know java to learn the way android programs are logically structured, we believe that the learning curve wouldn't be as large as the learning curve for Objective C for the iPhone. And also, all the syntax for the android applications are the exact same as java.

To develop in the android programming language, there are 3 integrated development environments (IDE's) to choose from. An IDE is simply the program that a developer develops programs on. If a programmer used notepad to code up a C program, then notepad is their IDE. There are 3 main IDE's to consider when programming an application in Android: Eclipse, Netbeans, and IntelliJ. The 2 that are of primary interest: Eclipse + ADT plug-in, and IntelliJ. Eclipse by itself cannot be used to code applications for Android: the ADT plug-in is a set of software libraries developed by Google to allow development of Google applications in Eclipse. Eclipse is free, as well as the community edition of IntelliJ. There's full featured paid for version of IntelliJ, but that won't be considered as the price is several hundred dollars.

The ADT (Android Development Tool) plug-in has to be downloaded and installed into Eclipse. There are more plug-ins for Eclipse than there are for IntelliJ: there are 1,276 plug-ins in the Eclipse marketplace, and 727 plug-ins in the IntelliJ Plug-in Repository. These 2 numbers are increasing each and every week. This isn't very relevant for the purposes of Android Development since both platforms will have thorough support for Android. However, the number of plug-ins tend to have more of an influence when coding in less ubiquitous types of libraries, which could potentially be a drawback. For example, in researching the algorithm that will be used the pattern recognition of the sign language glove,

the best candidate so far is the machine learning algorithm called the Hidden Markov Method. If there is a java plug in out there to help with the implementation of this algorithm, it would more than likely be found for Eclipse rather than IntelliJ.

In terms of automatic code completion and assistance, IntelliJ is better than Eclipse. The default IntelliJ Code completion and checker is much better and faster than Eclipse, despite some of the code completion and checking plug-ins that Eclipse has. The Usability of IntelliJ is also far more intuitive. Both IDE's have code completion, Dropdowns, project wizards, quick view, and other features that attempt to streamline the coding process and make things easier on the developer; however, the experience of doing it in IntelliJ is more satisfying, more intuitive, and easier to learn.

In terms of performance, IntelliJ and Eclipse are both very taxing on RAM usage, but there are instances where one performs better than the other significantly, and that's because IntelliJ chooses to index everything within the project on startup. Therefore, Eclipse loads projects on startup faster since it only partially indexes. But, once the project is actually started up, IntelliJ runs the project more smoothly. But when working with extremely large projects, IntelliJ won't be able to handle it because the project would be too much to have fully indexed in memory. This isn't a problem for our application because a single android phone application that's meant for translating sign language shouldn't be large enough of a project that indexing the whole thing would eat up several gigabytes of RAM.

Also, when it comes to programming Android applications, a GUI needs to be designed. The GUI builder in IntelliJ runs far more smoothly with no errors, and has a nicer layout than the Eclipse GUI builder. Eclipse is known to have random bugs that occur when designing the UI of an Android application. In terms of software for building GUIs, IntelliJ is a better option.

Based on all the things considered, IntelliJ seems to be a better option for developing our application. However, we may end up using eclipse on the side if it's absolutely necessary to do something. The primary reason that it may be necessary to use eclipse is if another language is needed besides Java, Groovy, or Scala. Those are the only 3 languages that are supported for free by Jetbrains, the company that had created IntelliJ community edition. There is however, plug-ins developed for IntelliJ that have been created by 3rd party companies to support more languages on the community edition of IntelliJ. For example, a vendor by the name of "advancedTools" has created a C/C++ plug-in for IntelliJ, and this plug in can be found on the IntelliJ community edition plug-in repository. The problem with trying to support languages indirectly through plug-ins made by members of the IntelliJ community is that anyone can publish a plug-in, and there's no testing done to make sure that a plug in won't break the IntelliJ system. However, every plug in has people giving the plug-ins ratings, and reviews. Therefore, a developer can tell which plug-ins is likely to be reliable

before they use them. The C/C++ plug-in discussed earlier has a 3.5 star average rating over 28 reviews, each giving large amounts of data about the plug-in. So this isn't a bad approach to get IntelliJ to support other languages than the ones officially supported. To get more languages to be officially functional on IntelliJ, our group would have to acquire a commercial edition which varies in price from $200 to $500, which is far beyond the scope of our budget.

3.10.2.2 | iOS

When considering development for the iPhone, the IDE must be considered. Apple has developed an IDE called Xcode for the purposes of allowing users to create applications for iOS. Like IntelliJ and Eclipse for Android development, Xcode has intellisense to go ahead and suggest corrections for mistakes, and generate code for you. Also, Xcode makes it much easier to transfer an app from iPhone to tablet, whereas it's a bit more of a complicated process for android applications. In general apple tends to have a very independent development environment: they do not have a 3rd party make anything for them. Every single bit of apple development software only interacts with other apple software.

Figure 27: Apple Software Environment

The advantage of having a development environment like the one shown above is that transactions of data from device to device are seamless. The error is minimized down to almost zero. A huge problem with this is the fact that developing for the iPhone requires the developer to use a Macintosh machine with an Intel based processor. This is a massive drawback as 3 members of our group have PC computers, and the only one with a Mac will not be working on the mobile application. Another problem with such a setup is the fact that Apple refuses to use some standard technologies, such as MicroSD cards and NFC (near field communication), and so the only way for a developer to use such technologies is to develop their application for a non-iOS platform. And furthermore, Apple's app store has a tedious and very time consuming processes for getting applications approved.

3.10.3 | Programming Languages

When it came to deciding on which mobile platform we should pick, the programming languages involved was a factor. Windows Mobile development uses C++ as the its development language. C++ can work directly with the hardware layer, and can therefore do things such as allocating and freeing memory directly. Not only that, the compiler for C is designed to convert C++ code straight into machine code that's capable of being run directly on the processor of whatever target processor the developer is aiming for. This is different from Java, and C#, which are the higher level languages that are used for Android and iPhone respectively. Java, and C# do NOT directly convert their high level code into low level machine code. What these 2 languages do is that they have their high level code converted into an intermediate language that's capable of running on a virtual machine. A virtual Machine is a software program that emulates a computer. The virtual machine's processor is different from the target processor the developer wants to run their program on. From this point, another compiler turns the intermediate language machine code that can run on the target processor. This process is demonstrated in the diagram shown below:

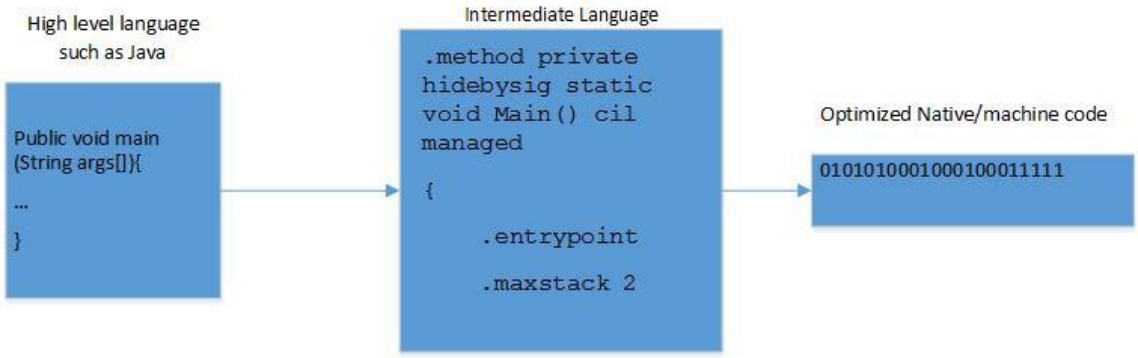


Figure 28: Compilation of high level programming language, Java

From this, it seems that a developer can actually optimize the performance of their application if they use C++ on the mobile development platform of windows mobile. This hypothesis comes from the fact that C++ compiles directly into machine code/native code that can run directly on the native processor that a developer is developing for; whereas java and C# need to first be converted to run on an intermediate virtual processor which differs from the target processor. This however, is a false judgment: C# and Java normally compile to be much faster. There are several reasons for this. One of them is the fact that C++ must be fully compiled before running on a machine, and the compiler can't query the hardware, so the C++ program is optimized to run quickly on machines in general, but it's not optimized to run as quickly as possible on the specific processor that the developer wants to run the program on. It works differently for C# and Java. After these programs have been converted to intermediate language code, the intermediate code is converted to run as efficiently as possible for just the target processor. Why is this possible for C# and java, but

not C++? There's a compiler called the JIT compiler, which compiles from intermediate language to machine language. JIT has the ability to query the target machine and figure out what features a processor has, and doesn't have, and compiles the java/c# program upon execution, the first time a program is run. C++ compilers can't do this, and thus try to mix optimizations to run good on many machines in general without specializing specifically on the target machine. Also, because of the fact that java and C# do not allow programmers to access pointers, the compilers for them are allowed to make certain assumptions about the high level code, that can NOT be made for C++ programs. There are just too many assumptions that are unsafe when the programmer has access to modify and create pointers to memory. Due to this, and other factors, C# and Java generally compile programs that run faster than C++ programs. However, there are optimizations that can be done in C++ that make it outperform java and C# if the programs end up dealing with things close to the hardware layer. This however, isn't a concern for the mobile platform because the mobile platform will not be writing or modifying anything close to the hardware level. Also, another problem with C++ is the fact that since the user has the ability to allocate and free memory via pointers, it creates the possibility that a user can accidentally refer to a freed memory location, or forget to free memory. This is a catastrophic error that can crash the machine. Considering all of these factors, Java and C# are better candidates for the mobile platform, and C++ will most likely not be used.

Apple iOS apps are coded up in Xcode using the language c#. This is good, as c#, like java, is manifestly typed, class-oriented, have runtime compilation, and have garbage collectors. Being manifestly typed means that the programming language does bookkeeping in a sense that if a variable's type is declared as "int", then that variable MUST hold an integer. variables aren't allowed to hold different types of data. The class orientation means that both languages are object based programs. This is a very big advantage when compared to procedural languages. In a procedural language, you can end up with a program that's thousands of lines. And then making a single change means that features all throughout the thousands of lines of code can get messed up. In object oriented languages, a program can be broken down into classes, and instances of those classes are called objects. If a developer needs to make changes to a the program, the changes are usually to a single class that the program is composed of. Changes to a single class do not have the capability to destroy functionality in the other classes. This level of encapsulation is extremely important as it makes code far more adaptable. C# and java also have runtime compilation, which is where high level code is converted to an intermediate language, and then compiled to machine code upon first execution in order to allow best optimization. And both languages have garbage collectors that automatically free memory whenever it is guaranteed that a variable or an object is no longer being used. This prevents a user from accidentally freeing a memory address that they'll later refer to, or forgetting to free memory. Both of which can cause system crashes.

In the end, Java and C# are both very ideal languages to use. The decision to use java over C# is simply due to the fact that android OS uses java and iOS uses C#.

3.10.4 | Machine Learning

One of the core components for this project is the hand gesture detector on an external device. There are two major challenges to this system: learning hand signs, both moving and stationary, and interpreting the hand gestures. The algorithm/equation to provide 100% accurate translations is not going to be found, only approximated based on data. The system needs to “learn” from the data provided to construct an algorithm that approximately models the real life function that maps sign language gestures to letters. This leads us to a particular field of study called machine learning.

Machine learning, a branch of artificial intelligence, allows the system to handle many problems in data mining, vision, speech recognition, and robotics. The more direct approach would prove difficult in such issues where the programmer would address all possible situations or patterns. By giving the system the ability to learn from the data, machine learning could give insight into how the external device of the project may recognize hand gestures.

The theory behind machine learning lies in the use of statistics. Given a sample data set, the system will infer the appropriate output. Because the mathematical model is based on statistics and making inferences, machine learning is not a science, but rather makes predictions through the use of inference algorithms, or learning algorithms. These learning algorithms start in training. To train a computer, it requires optimized algorithms to efficiently store and process large amounts of data. Then, the computer needs the ability to accurately represent the different instances of data. With the ability to handle and represent unseen data, this outlines the core of machine learning. These algorithms discussed in the learning theory vary with the approach developed to address the problem.

The challenges of learning from a given input vary with unsupervised and supervised learning. Depending on the availability of the data, this affects the implementation of the algorithm. With unsupervised learning, the data is never seen by the system, called unlabeled data, and it is within the power of the algorithm to find the hidden structure. On the other hand, supervised learning labels the data from the training set and allows the system to deduce the input with the predicted corresponding value. In the next session, there are different approaches to learning algorithms using these machine learning tasks.

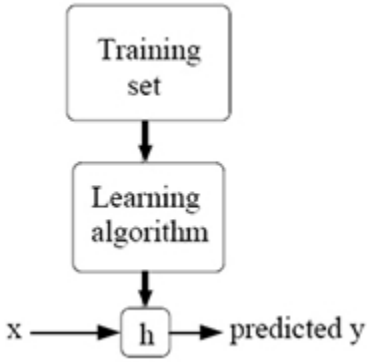


Figure 29: Supervised Learning Logic

Another example of unsupervised learning is when a toddler is deciding whether or not he should touch a hot cup of coffee. When the coffee isn't touched, no pain is felt. When the coffee is touched, sharp pain results. The toddler decides that he's better off not touching the hot cup. So basically, unlike supervised learning, where sample data has the form (input | correct output), the type of unsupervised learning explained above is of the form (input | generated output | grade given to resulting input – output pair).

3.10.4.1 | EFFECT OF PROBABILITY ON MACHINE LEARNING

When it comes to selecting a hypothesis that would be most likely to represent future values, how does one go about doing that? Consider the following scenario:
There in an input vector of 3 coordinates, with each coordinate being either true or false. The table below represents true with '1' and false with '0'. The final output is also true or false. The sample space consists of 5 inputs:

X_n			Y_n
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1

Table 11: Training set of learning algorithm\

Since each coordinate of vector X_n has a domain of just True or False, there are a total of 2^3 combinations of inputs, or 8 inputs. And there are 5 outputs that are known. Therefore, if we define function f as f: x -> y, meaning that f is a function that maps x to y, there are 8 possibilities that f can be. f is the unknown real life target function that the learning algorithm is trying to approximate. The function g is the approximation of f that's being calculated. The table below shows the

domains of X, g, and all 8 possible target functions. The cells shaded in green are from the training set D, and therefore their output values are known.

X_n			Y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1	1	1
1	0	1		?	0	0	0	0	1	1	1	1
1	1	0		?	0	0	1	1	0	0	1	1
1	1	1		?	0	1	0	1	0	1	0	1

Table 12: Possible values of Target Function f

Given that there are 8 possible target functions, it means that g could be any one of them. The whole point of machine learning is to find a function g that approximate f as closely as possible, since real life function f is far too complex of a function to find. But in the above example, there are 8 possible target functions that can be chosen. And since f is unknown, no assumptions can be made about what the output outside the training set D would look like. This might lead one to wonder, if no assumptions can be made about the data outside of the training set D, then how can a hypothesis be selected? Probability ends up providing some assistance.

Consider another scenario: there is a large box of marbles, where some of the marbles are green, and some are red. μ is the fraction of red marbles over all the marbles in the bin. If one was to select an entirely random sample of N marbles from the bin, then v is the number of red marbles in the sample over N. What can be said when comparing μ and v? While it is possible that bin can contain almost all red marbles, and the sample set of N marbles drawn contains mostly green marbles, this is not at all probable. Intuitively, v should be approximately equal to μ. This is in fact true based on the Hoeffding Inequality. This inequality states that for a sample size of N:

$$P(|v- \mu| > \in) \leq 2^{-2N\in^{\wedge}2}$$

Where P(*) is calculating probability, and ∈ is a positive value that gets selected. To explain the equation simply, it states that as the sample N grows, then it becomes exponentially unlikely the probabilities v and μ will be greater than a 'tolerance' ∈. It may seem that μ is a random variable, but it isn't: it is an unknown constant. V on the other hand can change based on the random sample that is selected. So based on this, the value of v gets approximately close to the value of μ as the sample size increases. If ∈ is made into a very small value so that v ends up being very close to μ, then a larger sample size is needed.

Now, does finding μ in the problem above relate to finding a function g that approximates a target function f: x->y? Yes it does, the 2 problems can be connected. From the set of hypothesis H being considered, select a single h∈H from the set. Also, consider every input in the domain of x to be a point. This includes points in the training set whose values are known, and also values outside the training set with unknown values. For each and every hypothesis being considered, h(x) = f(x) at some points, but not at others. The points where this condition is true are green, and the points where this condition is false are red. Since the target function f is unknown, it is also unknown how many points are red, and how many points are green. This sample space is just like the bin of red and green marbles. The portion of the red points over all the points is μ, which is unknown. The hypothesis that is the best is the one that will have the smallest μ, because μ is basically the error rate. Using the training set, the value v can be calculated, and based on the number of samples in the training set, the values of μ can be approximated as v ± ∈. As long as the value μ is approximately 0, then the hypothesis found is a valid one, otherwise it's not.

In the previous example with the truth table, the developer doesn't really have any control over the variable v since v is based off the hypothesis. The example with the truth table was meant simply to help introduce the concept of using the sample space probability to generate information about how good a hypothesis is on data outside the training set. The truth table example isn't really actual learning since only 1 hypothesis was being considered. If there's only 1 hypothesis, then what's actually going on is verification. Real examples like the sign language glove will end up having multiple hypothesis being considered that have some error rate.

3.10.4.2 | MACHINE LEARNING TYPES

In the next 3 sections, there are 3 types of machine learning that are discussed in detail. They include classification, regression, and clustering.

3.10.4.2.1 | CLASSIFICATION

Classifiers are a type of machine learning algorithm. What they do is that they look at input and then categorize them, or classify them. The classifier will use a discriminant, which is a conditional statement which separates input into classes. It's part of the classifier. Classifiers are meant to make predictions on new data assuming new data is very similar to old data. In other words, it has a sample dataset that it learns from, and it assumes that new data is very similar to the old data it has already seen.

An example of when a classifier can make a prediction on new data is when a classifier is trying to interpret human handwriting. When it comes to recognizing hand-written characters, even if a certain character can't be recognized, that

doesn't mean the whole word can't be recognized. The English language has 1 million words in it, which isn't an infinite set of character permutations. Therefore, if a letter is mistranslated, the word can still be figured out. For instance, if the classifier has seen several instances of the word "this" in its database, and the a word scanned in by the classifier is "t?is", the classifier should be able to predict that the word it just read is "this." You can also predict what words are supposed to be by considering the fact that words also follow a certain sentence structure.

An example of a classification algorithm is an algorithm that a bank can use to approve or not approve a customer for a loan. Suppose that each customer has a vector of data called 'x', which contains coordinates such as the age of the customer, the number of timely rent payments, number of late payments, etc. The set of all of these customers is X, so $x \in X$. Y is the set of all possible outputs, so Y = {-1, 1} where 1 means approved and -1 means not approved. A hypothesis that can be used to model the approval equation is the following, where d is the number of coordinates in the customer vector x_i and w_i is the weight that is given to the coordinate.

$$Approve\ loan\ if \sum_{i=1}^{d} w_i x_i > Threshold$$

$$Deny\ loan\ if \sum_{i=1}^{d} w_i x_i < Threshold$$

If the customer's score is above a certain threshold, then the loan is approved. The learning algorithm will take this hypothesis and the learning sample which contains sample data, and use that to figure out what weights and to assign for each coordinate of x, as well as the threshold.

3.10.4.2.2 | REGRESSION

Regression is similar to classification in a sense that both are supervised learning algorithms. A supervised machine learning algorithm is simply a learning algorithm that has been trained to assign an output to some input. So for the classification, the classification algorithm simply assigns a class to the input that it reads. Regression on the other hand, assigns a mathematical function to the input data. The mathematical function's general form is: $y = g(x|\theta)$, where x is an input variable, y is the variable dependant on x, and θ represents unknown function parameters that the regression learning algorithm is trying to figure out. In order for the regression learning algorithm to figure out these parameters, the general form of the equation must be known. Also, the number of sample data points must be greater than or equal to the number of parameters. If there are less data points (X_i, Y_i) then there are parameters to solve, it is impossible to solve for all parameters. If the number of data points exactly equals to the number of parameters to be solved, then a linear function can be assigned to the

sample data, unless the sample data is non linear. In the most common case where there's more data than there are parameters to solve, then values of the vector θ are chosen that best fit as much of the data as possible, so that new future values are likely to be close to what θ is decided to be.

3.10.4.2.3 | CLUSTERING

Unlike classification and regression, clustering is NOT a supervised learning algorithm. In clustering, a set of unlabeled data points are given, and groups of points that are close together are grouped together. Like in the diagram below, where 2 groups of points are relatively close to each other, and are therefore clustered together.

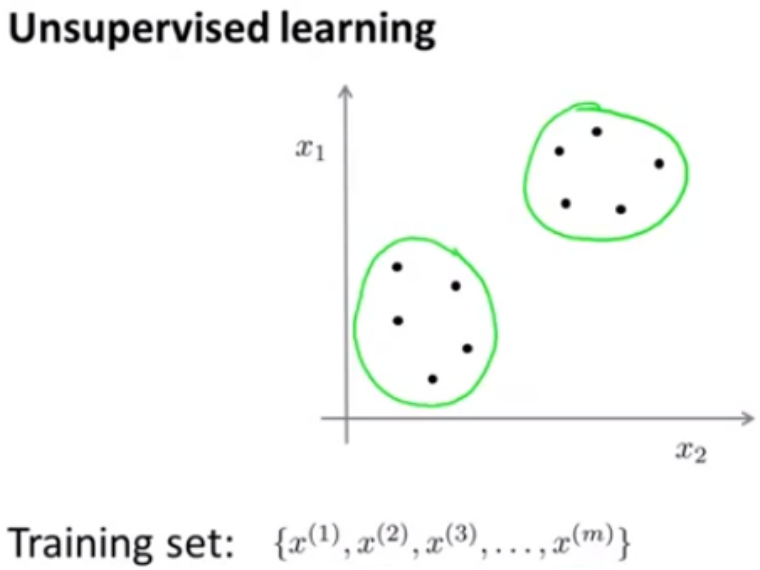


Figure 30: Clustering Algorithm using Unsupervised Learning

3.10.4.3 Hidden Markov Model

One of the problems that were being considered was what to make the timing of the glove. By timing, what is being referred is how long the user gets to make a gesture. For example, at time t= 0, the user may get 1.5 seconds to put in a gesture representing the letter 'A'. And then after that 1.5 second window, after the letter 'A' has been read in and decoded, then the user can input the next character. This method is rather troublesome and inconvenient to the end user, so the idea of using timing to separate the gesture inputs was thrown out the window. A new method needed to be created where a user can continuously input gestures without having to pause and wait for anything. This is where the Hidden Markov Model comes into play. The Hidden Markov model is what allows continuous gesture recognition for complex gestures such as sign language.

There is a paper that detailed the process of creating a program to read in sign language gestures using a camera, and documented the results of their project in a paper called “Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video.” In this paper, the Hidden Markov model is used to interpret the sign language gestures that have been read in through a camera. This paper mentions that surprisingly little detail about the hand is needed in order to interpret the sign language gestures. The only details about the hand that is of interest when trying to decrypt the sign language gestures are the following 3 things: hand shape, orientation, and trajectory. The hand shape is how much certain parts of the hand and fingers are flexed. The hand orientation is what position the hand is rotated in. The hand trajectory is the path that the hand follows, which is only used for some letters such as the letters J and Z, where the person gesturing will actually draw out the letter using their pinky and index finger respectively. Also, the Hidden Markov Method is currently used today in software that is designed to do handwriting recognition.

The 26 letters of the alphabet will be what makes up the “lexicon” of symbols that the Hidden Markov method will try to map the gestures to. Also, every time the user ends up inputting a gesture into their phone using the glove, the gesture is used to update the library so that the library is better at recognizing the letter. An important thing to understand when it comes to Hidden Markov Methods is that it uses the Bayesian Interface. A Bayesian is simply a way to reverse conditional probabilities. If you are given P(B|A), then you can calculate P(A|B) using the Bayesian Interface:

$$P(A|B) = P(B|A) * P(A) / P(B)$$
$$= P(B|A) * P(A) / \sum_A P(B|A) * P(A)$$

To illustrate a further understanding of the Bayesian Interface, consider the following example: a person has been diagnosed with a disease using a test that is 99% accurate. The disease is an extremely rare disease that affects 1 in 10,000 people. Even though the test itself is 99% accurate, does that mean the person most likely has the disease? The answer is no due to the fact that 1 in 10,000 people end up getting infected, and this is proved with Bayesian interface. If T is the test, then T is a random Boolean variable that can be assigned true or false, depending on whether the test comes out true or false. If D means disease, then D is a random Boolean variable that is true if the person really does have the disease, and false if they don’t. Since there is a 99% chance the test comes out true:

P(T = true | D = true) = .99

The above equation means that if the person truly does have the disease, then there is a 99% chance that the test will come out to be true

$$P(T = False \mid D = False) = .99$$

The above equation says that if the person truly doesn't have the disease, then the test has a 99% chance to come out false.

$$P(T = true \mid D = false) = .01,\ P(T = false \mid D = true) = .01$$

The above equation defines the probability that the test is wrong.

Since there's a 1/10000 chance that a person actually does have the disease, then P(D = true) = .0001, and P(D = false) = .9999

So even though the person tested true for the disease, the actual chance they have the disease is:

$$P(D = True \mid T = True) = \frac{P(T = True \mid D = True) * P(D = True)}{P(T = true \mid D = false) * P(D = False) + P(T = true \mid D = False) * P(D = True)}$$

$$P(D = True \mid T = True) = \frac{.99 * .0001}{.01 * .9999 + .99 * .0001} \approx 0.0098$$

So given that a test which has 99% accuracy came out positive, the actual chance of getting the disease is less than 1% since only 1/10000 end up getting the disease.

Now that a background of the Bayesian Interface, it's time to get into details about the Hidden Markov method. In the Hidden Markov method, there are 2 sets of random variables. One set Z, consists of a finite set of integers going from 1 to n integers. So $Z \in \{1,2 \ldots n\}$. The other random variable X consists of n sets of data, where each set of data can be anything… they can be integers, decimal points, complex numbers, whatever. So $X \in$ {discrete numbers, real numbers, higher dimensional real numbers}. These 2 random variables respect the following graph:

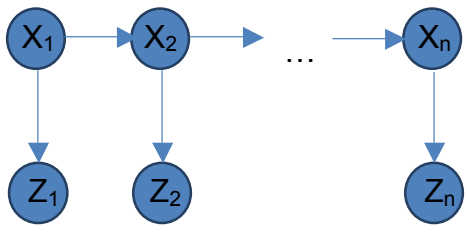


Figure 31: Relation between random variables in HMM

In the above diagram, the X variables are things that can be observed. The Z's are hidden. The subscripts can be seen as discrete units of time, so X_1 occurs at time t = 1 unit, X_2 occurs at time t = 2 units, and so on. To calculate the probability of one of these random variables X or Z, the following equation can be used:

$$P(X_1, X_2 \ \dots \ X_n, Z_1, Z_2 \ \dots \ Z_n) = \ P(Z_1)P(X_1|Z_1)\prod_{k=2}^{n} P(Z_k|Z_{k-1})P(X_k|Z_k)$$

The above equation has different parameters of input, including transition probabilities, and emission probabilities.

All the possible values that the hidden variable z could change into have probabilities that are defined by the transition probabilities, T(i,j). The transition probability is the likelihood that the random variable X will change to another random variable in the set.

$$T(i,j) = P(Z_{K+1} = j|Z_K = i) \ for \ i,j \ \in \{1 \dots n\}$$

The Hidden Markov method in the implementation of the glove will clearly have 26 different X variables for each of the 2 letters. Letter will have to have 25 transition probabilities calculated for it as each letter can transition to any of the other 25 letters. The total number of transition probabilities for n random variables is n(n-1)/2. The reason n(n-1) is being divided by 2 is to get rid of double counting. So the number of transition probabilities that need to be computed is 26(25)/2 = 325 transition probabilities. The transition probabilities serve as the parameter $P(Z_k|Z_{k-1})$.

Emission probability distributions are another parameter and have the following form:

$$\varepsilon_i(x) = P(x|Z_K = i) \ for \ i \in \{1 \dots n\},$$
$$x \in \{\text{discrete numbers, real numbers, higher dimensional real numbers}\}$$

This equation using selecting a particular Z_i and calculating a probability distribution over the entire set of all X. Emission probability distribution ends up being plugged in as the parameter $P(X_k|Z_k)$ and as the parameter $P(X_1|Z_1)$.

The last parameter that needs to be calculated is the initial distribution. This last parameter ends up being plugged in for $P(Z_1)$, and is defined as

$$\pi(i) = P(Z_1 = i) \ for \ i \in \{1 \dots n\}$$

Using these 3 parameters, the equation introduced earlier for calculating the probability of a random variable being selected can be rewritten as follows:

$$P(X_1, X_2 \ \ldots \ X_n, Z_1, Z_2 \ \ldots \ Z_n) = \ P(Z_1)P(X_1|Z_1)\prod_{k=2}^{n} P(Z_k|Z_{k-1})P(X_k|Z_k)$$

$$= \pi(Z_i)\varepsilon_{Z_1}(X_1)\prod_{k=2}^{n} T(Z_{k-1}, Z_k)\varepsilon_{Z_k}(x_k)$$

The key to the Hidden Markov model isn't really the probability distribution of the parameters themselves. The important aspect of it is the fact that the Hidden Markov's probability equation factors in the way defined above. Because of this, the algorithm can be used to do traceable inference. Also, as a side note, the T(i,j) is referred to as the transition matrix. If the Hidden Markov algorithm was applied to a dataset where there random variable Z could only be 2 values (either -1 or 1), and the observed value X could be any real number, then the transition table for such a setup may look something like this:

	-1	1
-1	.98	.02
1	.03	.97

Table 13: Transition Matrix example

So in the above table, this is an example of what a transition matrix may look like. The leftmost column represents what value Z currently is. The top row represents a value that Z can change into. So, if z = -1, there is a .98 probability that it will stay -1, but a .02 chance that it will change to 1. All the probabilities for each row in a transition matrix must add up to 1 in order to be a valid probability distribution.

3.10.4.3.1 | Forward Backward Algorithm using Hidden Markov Models

The Forward Backward algorithm is a particular case of dynamic programming. Dynamic programming is basically a way of avoid time consuming recursive calls by simply using a table in memory to store results of computations, and then basing newer calculations off older ones until an answer is found. For example, if someone was trying to calculate 10!, it can be done either recursively, or with dynamic programming. The dynamic programming approach calculates 1 * 2 and stores the value as in table index 0. Then it multiples table index 0 by 3 and stores the result in table index 1. Then it multiplies table index 1 by 4 and stores the result in table index 2, and so on until the final answer 9! * 10 gets found.

Going back to the HMM, assume that $P(X_k|Z_k)$, $P(X_z|Z_{z-1})$, and $P(Z_1)$ are all known. The forward backward algorithm will use that information in order to compute the following:

$P(Z_k|X_{1:n})$ where $X_{1:n}$ means all X from X_1 to X_n

The equation above figures out the marginal probability of Z for each and every X that is possible. In order to do this, there are 2 parts of the algorithm, a forward part and a backwards part.

The forward part: find $P(Z_k, X_{1:k})$
The backwards part: find $P(X_{(k+1):n}|Z_k)$

In the forward part, the marginal probability of Z_k is found over a subset of all X, which is the subset of X from X_1 to X_k. After this Z_k is found, the next step is to find the marginal probability of all the X's from $X_{(k+1)}$ all the way up to X_n, given Z_k has occurred.

To solve the forward part, understand that : $P(Z_k, X_{1:k})$ is the same as $P(Z_k \cap X_{1:k})$ and also understand that:

$$P(Z_k|X_{1:k}) \propto P(Z_k \cap X_{1:n})$$

If the above equation is difficult to understand, consider the following diagram and example:

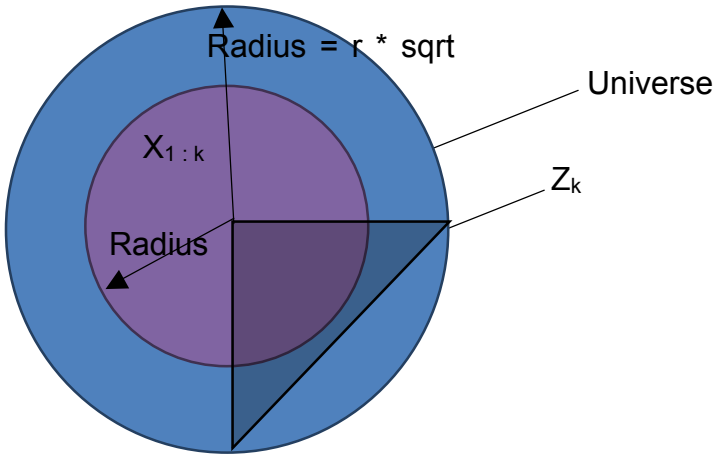


Figure 32: Probability Regions of Forward Backward Algorithm

The outer circle represents all things that are possible, and has a probability of 1. That's why it's called the universe: it's the universe of the probability distribution in discussion. The radius of the circle that represents the universe is r * sqrt(2), where r is just some distance value. If you were to compute the area of the circle representing the universe space, it's pi * (r * sqrt(2)) ^2 = 2 * pi * r^2 = universe area. The inner circle represents the probability of $X_{1:k}$ occuring. The area of this circle is pi * r^2. If you take the area of the inner cirlce and divide it by the area of the universe circle, you get the value ½. Therefore, the probability of $X_{1:k}$ occuring is ½. Also, since the triangle representing the probability of Z_k happening

overlaps ¼ of the inner circle, then the probability of both Z_k occuring and $X_{1:k}$ also occuring is ½ * ¼ = 1/8. So $P(Z_k \cap X_{1:k}) = 1/8$. If $X_{1:k}$ has occurred, then the likelihood that Z_k has also occurred is ¼. In other words $P(Z_k|X_{1:k}) = ¼$. Consider what happens if the triangle representign Z_k was cut in half such that $P(Z_k \cap X_{1:k})$ now equals 1/8. If that's the case, then $P(Z_k|X_{1:k})$ now equals 1/16. Notice that when cutting the probability $P(Z_k \cap X_{1:k})$ in half, $P(Z_k|X_{1:k})$ also gets cut in half. So, the probabilities $P(Z_k|X_{1:k}) \propto P(Z_k \cap X_{1:n})$ because by definition, 2 quanties are proportional if changing one of them by a factor causes a change in the other one by the exact same factor. However, this proportionality will only hold if it's just the triangle, or Z_k gets changed. If the inner circle is what gets modified, then the relationship isn't proportional anymore.

Using that information, another equation can be generated:

$$P(Z_k|X_{1:k}) \propto P(Z_k \cap X_{1:n}) = P(X_{(k+1):n}|Z_k \cap X_{1:k})P(Z_k \cap X_{1:k})$$

An important thing to note is that Z_k is conditionally independent of $X_{1:k}$, and so therefore, one of the unions with $X_{1:k}$ is redudant and the equation simplifies down to:

$$P(Z_k|X_{1:k}) \propto P(Z_k \cap X_{1:n}) = \underbrace{P(X_{(k+1):n}|Z_k)}_{\text{Backward part}}\underbrace{P(Z_k \cap X_{1:k})}_{\text{Forward part}}$$

The forward and backward parts basically multiply to give a value that is proportional to the value that is of interest. Once $P(Z_k \cap X_{1:n})$ is obtained, then obtraining $P(Z_k|X_{1:k})$ isn't difficult because this probability is over a finite set. This means that by summing over the finite set, the constant that you can multiply with $P(Z_k \cap X_{1:n})$ can be found such that $c * P(Z_k \cap X_{1:n}) = P(Z_k|X_{1:k})$

Once $P(Z_k|X_{1:k})$ is found, this gives us the ability to do inference. This is where we can determine what Z is based on the data X that is collected. So this is the point where the gesture recognition is estabilished. In mathematical terms, this means:

$$P(Z_k \neq Z_{k+1}|X_{1:n})$$

3.10.4.4 | LIBRARIES & TECHNOLOGIES

With the use of machine learning, the learning algorithms will likely use statistical computations and involve matrices and linear algebra. There are an array of resources that support supervised learning, classification, clustering, regression, and other approaches to machine learning. Research into existent libraries and tools could help reduce the workload in the classification and training necessary in the later development.

3.10.4.4.1 | Weka3

Weka3 is a popular Java tool for machine learning offering a collection of visual models and predictive modeling algorithms for data analysis. This software suite claims to be easy, portable across all operating systems, free under the General Public License, and even advertises an open online course. It is well into their development with the stable release of version 3.6.10. This tool is a software application that is more suitable for business users than consumers.

Through their graphical user interface, users could create mathematical models to analyze data. This application does not meet the interest of the project's end user. With the translation tool the glove and mobile application provides, the end user does not actively calculate mathematical models; the machine learning algorithm is a black box and used only as a service. The software designers of the group also seek a usable library to develop the learning algorithm and ultimately, the translation tool. The Weka3 toolkit does not suit both our user or meet the needs of the group.

3.10.4.4.2 | Octave

Octave is a high-level interpreted language, or scripting language, and is free software. Programs written in this environment allows the executable to be interrupted, and tasks may run one-by-one. The language is written in C++ as well as uses their standard C++ libraries. This language is a popular choice for the development of machine language concepts.

This scripted language focuses on numerical computations, suited for machine learning. With the use of C++, the team possesses prior experience with C-languages, and hence, there is a high familiarity with the language structure and syntax. It runs on most platforms including, Windows, Mac OS X, and most Unix operating systems. Octave proves to be a promising language to develop learning algorithms.

3.10.4.4.3 | PyML

PyML is an object-oriented framework written in Python, focusing on kernel-methods in machine learning. Kernel-methods are a type of pattern analysis algorithms, most known for support vector machines (SVM). SVMs prove useful for text, images, and sequential data.

This language may not prove suitable for the project. PyML is more practical for statistical analysis while the project scheme seeks pattern recognition algorithm. Python also may limit the project as it has been known to face issues with scalability, a key factor in machine learning. The PyML framework may be more suited towards proof of concept but is not recommendable for pattern recognition.

3.11 | UNIT TESTING

Unit testing is a recommended practice for software development to ensure the quality of code. This serves as an internal control, common with quality assurance teams in companies, when there are changes to the features. Typically, programmers may design the cases and business logic behind a class and expect their approach to address the problem was correct. More often than not, a hiccup may exist in their design. With unit testing, the programmer would invest time in thinking how to use the class. In this process, there would be a clear outline to various aspect of the program depending on the requirements of the assignment. Through this practice, developers may determine if their class is functional and compatible with other classes but most importantly, work as originally intended.

3.11.1 | JUnit

JUnit is an open source framework with the primary focus of executing test cases for Java programs. The technology helps maintain quality by bringing testing into the development of modules. The new test class uses the existing class and can check their main methods. With a single line of code, it asserts the true result and checks it against the return value from the existing class.

```
@Test
public void testMultiply() {

   // MyClass is tested
   MyClass tester = new MyClass();

   // check if multiply(10,5) returns 50
   assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
}
```

Figure 33: Sample Code of JUnit Test Class

The result of the one test is not defined by pass or fail, but rather interpreted print statements. The returned statement depends on the asserted test case. Then, the developer must infer if the class worked accordingly, or adjustments are needed. Sometimes, the unit tests are limited because of accessibility within the classes. In the case where methods or members are private or protected, the JUnit cannot verify their results. After a run through the test cases, the developer can analyze the results for false cases. Of those that are false, the JUnit reveals an issue with the existing class. Hence, JUnits may save time debugging classes and promote reliability of the code.

4 | DESIGN

In order to begin with our design, we have to understand the implementation of each hand gesture. Especially, we need to learn what fingers are used in each hand gesture to not only be able to reproduce the hand gestures but also to translate them into voice and text. In the following lists we show each hand gesture of the American Sign Language Alphabet , as well as a description of each hand gesture. We decided to use descriptions as a way of differentiating each hand gesture.

Figure 34: Enumeration of fingers

Also, we decided to enumerate the fingers as F1-F5, as shown in figure#, being F1 the thumb and F5 the pinky. Using this enumeration will easy the identification of each hand gesture. The first list contains numbers from 0-9, each number has its own distinction, as well as gesture.

Number	Distinction	Gesture
0	F1-F5 partial flex, contact with F1	
1	F3-F5 flexed, F1 partial flex	
2	F4-F5 flexed, F1 partial flexion, no contact in between F2 and F3	
3	F4-F5 flexed	
4	F1 partial flexion	
5	F1-F5 not flexed	
6	F5 flexed, F1 partial flexion	
7	F1 and F4 are in contact, each has partial flexion	

8	F1 and F3 are in contact, each has partial flexion	
9	F1 is in contact F2 each has partial flexion	

Table 14: Distinct Feature Table of Hand Gestures

The second list contains the 26 letters of the alphabet; just like the list of numbers, this list has each character, the distinction and gesture.

Character	Distinction	Gesture
A	F2-F5 flexed	
B	F1 flexed	
C	F1-F5 partial flexion	
D	F1, F3-F5 partial flexion, contact of F1 with F3 and F4	
E	F1-F5 flexed	
F	F2 flexed	
G	F3-F5 flexed, hand oriented to the side	
H	F1, F4, F5 flexed, hand oriented to the side	
I	F1-F4 flexed	
J	F1-F4 flexed with parabolic movement	
K	F4-F5 flexed, F1 partial flexion	
L	F3-F5 flexed	
M	F5 flexed, F1-F4 partial flexion	
N	F4-F5 flexed, F1-F3 partial flexion	

O	F1-F5 partial flex, F2 is in contact with F1	
P	F4-F5 flexed, F3 partial flexion having contact with F1, hand oriented to the side	
Q	F3-F5 flexed, hand oriented to the side	
R	F4-F5 flexed, F1 partial flexion, F2 is in contact with F3	
S	F1-F5 flexed	
T	F2-F5 flexed, contact of F1 with F3	
U	F4-F5 flexed, F1 partial flexion	
V	F4-F5 flexed, F1 partial flexion, no contact in between F2 and F3	
W	F5 flexed, F1 partial flexion	
X	F1, F3-F5 flexed, F2 partial flexion	
Y	F2-F4 flexed	
Z	F1 partial flexion, F3-F5 flexed, Z hand movement	

Table 15: Distinct Feature Table of Hand Gestures

4.1 | HARDWARE COMPONENT SELECTION

4.1.1 | Bluetooth Low Energy

One way to use a Bluetooth module is through a Bluetooth shield. This accepts the Bluetooth module and allows for an interface between the microcontroller and the Bluetooth transceiver. This method is much more straight-forward than using I^2C or the standard GPIOs on the microcontroller and allows serial communication. However, for the needs of the High 6, a drawback of the Bluetooth shield is its large size.

A key decision was that the smart glove must be functional and not compromise the hand gestures made in American Sign Language. To do this the size of the device must be small and the plethora of sensors should not inhibit motions made by the user's hand. Therefore a Bluetooth shield is inadvisable for the High 6 project due to its relatively large size.

The BLE mini produced by Red Bear Labs is a Bluetooth Low Energy transceiver and nearly all of the necessary hardware for communication is assembled by the manufacturer. This means all that is left is to connect the ports for serial communication to the microcontroller. The Red Bear Labs does not make the transceiver themselves though; the BLE module uses a Texas Instruments (TI) CC2540.

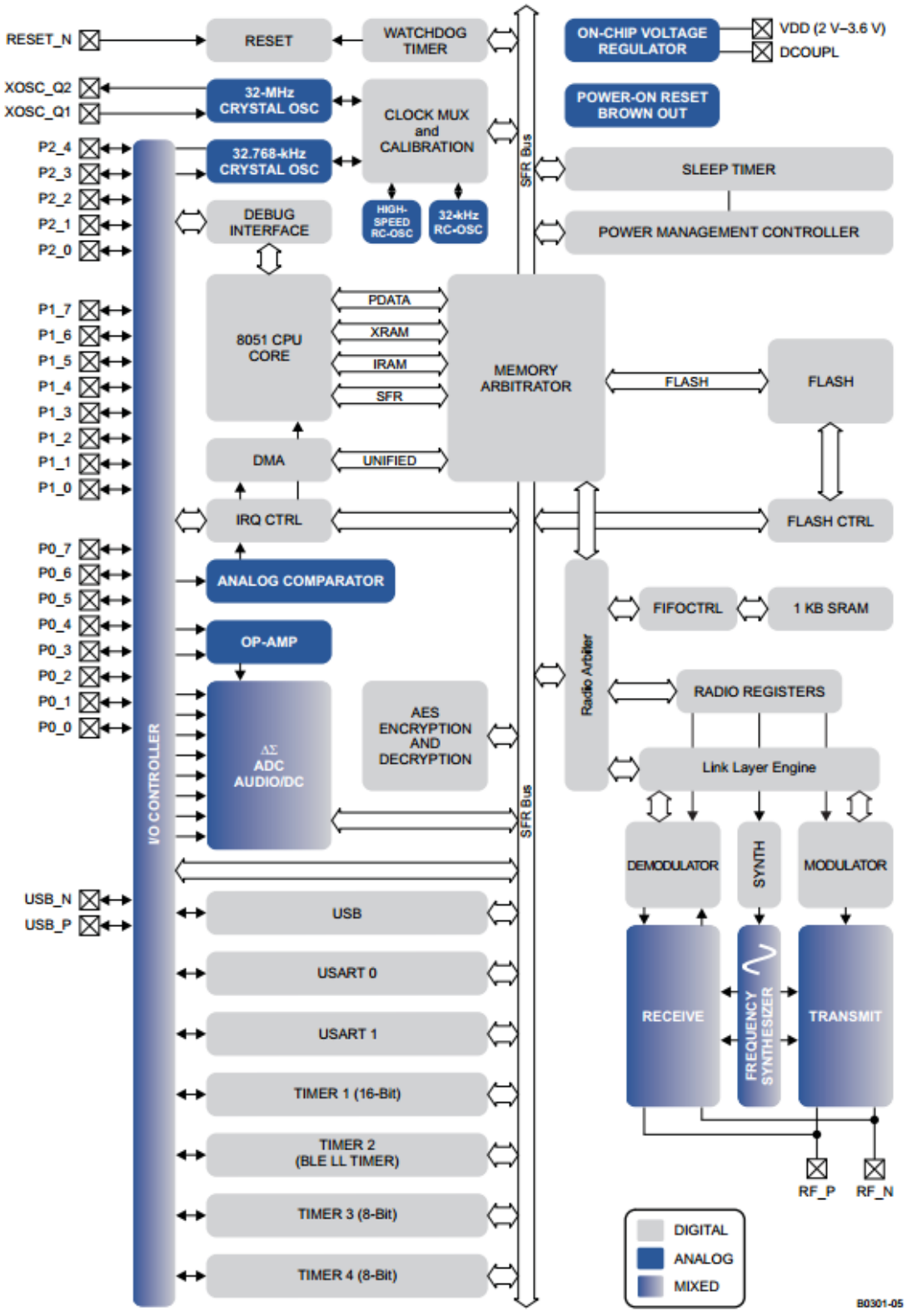


Figure 35: Block Diagram of a general Bluetooth transceiver.

The CC2540 does not include I^2C, but a similar product produced by Texas Instruments does. The CC2541 is a sister product to the CC2540, it too is a BLE module made by TI. The CC2541also has the option of 128 KB or 256 KB Flash RAM. Despite the added I^2C feature, the CC2541 has its drawbacks. The device features do not include a Universal Serial Bus (USB) port or an Operational Amplifier. Combined with the lack of small quantity distribution, the CC2540 is the BLE module that will be implemented for the design.

The Texas Instruments CC2540 Bluetooth Low Energy System on Chip (SoC) transceiver is the module most desired for the High 6 smart glove. This module features a USB port, Analog/Digital Converter, Operational Amplifier, and an Analog Comparator. The option of purchasing either a 128 or 256KB Flash RAM is available.

All embedded software developed for the CC2540 is done using Embedded Workbench for 8051. This is important to take note of since the workbench is purchased or only offers a 30 day free trial. It is important to take note of this since many other classical Bluetooth transceivers' are free.

When the device first starts up there needs to be an established device to connect to. The microcontroller will tell the BLE transceiver to find an appropriate device to pair with. This is done through a "handshake". In Bluetooth Low Energy protocols, the peripheral device will send out a signal called an "advertisement" indicating it wishes to connect to a central device. The glove will act as a peripheral device to the smart phone. After a connection is established a callback signal is sent to the central device acknowledging that the devices are indeed paired.

At this point the roles change and the glove will become a GATT Server while the smart phone will become the GATT Client. At this point the glove/phone relationship will resemble the master/slave interface mentioned earlier. The sole purpose of the glove is to transmit the collected sensor data to the phone for translation.

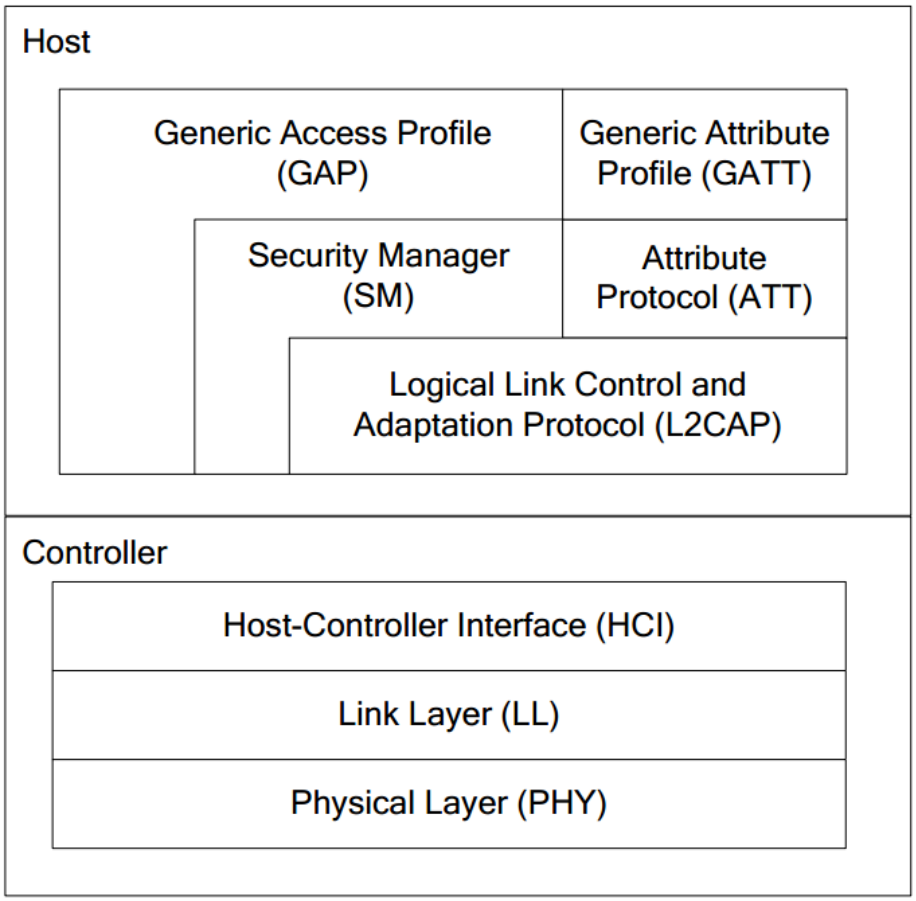


Figure 36: BLE Protocol Stack

For our design we will implement the glove in a single device configuration rather than the network processor. Single device configuration is the easiest and most common configuration when using the CC2540. This is the configuration that most projects use so it will be not only the most logical, but also the most beneficial to our research and hints at more reference material in comparison. It also provides the lowest power consumption making it the most appealing for our battery operated device. The configuration can be seen in the figure below.

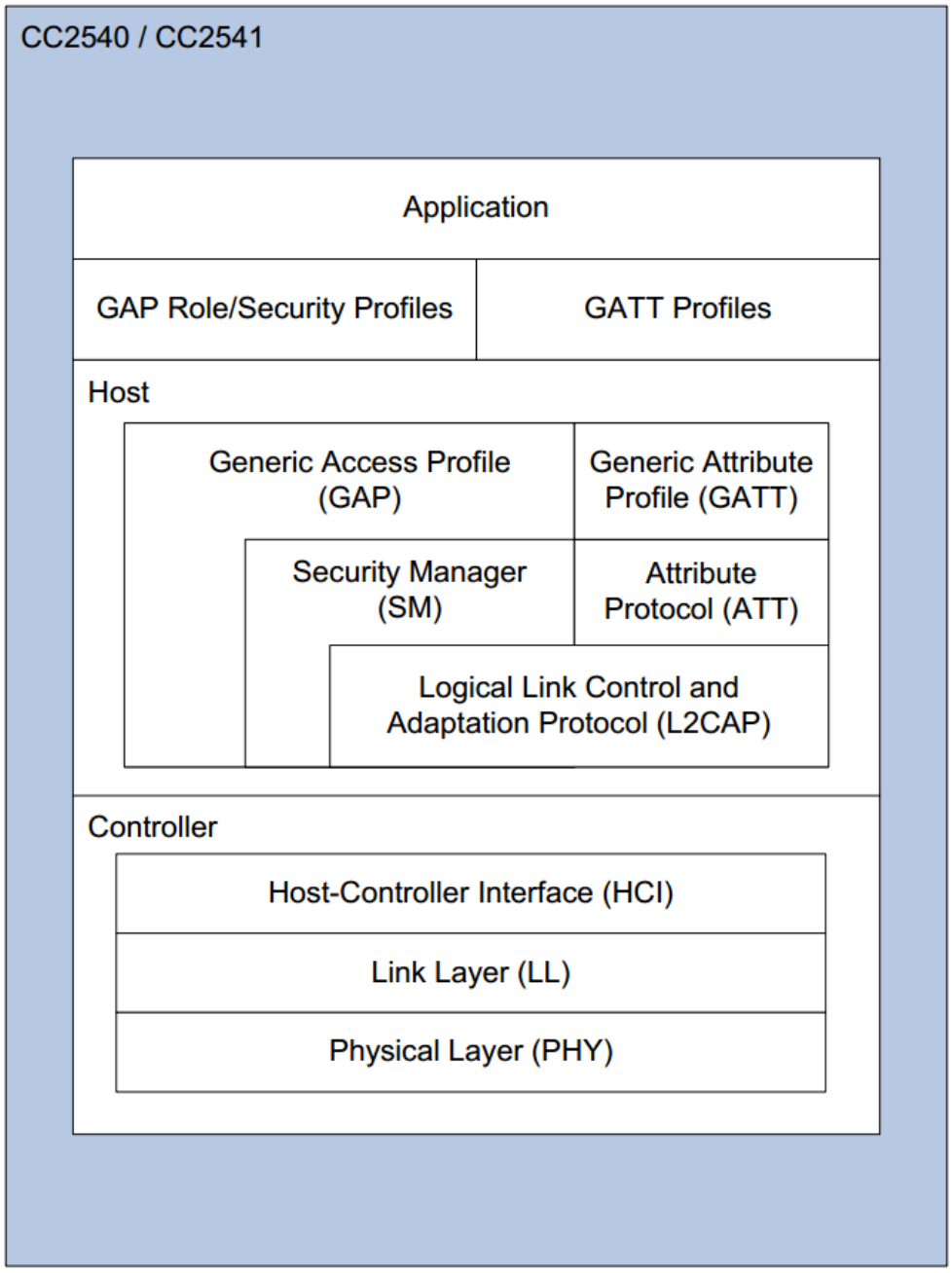


Figure 37: Configuration of the single device BLE application

Another interesting feature is that if there is any slave latency the slave device has the option of skipping a number of connection acknowledgements. This gives the glove an extended battery life since if it does not have any data to send it can choose to skip the Bluetooth commands and go into sleep mode. This feature provides a small bit of added use time to the glove. The decision of when to acknowledge is left up to the slave.

4.1.2 | SENSORS

As mention earlier, this project requires flex sensors, pressure sensors and a gyro/accelerometer in order to be able to get an accurate reading and interpretation of the hand gestures. In this section we will discuss the specific sensors we selected for our project. For flex sensors, we decided to use the 2.2" flex sensor "SEN-10264" from sparkfun.com.
For flex sensors, we decided to use the 2.2" flex sensor "**SEN-10264**" from sparkfun.com. According to its data sheet and shown in the following picture, this flex sensor has a flat resistance of 25k Ohms, a resistance tolerance of ±30%, a power rating of 0.50 watts continuous, and a bend resistance range from 45k to 125k Ohms.

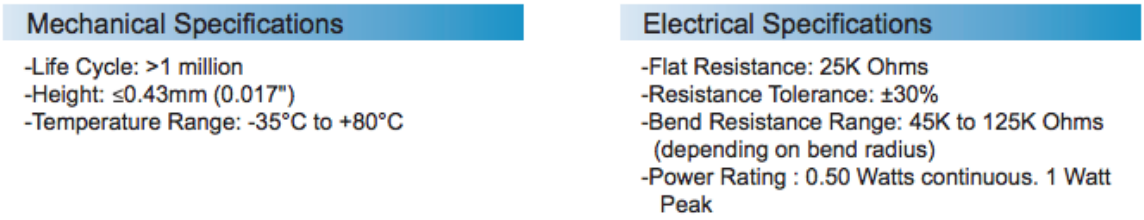


Figure 38: Flex Sensors Specifications

Furthermore, for the pressure sensors, we selected the "**Standard FLEXIFORCE Sensors: A201**" from Tekscan; we selected the 4-Pack trimmed 2-inch. This sensor is a 3-pin male connector; it has a dynamic range that can be modified by changing its drive voltage and adjusting the of the feedback resistor. Their response time is less than 5ms and they have an operating temperature range from -40°F - 140°F (-40°C - 60°C).

Moreover, we have selected the 3-Axis Gyro/Accelerometer IC-MPU 6050, "**SEN10937**" by sparkfun.com. This device is a combination of a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die. It has reduced settling effects and sensor drift, as well as sensor timing synchronization and gesture detection. Finally, this gyro/accelerometer has a programmable interrupt, which supports free fall interrupt, zero-motion detection, tap detection, high-G interrupt, and shake detention.

4.1.3 | I^2C

Implementing I^2C with certain IDEs can be troublesome. Fortunately the ATMega IDE isn't, its library simplifies the programming process for a smooth peripheral interfacing experience. Simply decide the address of using 8 of possibilities from 000 to 111, then chose whether the slave device will read (1) or write (0). Meanwhile if a slave device has not been called upon it does not react to the serial data line.

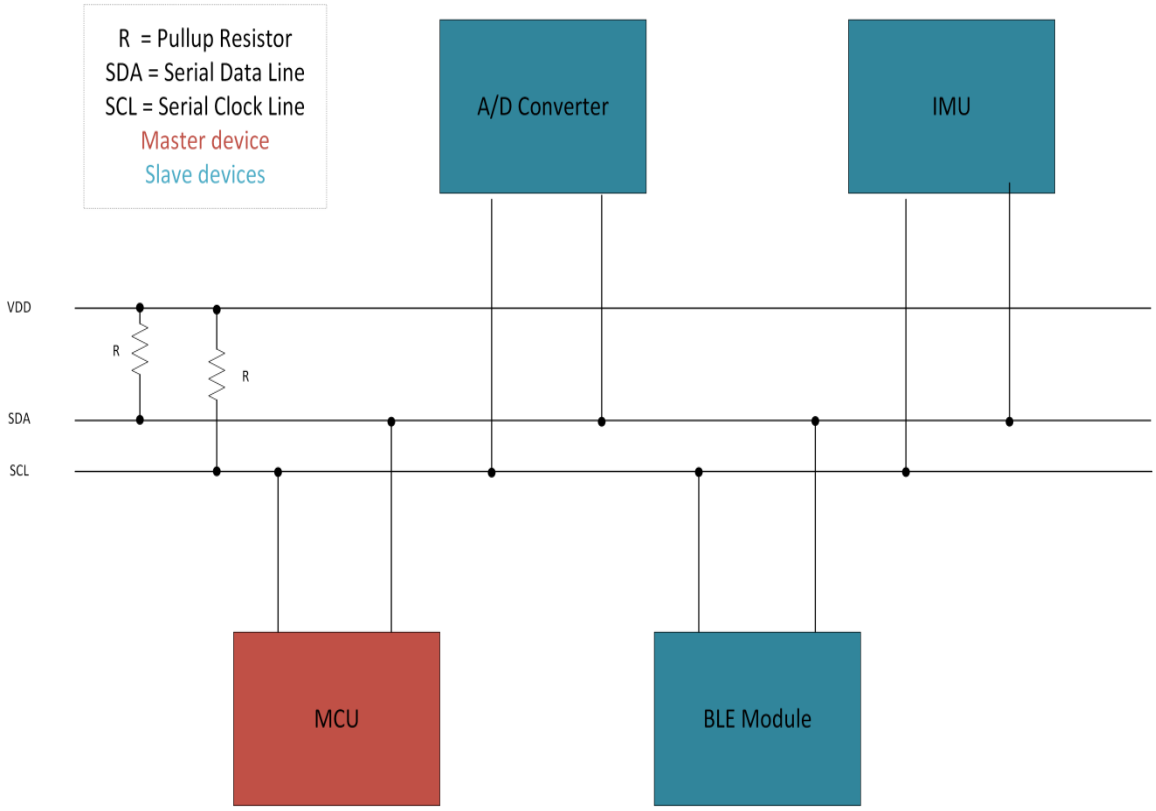


Figure 39: Startup configuration for the master/slave devices

The resistors seen are the pull up resistors to make sure that the signal settles at the expected logic level.

4.2 | SOFTWARE DESIGN

In the scheme of the language application, the group decides to develop on the Android platform. The Android community is the most suitable environment for new designs and prototypes, especially for small student projects. It is open-source allowing the group to modify the software and distribute the project app across their app market, Google Play, without additional fees or permission.

Java, the native Android developing language, is the chosen language to develop the program on the mobile device. It complements the skills set of the team's software developer. Most of their experience stems from various applications of Java classes and projects. Many integrated development environments also support this object-oriented language, and there exists an extensive community to support the integration of different technologies that may accompany the development of the project solution. The selected environment to code the Java classes for the application is IntelliJ IDEA 12 Community Edition. Their software development kit integrates other kits and technologies, along with productivity-boosting features and a powerful XML editor, to provide an easy to setup and develop a mobile app. Although the Community Edition of IntelliJ limits the

language support and development in web and mobile, the supported items satisfy the necessities of the Android application scope.

4.2.1 | COMPATIBILITY

The product specifications of High 6 require select phones for the proper use and functionality of the mobile application. With the low power communication through Bluetooth 4.0 (also known as Bluetooth Smart or Bluetooth Low Energy) and the Android operation system platform in the software design, this limits the use to a select number of smartphones. The table below lists the some of the known smartphones compatible of communicating with the Language Glove:

Manufacturer	Model
Casio	G'zOne Commando 4G LTE
Samsung	Galaxy S3, S3 Mini, Galaxy S4, S4 Mini, Note 2 and Note 3
LG	Nexus 4, Nexus 5, Optimus G, 4X, G2 and up
HTC	One, One Mini, One Max, Desire 300, Desire 601, Desire 500, Butterfly S
Xiaomi	Mi2 and up

Table 16: Common Compatible Android Smartphones

The table above contains some of the more common devices within the design of the product. To ensure the device used to interface with glove works accordingly, check against the following specifications for the required minimum system requirements in the Android mobile device and/or tablet:

Specification	Data
Operating System	Android 4.0 (Ice Cream Sandwich)
Processor Speed	1.5 GHz
RAM	1 GB
Memory	10 MB
Bluetooth	Version 4

Table 17: Minimum System Requirements for Mobile Device

Failure to adhere to the requirements of the system, especially the operating system and Bluetooth series, the product may not be guaranteed to function according to their initial design.

4.2.2 | SYSTEM STATE MACHINE

Before considering the stepping into the software design, the expected behavior of the software must be drawn first. With the expected behavior outlined, the development software design would be easier by having content to compare it against. The application may be represented by a state machine diagram to keep track of the events that take place from the startup to the exit of the program. For example, before the Android application may open to the main menu, there are several steps to take before the main page loads. Most importantly, Bluetooth compatibility must be checked before any other setup and configuration takes place. The device must be to communicate with BLE devices. In the figure below, the state machine diagram shows the expected behavior of the application functions.

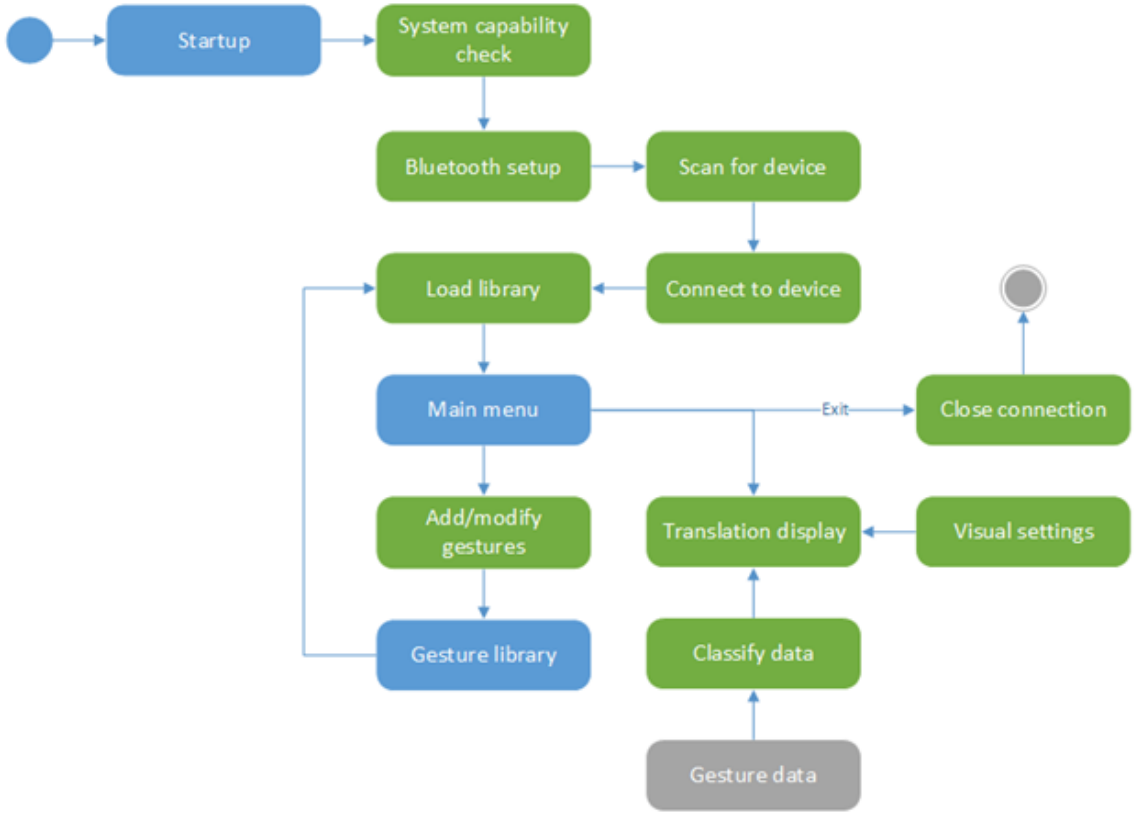


Figure 40: System State Machine Diagram

4.2.3 | ANDROID APPLICATION

The Android application is composed of three main components: the Bluetooth unit; the gesture unit; and the application manager. The Bluetooth unit ensures connectivity to the peripheral Bluetooth device and captures the gesture data transferred. The gesture unit is responsible for the addition, modification, and recognition of hand gestures. To add and modify, the unit must classify the unique motions to each gesture and associate them to the appropriate message. Then later, the gesture unit must apply its classifier to unseen data and recognize the matching gesture. Finally, the data manager is the core component bringing the user interface to the functioning components behind the scene, the Bluetooth and gesture unit. Together, these three pieces allow the software to provide the overall hand gesture translation service.

4.2.3.1 | BLUETOOTH

4.2.3.1.1 | BASIC BLUETOOTH TERMINOLOGY

The Bluetooth link requires two devices each playing a specific role in communication. Before discussing the roles and responsibilities, there are a few key terms and concepts necessary to understanding the setup, specifically within Bluetooth Low Energy. The General Attribute Profile, also known as GATT, is a high-level definition where it describes the characteristics and services used in the application and drives the behaviors and use cases. Characteristics are analogous to classes where descriptors describe the characteristic value. These services are collections of characteristics that describe the behavior of a device. The GATT is built on top of the Attribute Protocol (ATT), allowing the sending and receiving of data, known as attributes. This outlines the basic of terminology in Bluetooth communication before stepping into the application.

4.2.3.1.2 | GIVING BLE PERMISSIONS

Before using any features Bluetooth Low Energy has to offer, the application must be given the Bluetooth permission, known as Bluetooth. Given this permission, then the adapter will allow connection scans, requests, and the transfer of data. The permissions must be declared within the manifest file. These files hold information about the system, such as the version name and number, user id and label, and even accessibility to components within the device. The permissions file is written within the application itself.

To set Bluetooth Low Energy capabilities within the smartphone, the following fields must be included:

Field name	Value
Android:name	Android.permission.BLUETOOTH
Android:name	Android.permission.BLUETOOTH_ADMIN

Table 18: Bluetooth Low Energy Permissions

With these permissions, it gives the application to access the Bluetooth adapter and sets the application to be a Bluetooth Administrator. As an administrator, it grants the user to initiate Bluetooth scans or manipulate the settings within the adapter.

4.2.3.1.3 | FINDING BLUETOOTH DEVICE

Within the Android implementation of Bluetooth Low Energy, there are two main roles to establish a link: the central role and peripheral role. One device acts as a listener, actively scanning for advertisements to connection. This device would serve as the central role where it can only support a peripheral. The other device, the peripheral role, acts the speaker, making the advertisement to connection. Likewise, the peripheral may only support a central device. Hence, there may only one of each role in the construction of a BLE connection.

Taken into the scope of the project, the Android device supports the central role. Within the application, the system must have the ability to do the following:

- Discover the language glove
- Establish a stable BLE link with the language glove
- Recognize and re-connect to the language glove
- Error handling
    - System compatibility between the Android device and the language glove
    - Failed peripheral scan
    - BLE link exceeding maximum distance of 50m
    - BLE interference and breaks

On a fresh machine, the application does not recognize any device and must discover the language glove internally. The first step is to check for Bluetooth capability. With the introduction of BLE in 2010, there may be devices that do not support BLE communication. If the device has Bluetooth v4.0 or Bluetooth v4.0 LE, then the application makes sure the component is turned on. Otherwise, the application should inform the user of the incompatibility prior to startup of the Android application. Now, the system is ready to find the language glove device.

Serving as the central device, the Android system scans for advertisements created by the peripheral device, the language glove. This process is a costly operation for the battery, and it is in good practice to keep scanning to down a minimum. To keep the energy use down, scanning should stop immediately after a successful attempt to connect to the peripheral. In the case where no device is found or the language glove is not turned off, the scan should timeout, stopping the scan and informing the user of the missing signal. Thus, the scan session will last for 15 seconds or until there is a signal from the peripheral.

4.2.3.1.4 | AUTO-CONNECT BLUETOOTH DEVICES

In the case of a successful advertisement response, there will be a callback response that signifies the scan results. From here, the Bluetooth device is added to a list maintained by the Android adapter. Now in the case of the first time setup, the Android application is ready to read BLE attributes send from the language glove and shall save the device information to reestablish the link.

To allow the Android to recognize the language glove after establishing a connection once, this may be achieved by matching services. Each Bluetooth device support a specific set of services dictated by its desired behavior and functionality. Considering the language glove, it would support a hand gesture service, providing a collection of characteristic data on the posture and data. Knowing the service the device supports, the Android application may use this in its scan for Bluetooth devices. These services distinguished by a UUID, the universally unique identifier. The system would use an array of UUID objects that describe the list of services in addition to the callback response parameter and broadcast it in the initial scan. When a device with a matching set of services advertises itself during the scan, the connection may automatically re-establish itself during the startup of the Android language application.

4.2.2.1.5 | CONNECTING TO GATT SERVER

Once two devices establish their connection, the Android device may serve one of two roles: the GATT server or GATT client. Again, each role may only support the opposing role – one GATT server to one GATT client. The data transmission over the BLE link flows in one direction. The device acting as the GATT server forwards the data to the client; the device acting as the GATT client receives the data. For the language application, it will serve as a GATT client by implementing the Bluetooth service in the BLE API. To connect to the GATT server, this may be achieved by a simple “connectGATT” method call, returning a reference to the Bluetooth GATT connection. The method requires three parameters:

Parameter	Description
Context object	Reference to this GATT client
Auto-connect	Boolean to determine auto-connect when device is within BLE range
Bluetooth GATT callback	Pass results to the GATT client operations

Table 19: Parameters to connect to GATT server

With the Android device and the language glove connected, gesture data may pass from the language glove to the mobile device. The details of service for the Bluetooth connection will be discussed in the next section.
Finally, when the application is ready to close, the Bluetooth LE connection must be closed releasing the Android system resources appropriately. This may be achieved by calling the close method for the Bluetooth GATT connection variable. The conditions for closing the device are as follows:

- When the device is closed through the application console
- The application is closed by a Home key press
- The application is not compatible with the Android mobile device

The diagram below captures the states taken for Bluetooth setup during the startup of the Android application:

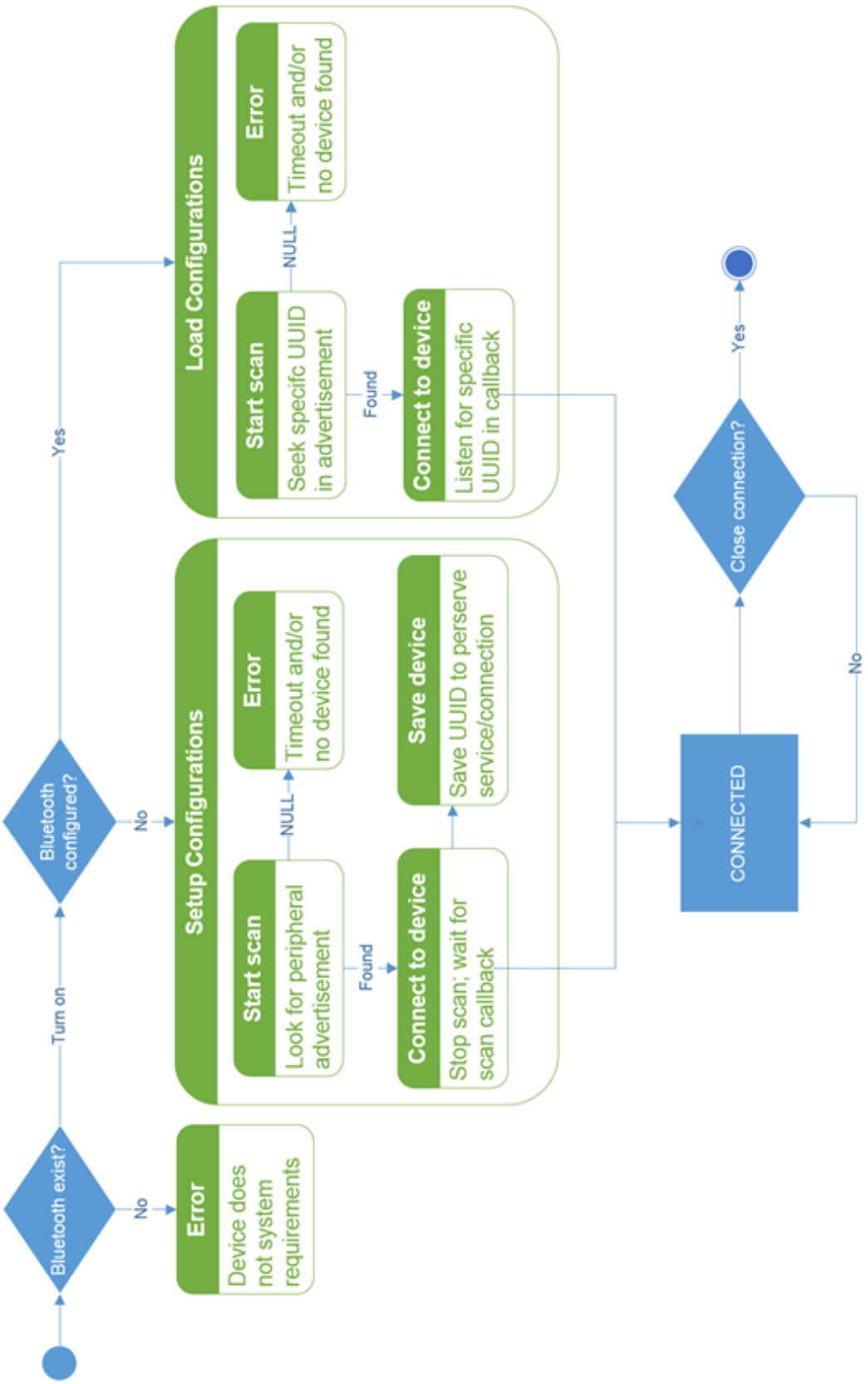


Figure 41: Bluetooth Central Role UML State Diagram

4.2.3.1.6 | RECEIVING BLE DATA

Given that the Bluetooth LE connection is successful, the gesture data taken from the glove is ready to be passed to the Android application. This entails the reading and writing of attributes. The language glove would send the collection of characteristics in the supported services, and the mobile device would read these and ready the data for the translation. In this section, there are discussions of the supported services and the data structures necessary to meet the project requirements.

There are several variables to consider when capturing the dimensions of a hand gesture. These include:

- Flexion of each finger
- Contact among fingers (e.g. the thumb touching the base of the middle and ring fingers)
- Orientation
- Three-dimensional movement

One service will be sufficient to encapsulate the gesture data and shall be called Gesture Monitor Service. This service includes the following characteristics to represent the listed factors:

Characteristic	**Characteristic List Variable**	**Grouped in**
Thumb flexion	Hash table <String, String>	Flexion array list
Index flexion	Hash table <String, String>	“”
Middle flexion	Hash table <String, String>	“”
Ring flexion	Hash table <String, String>	“”
Pinky flexion	Hash table <String, String>	“”
AccX	Hash table <String, String>	Accelerometer array list
AccY	Hash table <String, String>	“”
AccZ	Hash table <String, String>	“”

Table 20: Characteristic Data Table for Gesture Monitor Service

The table above shows the expected characteristics within the Gesture Monitor Service. The use of array lists and hash tables are common in the server's service implementation. As the live data continue to increase, the data structure must be able to store the current datasets and support the incoming data as well. With the use of hash tables, it can store the values of each characteristic value located at a position determined by a characteristic UUID key. Similar to services, characteristics are also uniquely identified through a UUID. Because there is a small set of characteristics groups, these hash tables may be stored in an expandable array list. Typically, there would be another dataset for service UUID with service name setup, similar to the characteristic table above, but there is no need for table for services. The language glove focuses around transferring gesture data alone. The rest of the project's operations are exclusive to the mobile device: translation, gesture recognition, gesture libraries customization.

4.2.3.1.7 | TRANSLATING BLE DATA

Within the delivery of the services across the BLE link, there is of excess information. Services and characteristics are serialized by UUID to uniquely identify content. Once the information successfully makes it to the client-side, or the smartphone, it loses value to carry it through the application to be translated. Instead, the UUID for the service and characteristics may be discards. The significant data is the characteristics data within the Gesture Monitor Service. This holds the finger flexion, contact points, and the movement within the hand system.

To reduce the service data to the significant gesture data, the ArrayList mentioned in the Characteristic Data Table earlier must be transferred to another data structure. Originally, the characteristics stores the gesture data as strings captured from the glove's sensors. These strings have the most precision, but it is expected that these values are limited.

Sensor	Expected Range
Flex sensors	45k – 125kΩ
Accelerometer	0 – 19.62 m/s^2

Table 21: Expected Sensor Data

The expected range of values gives insight to the data types needed for translation of data structures. The flex sensors have large numbers with a relatively small range. The accelerometers are especially small in both magnitude and range. These patterns are important in the application, turning unseen data into the appropriate message. For example, when the Android application is exposed to a gesture, there will be fractional differences in the recognized gesture and the incoming gesture. This beckons the question of precision when handling these operations. Given the full precision data, the size

may not be noticeable to perform the translation, but scaled to a larger set dataset, the size of the data type used may reflect on performance time.

There is a compromise between maintaining accuracy of the data and the size of the data structure. With growing arrays of large sets, 32-bit doubles may drag the down the performance. It would prove to be especially wasteful for the flex sensors. Decimals for 45kΩ, the smallest measurement, of the flexion would become insignificant to keep and use for the later. Integer would be more appropriate, but the accelerometer must retain its decimal value with its smaller range. The midway point to the data structure would involve ArrayLists of float types.

Variable	Type	Stored in
Thumb flexion	Float	ArrayList
Index flexion	Float	
Middle flexion	Float	
Ring flexion	Float	
Pinky flexion	Float	
AccX	Float	
AccY	Float	
AccZ	Float	

Table 22: Data Structure for one unit of gesture data

In addition, it is important to note that all the variables are of float-type. This is because this data is involved in the same mathematical model. From a developer's perspective, there may be typecasting present in the calculations. If a floating point number and an integer perform some algebraic operation with each other, the float becomes truncated, losing precision and creating rounding error. Thus, all variables are float-type to compromise between mathematical accuracy and data size.

4.2.3.2 | GESTURE UNIT

4.2.3.3 | APPLICATION MANAGER

The Android half of the project will be driven by the application manager. This component is the middle man between the user interface and the background classes and methods that support the overall language translation system. It is responsible to take the steps necessary to allow gesture translation once the application startup is complete. The manager also needs to be capable of smoothly handling errors when interacting with the user or the other software components. This includes interacting with the Bluetooth component and gesture unit to achieve the following:

Gesture Unit
1. Create a gesture library
2. Modify a gesture(s) within an existing library, including adding, changing or deleting a gesture(s)
3. Receive a list of supported gesture libraries and able to select active library
4. Pass one or multiple units of gesture data and receive an accurate translation
5. Receive NULL case in the failure of an invoked method

Bluetooth Unit
1. Able to control the Android Bluetooth LE component (turn on & off)
2. Receive a list of available Bluetooth LE device(s) to select from or automatically connect to language glove
3. Receive a data handler on live data
4. Receive NULL case in the failure of an invoked method

Table 23: Functional requirements of Interacting Software Components

The table above lists the functions as the application manager interacts with the components, but this looks different when the user utilizes different aspects of the application. In the following sections, different aspects of the Android application will be discussed and how the user may use it.

4.2.3.3.1 | GRAPHICAL USER INTERFACE

The graphical user interface (GUI) focuses around the idea of being readable, user-friendly and intuitive. While there is heavy hauling calculation and design in the background processes, the GUI should remove any disparity in the knowledge difference and ensure the end users, sign language speakers, may easily communicate to their respective audience. With their audience in mind, this brings up the first concern of the GUI to achieve our non-functional goals – readability.

Catering to mobile devices, the screen size may vary dramatically affecting the font size. The Samsung Galaxy SIII mini and the Casio G'zOne Commando 4G LTE are two of the smaller Android devices that support BLE with their 4.0" screens. Compared to more recent smartphones suitable for the project, the builds of the HTC One Max and the Samsung Galaxy Note III include a hefty screen size of 5.9" and 5.7" respectively. With screen size differences up to 47.5%, there may exist a disparity in readability. Then, there are other details that may affect readability including:

- Font
- Color style (text-to-background contrast)
- Alignment
- Spacing

Considering the natural build differences along with the styling of the text, readability becomes an important factor to consider.

The font size among different devices will not be uniform. The application must scale its solution accordingly across different mobile environments. With this in mind, the font size must be adjustable, but will default to a specific size along with the other font styling factors mentioned earlier. The values for the font styles and color scheme the group finds suitable are as follows:

Style	Default
Font	Roboto Condensed
Font Size	24 sp
Color Style	Primary text in black; secondary text in gray; background in white
Alignment	Full
Spacing	1.5 sp

Table 24: Default font styles

This style can be supported by one typographic tool set for Android known as Roboto. Used since Ice Cream Sandwich, these tools include the styles mentioned plus rhythm. The Roboto design is based on a skeleton with geometric forms surrounding the frame. This allows scaling of the font without any pixilation. The font size uses sp, or scale-independent pixels, to work against the system-wide scaling factor. With each smartphone, there may be customizable scaling factors that change the fonts, including those within the language application. To prevent inconsistent font size profiles among different devices, sp is a great aesthetic feature to have in font styling. It also provides two default color styles for the primary and secondary text against the background to provide separation through color contrast.

Figure 42: Supported color styles

The alignment has an underlying grid system to maintain text positions. Rhythm is the last feature of Roboto, which is better known as vertical rhythm. This gives the text appropriate spacing between bodies of text equivalent to the line height with text rather than simply the line. Rhythm is not essential to our text outputs, but Roboto packages the essential text styling to improve readability.

Android also give the developer the ability to create different XML layouts for different screen sizes. In the Android OS, there are 4 different categories of screen sizes: small, normal, large, and extra large. X-large screens are at least

960dp x 720dp. large screens are at least 640dp x 480dp. normal screens are at least 470dp x 320dp. small screens are at least 426dp x 320dp

dp stands for density independent pixels. Density independent pixels are pixels that are virtual, and the size of a density independent pixel is equal to one physical pixel on a standard 160 dots per square inch screen. For each of the 4 screen sizes, a developer can come up with custom layouts. So if the layout for an extra large screen looks good on an extra large screen, but then ends up looking bad on a small screen, that's no problem. Once the phone detects that the screen size is small, it will automatically use the layout that's defined for small screen sizes. So, to help compensate for all the different screen sizes, our application will have 4 different layouts. Also, our application will not have a landscape view. It will be locked in portrait mode. This is so that we don't have to do 8 entire layouts just so that the app can go into landscape mode. After the layouts are made in XML, the android development environment will take the XML files and create java classes out of them. Aside from just having different layouts for different screen sizes, android also allows a developer to add different qualifiers to the layouts, such as what region the phone is in. This can allow the app to be set to a different default language if our group were to continue working on this app after the senior design project.

4.2.3.3.2 | MENU NAVIGATION SYSTEM

Considering readability and the limited screen space, menu options may significantly reduce the available space to have one continuously visible to the user. Coming from that perspective, the menu must be able to toggle between visible and hidden. This is especially important during the translation and active input into the Android system. The space lost to the visible menu options would draw away from the readability of the output text area. The active form may become congested with the menu and the text. At the same time, it is important that the user is aware of the available options within the active form of the Android application if the menu is hidden. Thus, in the application, it would prove helpful to allow the menu system to be visible during the first seconds of the application before it minimizes or hides.

The basic layout of the application's forms would looks similar to the following:

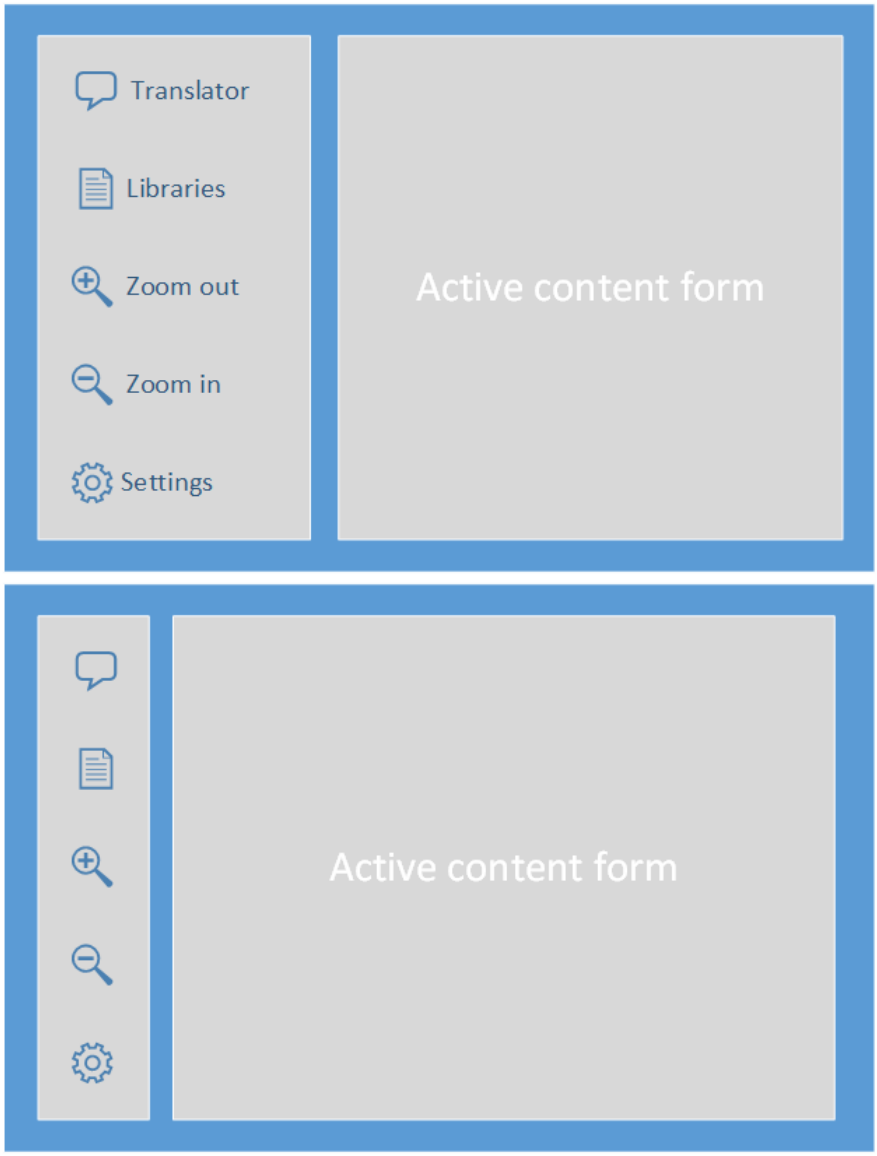


Figure 3143: Generic Application Layout

The above image captures the standard form of the Android application for the different supported functions. Notice that it is in the landscape orientation. The display will fix to either landscape orientation but will not support portrait orientation. Also, the orientation of the screen will only change accordingly if the system-wide display auto-rotate feature is enabled. Otherwise, the system will lock the response to the orientation change of the smartphone.

These menu items map to the core functioning features of our application:

1. Translator is the menu item that opens a console set to translate the user hand gestures. Upon opening the application, this is the starting point.
2. Libraries is the menu item that gives the user access to the supported gestures through the Library Gesture Editor. They allow the user to select

the supported hand gesture library when using the Translator tool and modify the libraries, including adding a new one.

3. The Zoom out/Zoom out in keys allows quick access to scaling font sizes. This option is only available to the Translator tool and disable within other tools. When active, it provides the user convenience to increase or decrease the font size rather than changing it manually within the Settings. Each key press would increment or decrement the font size to the next supported font size.
4. Settings is the last menu option that allows user to adjust the typography. Similar to the Zoom in and Zoom out items, this item is also exclusively available to the Translator item. The styling and appearance includes font, font size, and color styling. The Settings also gives the option to see the active library set as well as change it.

4.2.3.3.3 | LIBRARY GESTURE EDITOR

The Library Gesture Editor is the main access point to the Gesture Library for the end user when the Libraries menu item is selected. Within the editor, the user is given the ability to essentially add, edit, and remove a library or gesture within a library. The following set of functions is given to the user:

Supported Library Functions
1. Create a new gesture library
2. Select the active gesture library
3. Delete a gesture library
4. Restore original gesture library

Supported Gesture Functions
1. Add a new gesture
2. Change an existing gesture
3. Delete a gesture

Table 25: Library Editor Supported Functions

These functions capture the learning capabilities of the language application. The application comes with an original library set that include the essential gestures to communicate with their respective audience. This includes:

- American alphabet letters
- American numbers zero to nine

Additional gestures may be added into a library repository. When doing so, the application requires the user to demonstrate up to ten gestures. Depending on the consistency of the gesture, the learning process varies from four to ten gestures.

As a key objective, the steps taken to modify the existing or new library must be intuitive. To address that aspect of the non-functional objectives, the editor takes a minimalistic approach to GUIde the user on the procedure of the editor function chosen. The diagram below will show the simple layout of the editor options:

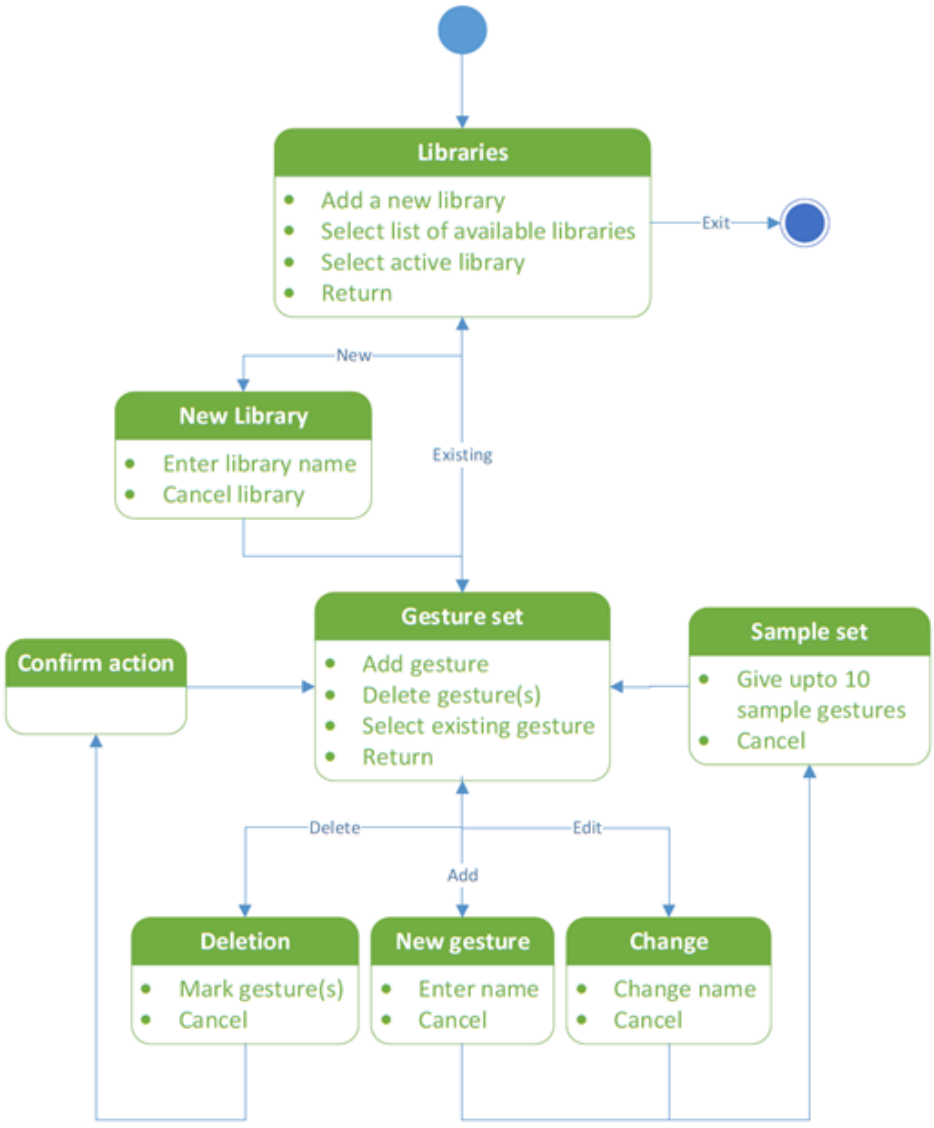


Figure 44: Library Editor UML State Machine Diagram

It is important to note that the options available in each state. For instance, the Gesture set state has four menu items:

- Add gesture
- Delete gesture(s)
- Select existing gesture
- Return

These items are visible in the menu bar fixed to the left side of the landscape oriented form. For original menu bar items, they would be overwritten with the editor's options until the user exits from the editor and returned to the Translator. The menu option setup for the Gesture set state would look similar to the figure below:

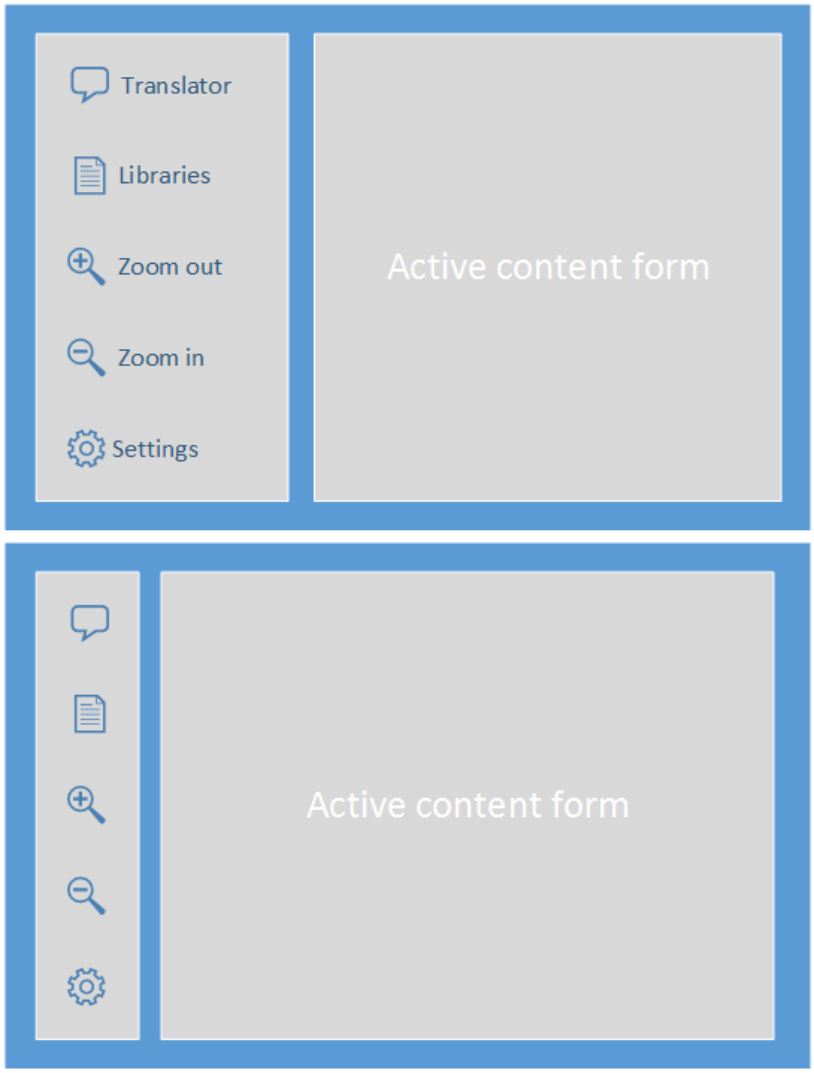


Figure 45: Changing Menu Options within Library Gesture Editor

The user would see similar changes to the menu options as he/she navigates through the different states of the Library Gesture Editor.

4.2.3.3.4 | TRANSLATOR

The translator consists of 2 components. One translator component is the component that is responsible for taking the sign language gestures and converting them into actual ASCII characters. This component will be entirely on the android smartphone/tablet, and implements the Hidden Markov Method. In order for this module to work, there needs to be a large enough training set already implemented to allow proper classification of user input. In terms of how exactly this software component works, it will implement dynamic programming methods to do the statistical equations that were discussed in the section about the Hidden Markov Method. Dynamic programming will be used because it is far more efficient than recursive methods which redundantly recalculate data whereas Dynamic programming saves data to prevent the CPU from wasting time. The mathematics for the Hidden Markov Method is still in the process of being fully understood. The overall functions of the software will be decided once the Hidden Markov is understood well enough to create a system overview of the components that make up the gesture to letter translator component. The data that this component receives will be created from packets of buffered data from the glove sent ever s seconds. The packets will be recreated into a continuous stream of information, because the Hidden Markov Model will require a continuous stream of data in order to accurately tell what letters are being gestured by the user.

The other major component of the translator is the text to speech feature. We will not be designing a text to speech component from scratch, because this is not the main feature of our application. The main feature of course will be converting hand gestures to ASCII text characters. But it was decided that since there's some built in android libraries for converting text to speech, it would be a good feature to implement. The android text to speech features are all found in the android package android.speech.tts.TextToSpeech. All functions and text to speech features will be pulled from that class. Textospeech.onInitListener() will take in the text that has been translated, and Texttospeech.speak() will go ahead and speak the text that has been given to it. Texttospeech.onInitListener() will be called when the text to speech object is being created through the constructor for it. The constructor for it is TextToSpeech(Context context, TextToSpeech.OnInitListener listener). The context is basically settings on the phone that are relevant to the text to speech class. It contains information such as the language being spoken, etc. The android package android.speech.tts.TextToSpeech is found in android API 18, which is the library that supports Android Operating System 4.3 (code named KITKAT). The phone that our group is planning to use is the Samsung Galaxy Note II, which has Android 4.3. If another person wants to use this application, they must have

android API 4 or higher. The higher their API the better because more features of the text to speech class become available as the API gets higher.

4.5 | FPGA

4.5.1 | DEVELOPMENT BOARDS

A development board is a PCB that contains a microprocessor that facilitates the experimentation of an FPGA. A development board has power circuitry, programming interface, input and output indicators, and input and output pins. Besides its low cost, a development board allows the development of both simple and complex projects and applications. From the multiple development boards available in the market we decided to use the ATMega-Leonardo. This development board is based on the ATmega32u4 and its hardware and software are open source.

We selected this development board for various reasons: First of all, it met all the requirements our project has such as low power, enough documentation available to the users, and most important it has enough pins to connect the overall system, which is the 21 sensors and the Bluetooth manual. Second, our group wanted to have experience in different microcontrollers. Since our group experience with the MSP430, we thought this project would give us the opportunity to broaden our knowledge in another microcontroller development board. Lastly, this board is open source and has a community available and willing to help with any question or sharing information and documentation needed, which will help increase our learning curve at a faster rate than without any type of help or documentation.

4.5.2 | FPGA COMPONENTS

Our FPGA will have three main components: microcontroller, Bluetooth, and sensors. The following picture depicts the ATmega32u4 microprocessor. It is a low power 8-bit microcontroller with advanced RISC architecture; it executes instructions in a single clock cycle, and allows power optimization while maintaining its processing speed. This microcontroller has a total of 20 pins, 12 of which can be used as analog input channels. The 12 analog inputs are labeled A0-A11; inputs A6-A11 are located on digital I/O pins 4, 6, 8, 9, 10, and 12.

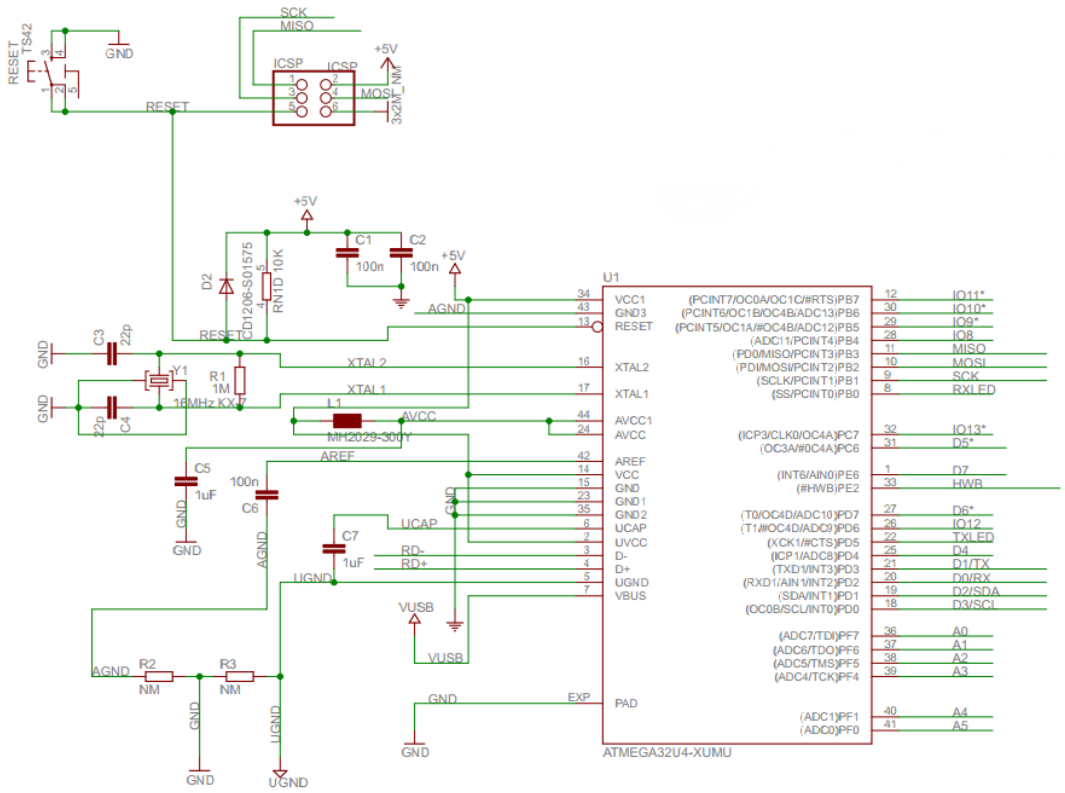


Figure 46: ATmega Microcontroller Schematic

Besides connecting the 21 sensors to the microcontroller, we will connect the Bluetooth Low Energy as well. The picture below shows the schematics of the BLEmini from Redbear that we will be using to send the information gather from the hand gesture to the android device. This BLE will be connected to the ATmega32u4 through pins 0-7, 10-13, or A0-A5 except pin 8-9 since they are the default handshaking switches. We will also have AC-DC converters for the flex sensors, pressure sensors, gyro/accelerometer, and BLE.

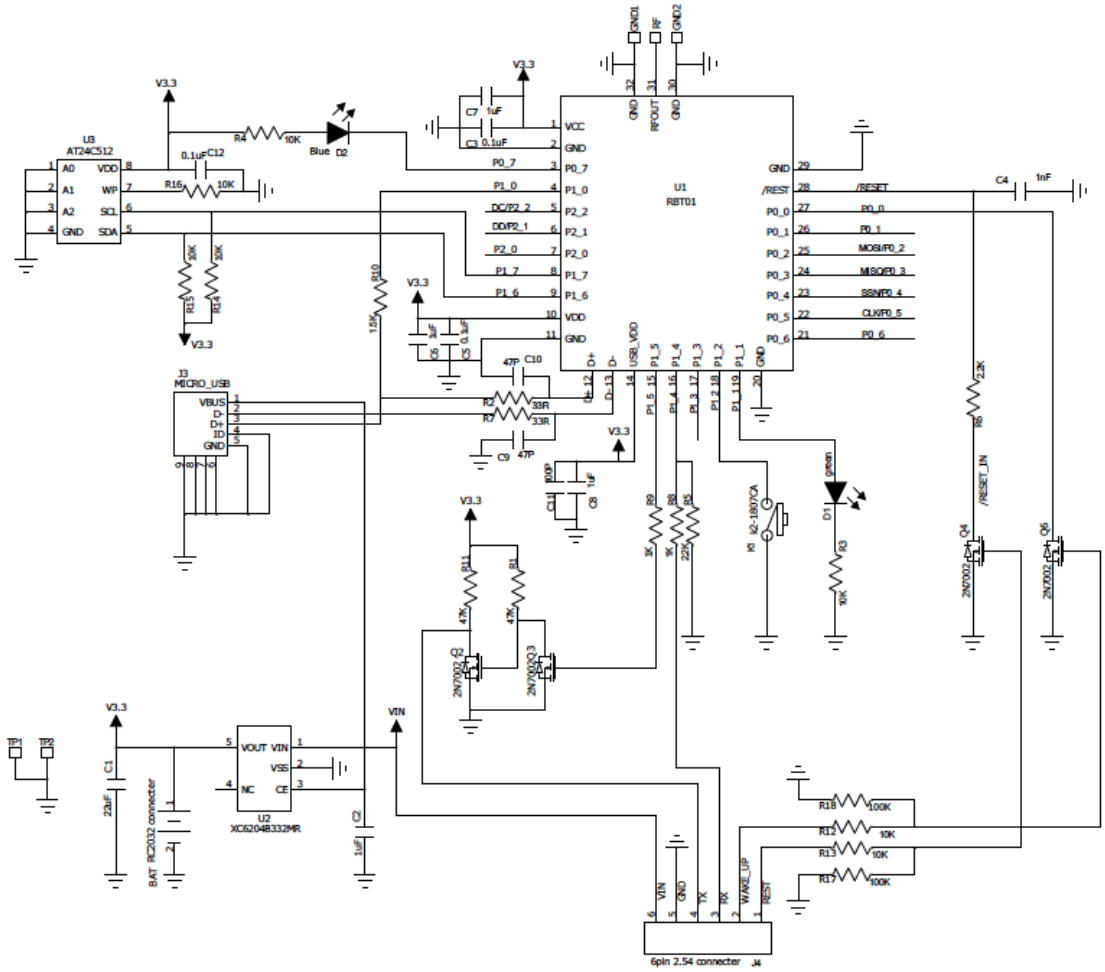


Figure 47: BLE mini Schematic
Image courtesy of RedBearLab

4.6 | GLOVE AESTHETICS

The glove used for this project will be a golfing glove. We selected this type of glove since it provides moisture resistance, as well as it maintains the hand at a cool temperature and it has a comfortable fit. There will be two flex sensors attached to each finger and one that goes in the palm of the hand. The pressure sensor will be located as follow: one between the index and middle finger, one at the tip of the index and middle finger, and one at the tip of the thumb. Since the hand rest naturally in a mild curve, we believe that the best position to put the circuitry will be at the top of the hand. The sensors will be mounted inside each finger of the glove and one inside the glove on top of the palm of the hand, as shown on the following figure.

Figure 48: Mounting of Glove

Besides the flex sensors and pressure sensors, on the master circuitry we will have the gyro/accelerometer, Bluetooth (BLE) module, and the battery. The rechargeable batteries will be wired on to the circuit by soldering into the PCB. The master circuitry will be encased with a plastic container that will be sewed on to the glove. We have yet to select the material of the container; this material needs to be heat resistant so it does not melt because of the power running through it. The rechargeable batteries have their own encasing in that way when it is time to recharge them will be very easy to do by placing them into the charger.

5 | PROTOTYPE

5.1 | TEST ENVIRONMENTS

5.1.1 | Temperature

The temperature of the glove's performance is defined by the temperature ranges in which the components in the glove will operate without error in data collection and transfer. Because of the nature of the subtle differences in hand gestures, if one device fails then the entire glove's functionality is compromised. Below is a chart of the operable temperatures as told from the manufacturer for each device.

Device	Temperature Range (℃)
Flex Sensors	-35 < T < 80
Pressure Sensors	-40 < T < 140
IMU	-40 < T < 105
Bluetooth Module	-40 < T < 125
A/D Converter	-40 < T < 85
Power Source	-10 < T < 75
Voltage regulator	-40 < T < 125

Figure 48: Temperature Ranges for the High 6 glove components

5.1.2 | WEATHER CONDITIONS & LIMITATIONS

Even though the golf glove material is perfect to keep the user's hand cool, it is important to keep the glove away from extreme weather conditions. More important, to maintain the glove away from dust and humidity due to the fact that too much of either one can cause a decrease on performance of the gyro/accelerometer. Therefore, is recommended to use the glove inside a place with air conditioning and low humidity. Furthermore, this glove cannot be used or submerge under water. Water will damage the battery, circuitry, sensors, BLE, and gyro/accelerometer.

5.1.3 | ELECTROSTATIC DISCHARGE

Electrostatic discharge occurs when two objects with different potentials come into direct contact with each other. Electrostatic discharge can also be developed when high electrostatic fields occur between two objects that are in close proximity. Moreover, electrostatic discharge is one of the major causes of device failures. The discharge mainly occurs due to an imbalance of electrons on the surface of the material. Usually, an electrostatic discharge is created by insulator surfaces that are either rubbing together or pulling apart. Electrostatic discharge can occur in four different ways such as a charge body touching an IC, a charge machine touching an IC, an electrostatic field introducing a voltage across a dielectric, and a charged IC touching a grounded surface.

Since electrostatic discharge is the major cause of damage in semiconductor integrated circuits, there are three methods widely used to describe method of establishing electrostatic discharge thresholds. The first model is Human-Body Model. It simulates the human body discharging accumulated static charge

through a device ground. The second model is Machine Model; this model simulates machines discharging accumulated static discharge through a device in ground. The third and last model is Charged-Device Model. This model simulates charging and discharging events that may occur in production equipment and during processes. This type of model tends to occur on when there is metal-to-metal contact in manufacturing.

It is very important to keep each integrated circuit under the same potential as its surroundings. The most important rule to avoid electrostatic discharge damage is to maintain each device and everything in close proximity at electrostatic discharge ground potential. Besides the rule of thumb, it is important to follow Texas Instrument's four supplemental rules:

• Any person handling ICs must be grounded either with a wrist strap or ESD protective footwear, used in conjunction with a conductive or static-dissipative floor or floor mat.
• The work surface where devices are placed for handling, processing, testing, etc., must be made of static-dissipative material and be grounded to ESD ground.
• All insulator materials either must be removed from the work area or they must be neutralized with an ionizer. Static-generating clothes should be covered with an ESD-protective smock.
• When ICs are being stored, transferred between operations or workstations, or shipped, they must be maintained in a Faraday-shield container whose inside surface (touching the ICs) is static dissipative.

Having into consideration electrostatic discharge, we believe that by selecting an insulating golf glove and the two separate encases for the battery and the PCB we will be able to decrease any discharge between the sensors, PCB, battery, and most important the user.

5.2 | HARDWARE SPECIFIC PROTOTYPING

5.2.1 | FLEX SENSORS

In order to make sure our flex sensors are working properly we will create a fabric bend for each flex sensor, then we will connect each end to a multimeter and we will measure each flex sensor resistance in ohms.

5.2.2 | PRESSURE SENSORS

To test our pressure sensors we will connect each pressure sensor in the breadboard and will apply pressure to each sensor. We will squeeze the conductive plates on the pressure sensors and we will measure the change in resistance in each sensor as well as the current passing through it.

5.2.3 | Bluetooth Low Energy

Part of the debate over using BLE for the High 6 glove was that the relatively new technology is not particularly well documented in the online community. In contrast, Bluetooth v3.0 modules have been available to the market for quite some time. As a result, the online community has put forth a wealth of information on many scenarios and cases for programming embedded systems with Classic Bluetooth capabilities. Since this project is about device feasibility and application prototyping rather than a production ready device, the group decided to use the new Bluetooth v4.0 Low Energy technology.

5.2.4 | IMU

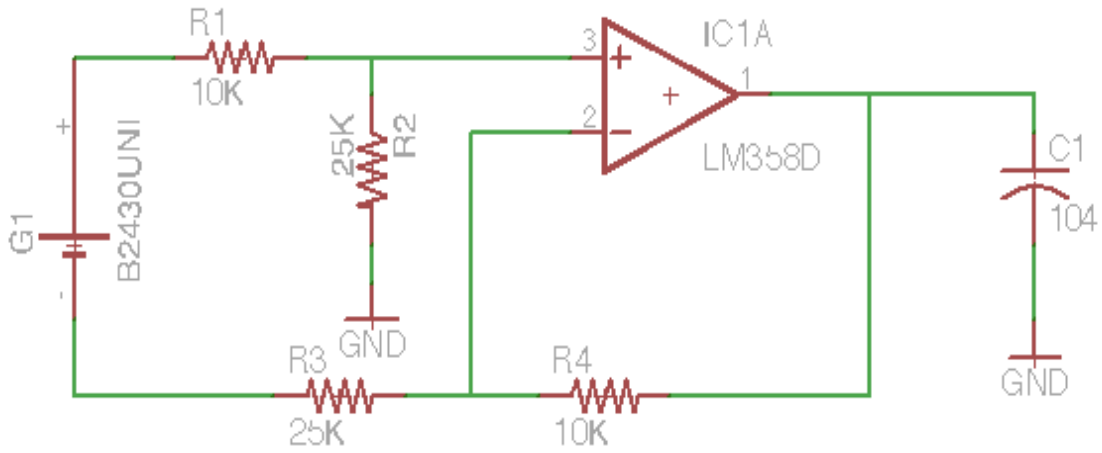
With the Inertial measurement unit correctly wired, integration of the IMU relies on the serial I^2C interface. The code will be written in the ATMega IDE and the IMU will be asked to transmit data back to the microcontroller in sequence with the Analog/Digital converter. The rate at which the IMU can interpret motion is 2000˚/second and the sampling rate can be modified with a simple change in code.

5.2.5 | MCU

The microcontroller is the heart of the gloves design and should therefore special attention should be given. Connecting the I^2C devices to the SDA and SCL lines allows the remaining sensors and BLE module to connect to the GPIOs. Several of the channels have the Analog input feature; therefore the code written will simply take the values from the pressure sensors and flex sensors on these pins. Last, the BLE module must be connected in to the remaining digital pins on the Atmega32u4 microcontroller. Since the MCU is the master device it will be where most of the local computing will take place.

5.2.6 | POWER CIRCUIT

In order to test the voltage between the positive and negative poles of the battery, we will use an operational amplifier to do this job. An operational amplifier is a voltage amplifier with high gain. Operational amplifiers are linear devices that have all the properties required for DC amplification. An amplifier of this type has three terminals, consisting of two high impedance inputs. As shown in the following picture, the operational amplifier will be used to measure the voltage on the battery.



Figure#. Operational Amplifier

The output voltage from the operational amplifier will be obtained by: $\mathrm{Vout} = \frac{\mathrm{R2}}{\mathrm{R1}}\,\mathrm{V} + \mathrm{V}$. The V_{out} is the output voltage from the operational amplifier to the microcontroller.

Besides the voltage of the battery, we will also test the temperature of the battery. It is very important to check that the charging of the battery terminates when the battery's temperature reaches above the operating limit. At the same time, we will test the temperature by taking the measurement of the temperature coefficient resistor, which is powered by the V_{dd} of the microcontroller. Since the resistor is a temperature coefficient resistor, the temperature should change when the temperature changes.

5.2.7 | GLOVE & MOUNT

We will perform two tests, heat test and weight test, on the glove and mount to make sure it meets physical requirements to be wearable and able to function without being harmful to the user.

Based on the physical requirements, we need to check that the weight of the glove is less than two pounds. In order to check this, we will put a 3-pound weight system on the golf glove and the tester will make hand gestures as well as perform the 26 hand gestures of the American Sign Language Alphabet. This movements and hand gestures will take less than a minute each, so in total the tester will spend about 30-40 minutes testing the weight of the glove. In this way, we will make sure that we glove will be able to support to support the weight of the circuitry.

To make sure that glove and mount of this project are the correct ones, we will also perform a heat test. We will place the glove and its mount on a container at 75 ˚C for an hour and a half. We selected this temperature because it is greater than the actual temperature that the electric circuitry could dissipate. In order to accomplish this, the glove will be place on a metal tray to avoid any accidents that may occur. Once we confirm that the glove and mount can resist the heat, we can say this test has passed successfully and the glove and mount are ready to be used.

5.3 | SOFTWARE SPECIFIC PROTOTYPING

5.3.1 | BLUETOOTH UNIT

The development of the Bluetooth unit is the smallest but remains vital to the overall performance of the Android application. There are two permissions necessary for this component: Bluetooth and Bluetooth Administrator. Through the use of Bluetooth, the device must be given access to the adapter. Because the application will scan for the language glove, the Bluetooth Administrator permission also becomes essential.

Now, that the application may perform device scans, finding the Bluetooth device is the next step. This phase requires the corresponding Bluetooth module on the language glove to be functional. Regardless of the development of the glove's progress, there must be error handling present to handle no device found and compatibility conflicts. When glove's Bluetooth device is functional, the Android application waits for a Bluetooth callback. The callback determines the success of the scan results.

When a device is successfully found, the Android smartphone must connect to corresponding device as a GATT client. This will make the application the listener for incoming data packages, or services, from the glove. Just as how the connection is able to be opened, the connection must be able to close. The BLE connection closer implementation follows next. Afterwards, this point marks the first milestone when the devices may connect and disconnect successfully, and data is ready to be transmitted over BLE link.

Before advancing into implementing the full Gesture Monitor Service, the BLE link must be tested. For simplicity, we would forward a test string packaged within a service according to the design. This will determine the accuracy of the implementation of the Bluetooth connection thus far.

After passing the first milestone of sending the first service contains a string, the full service is ready to be added to the developing Bluetooth unit. The last unit of work is the auto-detect of the compatible devices. After the first time two devices

are connect, the application should become aware of the glove when they both turned on and within range of each other.

5.3.2 | GESTURE UNIT

5.3.3 | APPLICATION MANAGER

The application manager prototyping starts with generic forms that serve as a template. The template will contain two forms: one smaller rectangular form fixed to the left side that is extendable by gesture moving the visible edge in or out; the other of a fixed size taking 90% of the screen height. When the left form expands, the forms will overlap. The template would look similar to the following:

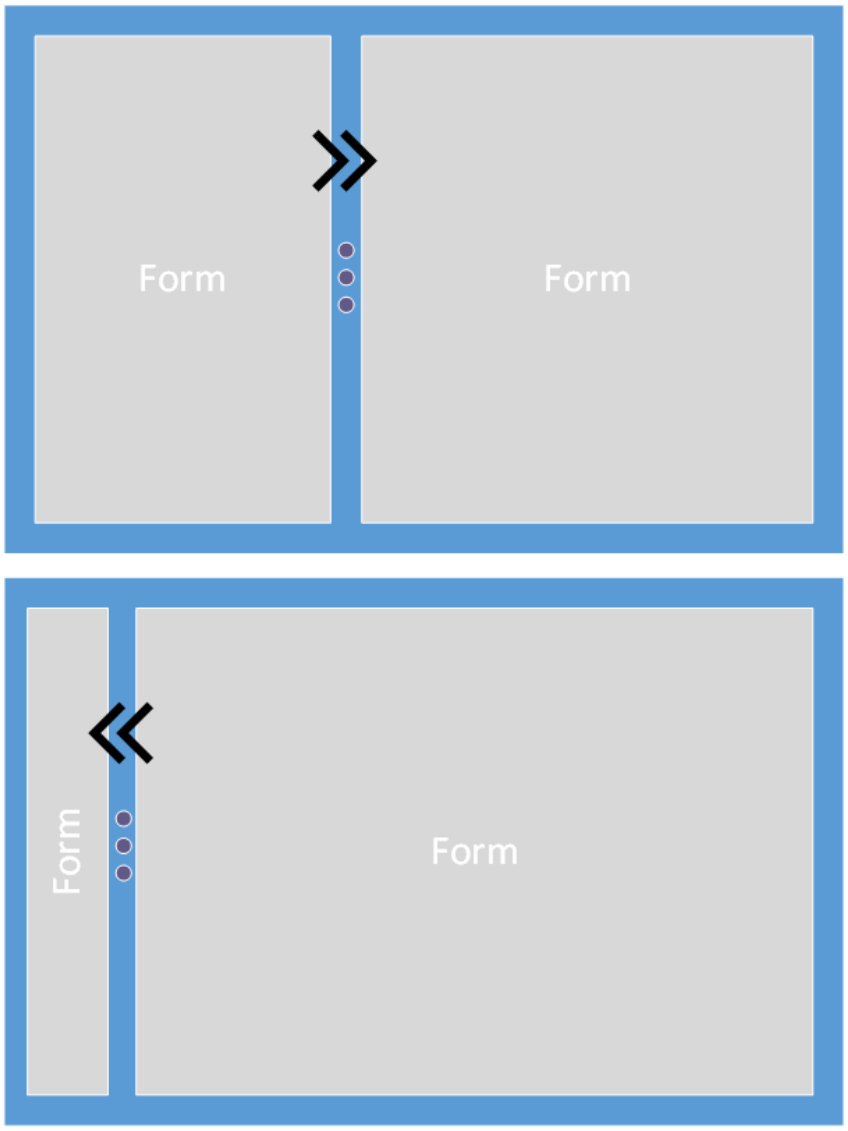


Figure. Application Template

Using this template, the different features may populate the forms with their menu options and content. The next step is the development of all pages and linking them appropriately. All of the clickable components that trigger a backend event outside the graphical interfere is not included. Instead, the button events will be null and simply link between the different pages.

Finally, the typographic styling within the Settings, Zoom in, and Zoom out is the last component before integration of the three core components of the application. Lorem ipsum text will be fixed to the Translator's main output field in this phase of the prototype. Then, the styling changes may be visible to the text.

5.3.4 | SYSTEM DEVELOPMENT

Prior to integrating the subsystems of the software components, the figure below is the visual representation of the developmental phases for the software components in parallel.

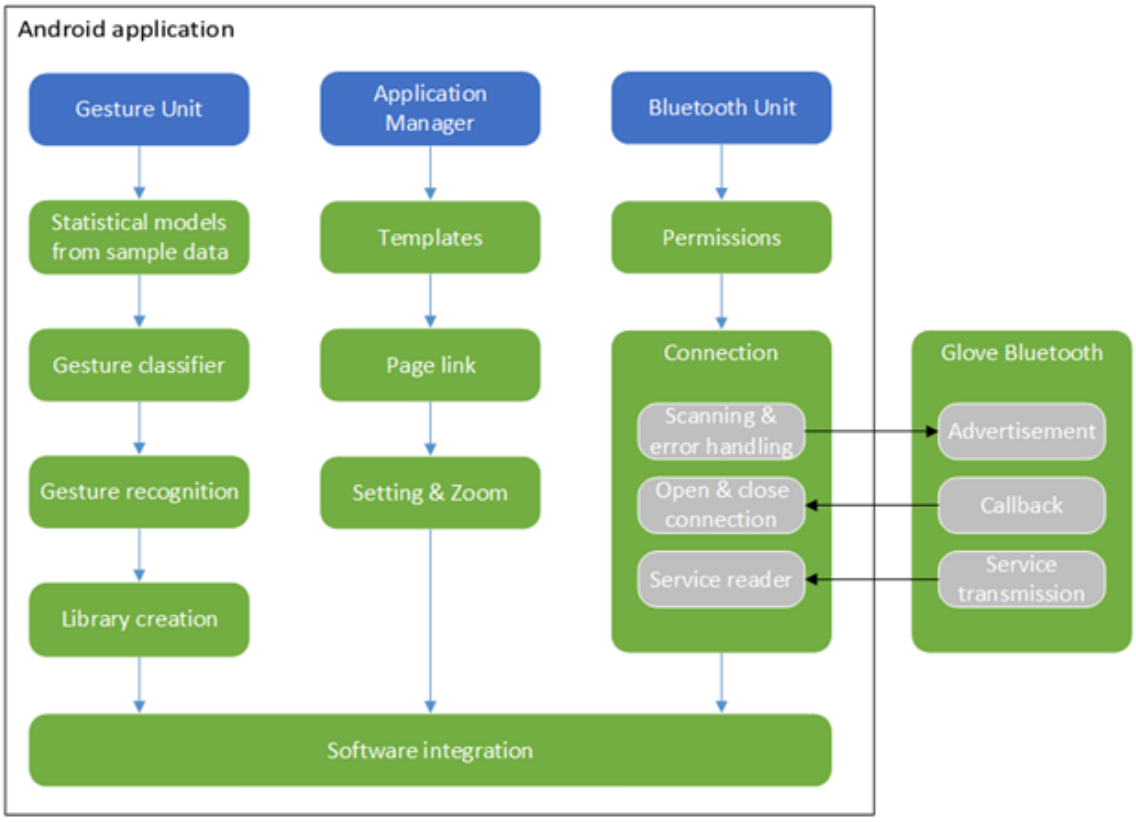


Figure. Development of Software Prototype

5.3.5 | SOFTWARE INTEGRATION

Once development of the individual components is complete, the application manager has to ensure they may properly interaction. The figure below lists the methods necessary for the three components to interact.

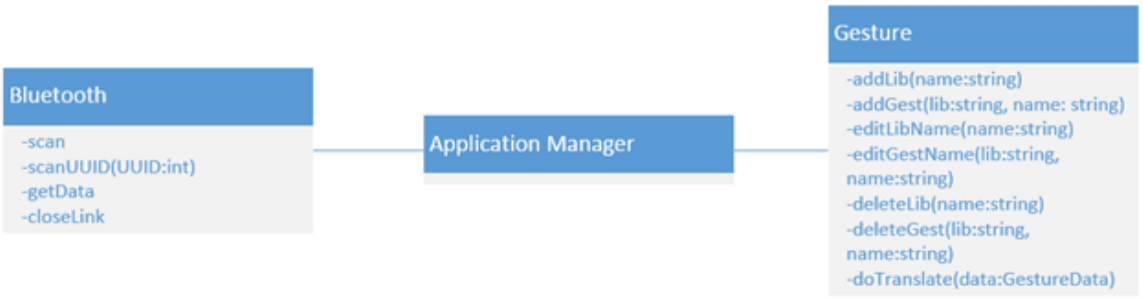


Figure. General Class UML Diagram

The general class UML does not display the intricate methodology that supports the component. Excluding the GUI, the application does not need additional methods to interact with the Gesture and Bluetooth units. Instead, it triggers from the events from the GUI and executes the appropriate procedure.

5.4 | UNIT TESTING

Prior to performing full system test, there is testing at the individual units of code or at a programming module, known as unit testing. The method proves especially useful when developing new technologies. The design of the gesture recognition may be tested when exposed to unseen data after it learns a gesture. The integration of the Bluetooth would be simplified if a test module successfully receives service data from the adapter. Though unit testing may not catch all issues, it will certainly reduce them. The following lists the test cases to check against for the different units of code:

Bluetooth

- Can it handle no device advertisement in scan?
- Can it handle no device advertisement given a service UUID to scan for?
- Can it handle invalid service UUID?
- Can connection transmit service data correctly?
- Can it handle connection breaks?

Gesture model

- Can it handle negative accelerations?
- Can it learn from one gesture units?
- Can it learn from multiple gesture units?
- Can it handle inconsistent sample data?

7 | ADMINISTRATIVE CONTENT

7.1 | PROJECT BUDGET

The Boeing Company sponsored part of our project's expenses. In the initial estimation of High 6, the group based the cost projection from a similar project during the semesters of Fall 2012 to Spring 2013, The Grid. In comparison, this project required a special set of sensors to capture additional dimensions of hand movement. The differences were minor, but the cost of this project estimated to be around $518. Since the initial estimate, the group lacked a device with Bluetooth 4.0 capability. To meet the demands of our project requirements, the project allocates an additional $250 budget out of the groups' personal funds for a tablet. Hence, group 6 has a project budget of $250 for a mobile device equipped with Bluetooth 4.0 that was not covered by our sponsor, Boeing.

DESCRIPTION	QTY	PRICE
Flex sensors	11	$88.00
3-Axis Gyro/Accelerometer	1	$13.00
Pressure sensors	4	$75.00
Glove	1	$20.00
Bluetooth module	1	$12.00
Li-Ion Batteries	2	$12.00
Recharging Battery Dock	1	$15.00
2-Layer PCB	1	$33.00
PCB	1	$200.00
Microcontroller	1	$50.00
Tablet equipped with Bluetooth 4.0	1	$250.00
Boeing sponsorship	**-**	**- $518.00**
TOTAL	**-**	**$250.00**

Table __. Project Budget Breakdown

7.2 | Milestone Gantt Chart

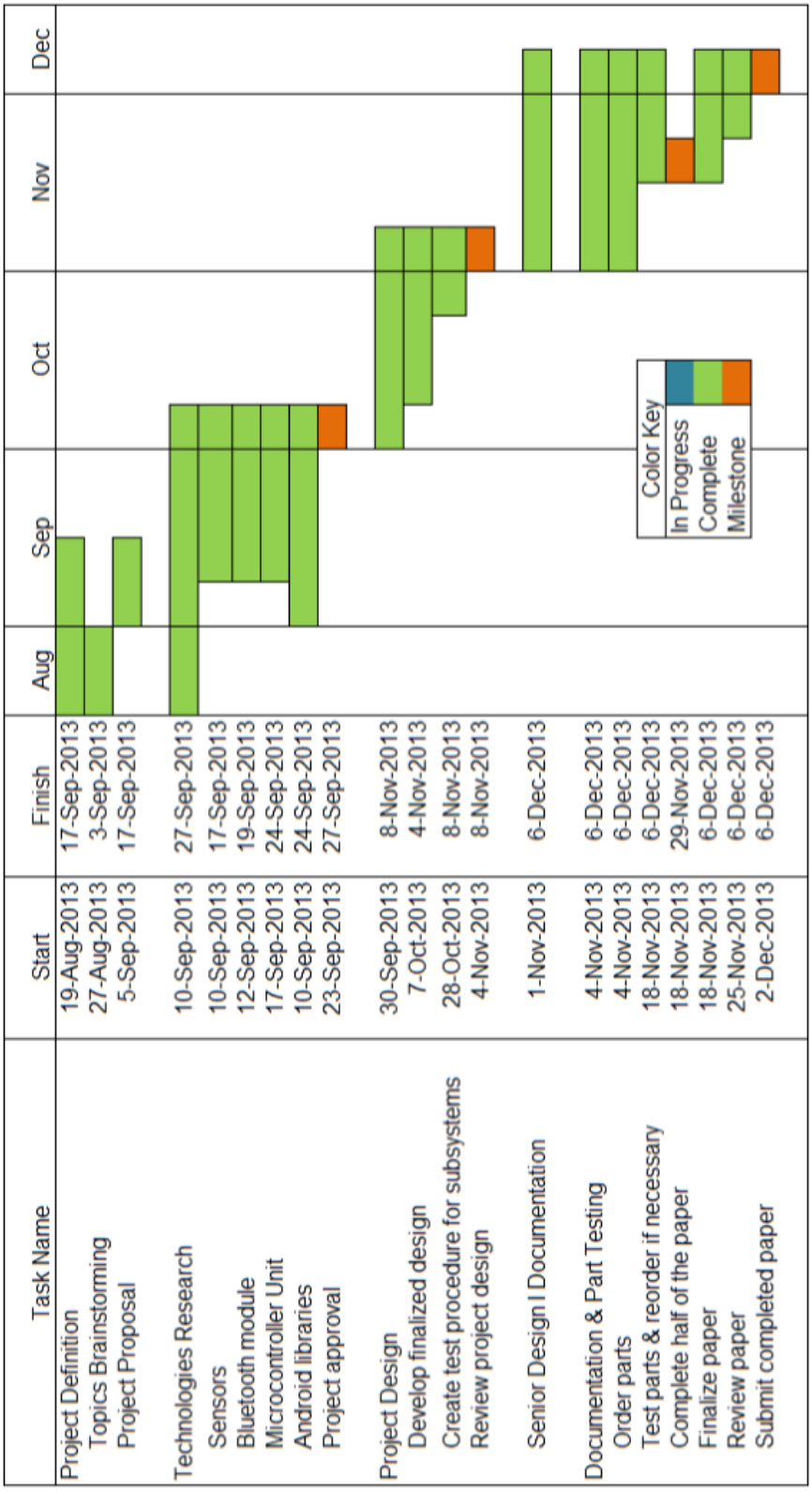
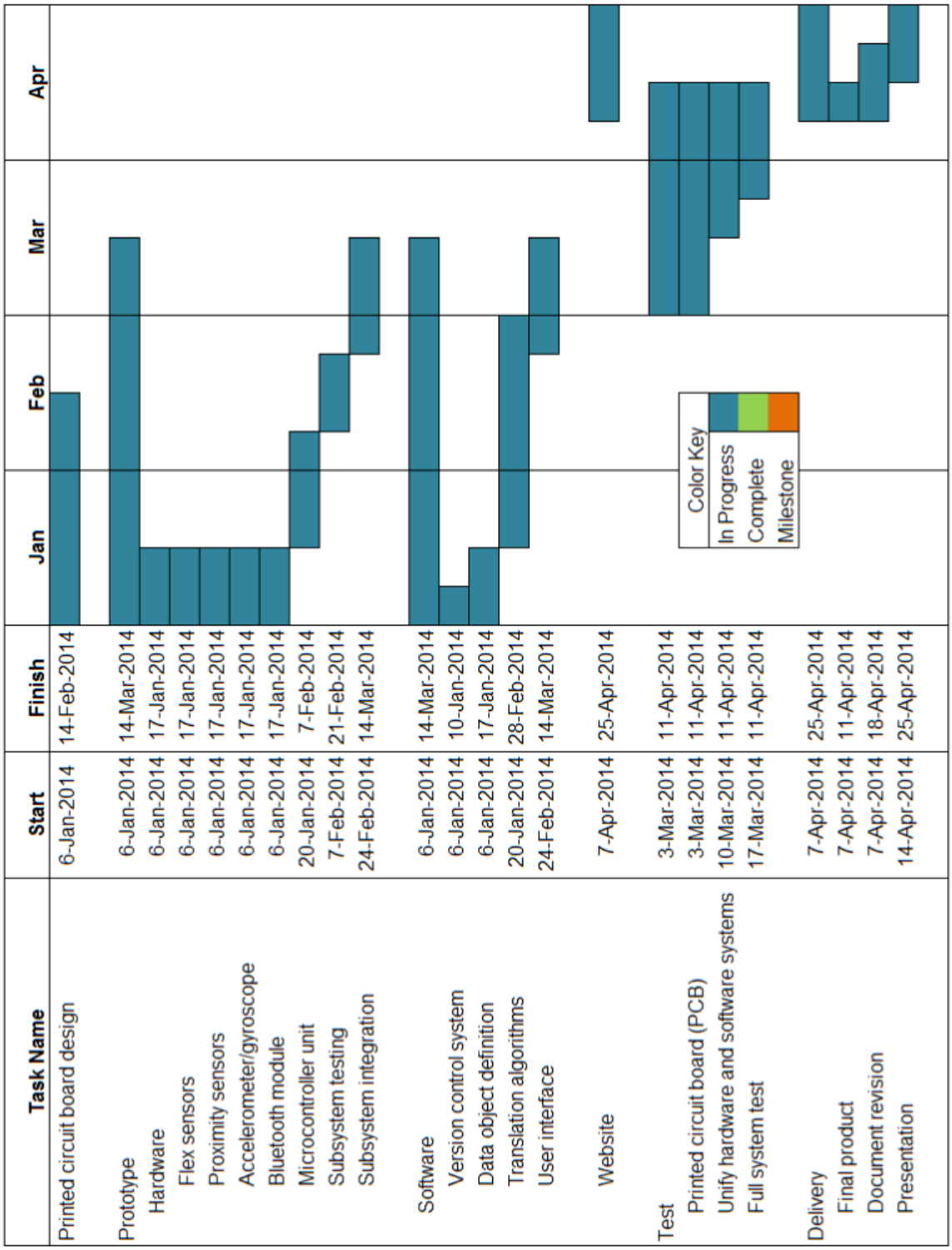


Figure __. Fall Semester Project Schedule



Task Name	Start	Finish
Printed circuit board design	6-Jan-2014	14-Feb-2014
Prototype	6-Jan-2014	14-Mar-2014
Hardware	6-Jan-2014	17-Jan-2014
Flex sensors	6-Jan-2014	17-Jan-2014
Proximity sensors	6-Jan-2014	17-Jan-2014
Accelerometer/gyroscope	6-Jan-2014	17-Jan-2014
Bluetooth module	6-Jan-2014	17-Jan-2014
Microcontroller unit	20-Jan-2014	7-Feb-2014
Subsystem testing	7-Feb-2014	21-Feb-2014
Subsystem integration	24-Feb-2014	14-Mar-2014
Software	6-Jan-2014	14-Mar-2014
Version control system	6-Jan-2014	10-Jan-2014
Data object definition	6-Jan-2014	17-Jan-2014
Translation algorithms	20-Jan-2014	28-Feb-2014
User interface	24-Feb-2014	14-Mar-2014
Website	7-Apr-2014	25-Apr-2014
Test	3-Mar-2014	11-Apr-2014
Printed circuit board (PCB)	3-Mar-2014	11-Apr-2014
Unify hardware and software systems	10-Mar-2014	11-Apr-2014
Full system test	17-Mar-2014	11-Apr-2014
Delivery	7-Apr-2014	25-Apr-2014
Final product	7-Apr-2014	11-Apr-2014
Document revision	7-Apr-2014	18-Apr-2014
Presentation	14-Apr-2014	25-Apr-2014

Figure __. Spring Semester Project Schedule

APPENDIX A | Permissions Request

RedBearLab <support@redbearlab.com>
Thu 11/28/2013 8:18 PM

mark as unread

Hi Laura,

No problem and thank you for your support.

Best Regards,
RedBearLab

REPLY REPLY ALL FORWARD ...

lalis.rubiete
Tue 11/26/2013 6:17 PM
Sent Items

mark as unread

To: info@redbearlab.com;

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from RedBearLab to use the schematic of the BLE mini from the
website: https://github.com/RedBearLab/BLEMini/tree/master/PCB

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Admin - SourceTech411 <admin@sourcetech411.com>
Wed 11/27/2013 12:53 PM

mark as unread

To: 'lalis.rubiete' <lalis.rubiete@knights.ucf.edu>;

1 attachment

fpga_market~.png

Hi Laura,
We are happy to give you permission to use the image / graph from the article for your project.
Attached is the image for your convenience.
Regards,
Jim Allen

From: lalis.rubiete [mailto:lalis.rubiete@knights.ucf.edu]
Sent: Tuesday, November 26, 2013 12:46 PM
To: editor@sourcetech411.com
Subject: FW: Permission to use Graph from Website

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Source Tech 411 to use the graph of the article "Top FPGA Companies From 2013" from the
website: http://sourcetech411.com/2013/04/top-fpga-companies-for-2013/

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Parimal Patel <parimalp@xilinx.com> on behalf of Xilinx University Progra

mark as unread

Wed 11/27/2013 10:21 AM

To: lalis.rubiete <lalis.rubiete@knights.ucf.edu>; Xilinx University Program <xup@xilinx.com>;

Hi Laura,

You may use the snapshots of the table in your report by mentioning that Screenshots are copyright of Xilinx, Inc. and reproduced with permission.
Regards,
XUP Team

From: lalis.rubiete [mailto:lalis.rubiete@knights.ucf.edu]
Sent: Tuesday, November 26, 2013 12:54 PM
To: Xilinx University Program
Subject: Permission to use a Table from Website

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Xilinx All-Programable to use the table of Xilinx FPGAs from the website: http://www.xilinx.com/fpga/

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Vaughn Betz <vaughn@eecg.toronto.edu>

mark as unread

Tue 11/26/2013 5:00 PM

Hi Laura,

Sure, that is fine.

Vaughn

REPLY REPLY ALL FORWARD ...

lalis.rubiete

mark as unread

Tue 11/26/2013 3:18 PM

SD permission

To: szedo@mmt.bme.hu;

- You forwarded this message on 11/26/2013 3:29 PM.

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Professor Gabor Szedo in the Department of Measurement and Information Systems at the Technical University of Budapest to use the SRAM based FPGA picture from the website: http://home.mit.bme.hu/~szedo/FPGA/fpgahw.htm

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Hi Laura,

Please proceed and use the diagram.

Regards,
Gerard

Dr. Gerard Edwards
Associate Member of Institute for Renewable Energy and Environmental Technologies (IREET)
AMI MSc & MSc System Engineering & Engineering Management Course Leader
Engineering, Sports and Sciences Academic Group
Office No. B1-4b
University of Bolton
Deane Road
BL3 5AB
United Kingdom
Tel. (01204) 903865
E-mail: ge3@bolton.ac.uk

From: Holland, Linda
Sent: 27 November 2013 08:40
To: Edwards, Gerard
Subject: FW: Permission to use Website Table

From: lalis.rubiete [mailto:lalis.rubiete@knights.ucf.edu]
Sent: 26 November 2013 20:35
To: ami
Subject: Permission to use Website Table

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from the University of Bolton to use the table of the Technology Programming Summary from the website: http://www.ami.ac.uk/courses/ami4460_fpga/u02/

Brandon Crick <Brandon.Crick@cadex.com>
Tue 11/26/2013 4:23 PM

mark as unread

To: lalis.rubiete <lalis.rubiete@knights.ucf.edu>;

Action Items + Get more apps

Hello Laura, that would be fine and we just ask that you reference BatteryUniversity.com and the author, Isidor Buchmann.

Good luck with your project!

Brandon Crick | Marcom Manager
Cadex Electronics Inc.
22000 Fraserwood Way, Richmond BC V6W1J6
604.231.7777 x319 phone | 604.231.7755 fax | www.cadex.com

>>> On 26/11/2013 at 1:01 PM, lalis.rubiete <lalis.rubiete@knights.ucf.edu> wrote:

> Hello,
>
> My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from the Battery University to use the table of the six most commonly used rechargeable batteries from the website: http://batteryuniversity.com/learn/article/whats_the_best_battery
>
> This paper will not be published and it will only be used for educational purposes.
> Your permission will be greatly appreciated.
>
> Thank you,

TEXAS INSTRUMENTS

Products | Applications | Design Support | Sample & Buy

TI Home > Semiconductor Technical Support > Product Information Center >Email Semiconductor Technical

Thank you for your interest in Texas Instruments Semiconductor products and services.

Your request has been sent to:

Texas Instruments Semiconductor
Product Information Center - Americas.

Prefix:	Miss
First Name:	Laura
Last Name:	Rubio-Perez
Job Title:	Student
Company:	University of Central Florida
E-Mail:	lalis.rubiete@knights.ucf.edu
Phone:	9547298849
FAX:	9547298849
Country:	USA
Address1:	2223 River Park Cir
Address2:	Apt 324
City:	Orlando
State:	FL
Postal Code:	32817
Part Number:	n/a
End Category:	General Information
Application:	Other
Design Stage:	Prototype
Production Quantities:	n/a units
Production Date:	n/a

Problem:
Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Texas Instruments to use the Figure 5: "Typical C-V Charge Profile" from the website:
http://www.ti.com/lit/an/snva557/snva557.pdf

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Laura Rubio-Perez
University of Central Florida
Electrical Engineering and Computer Engineering Student

Permission to use Mechanical and Electrical Table of Flex sensor found on datasheet

REPLY REPLY ALL FORWARD ...

mark as unread

lalis.rubiete
Sun 12/1/2013 5:19 PM

To: support@spectrasymbol.com;

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Spectra Symbol to use the Mechanical and Electrical Table of the 2.2" flex sensors from the website: https://www.sparkfun.com/datasheets/Sensors/Flex/flex22.pdf

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Laura Rubio-Perez
University of Central Florida
Electrical Engineering and Computer Engineering Student

Permission to use Website Alphabet and Number charts

REPLY REPLY ALL FORWARD ...

mark as unread

lalis.rubiete
Sun 12/1/2013 5:04 PM

To: cedir@indiana.edu;

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from the Indiana University Bloomington to use the alphabet chart and numbers chart from the website: http://www.iidc.indiana.edu/cedir/kidsweb/amachart.html

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Laura Rubio-Perez
University of Central Florida
Electrical Engineering and Computer Engineering Student

Permission to use Picture from Website

REPLY REPLY ALL FORWARD ...

mark as unread

lalis.rubiete
Sun 12/1/2013 6:00 PM

To: info@ziffdavis.com;

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Ziff Davis to use the graph from the definition of antifuse from the website: http://www.pcmag.com/encyclopedia/term/37821/antifuse

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Laura Rubio-Perez
University of Central Florida
Electrical Engineering and Computer Engineering Student

Permission to use Picture from Website

REPLY REPLY ALL FORWARD ...

mark as unread

lalis.rubiete
Sun 12/1/2013 6:00 PM

To: info@ziffdavis.com;

Hello,

My name is Laura Rubio-Perez and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I am interested in receiving permission from Ziff Davis to use the graph from the definition of antifuse from the website: http://www.pcmag.com/encyclopedia/term/37821/antifuse

This paper will not be published and it will only be used for educational purposes.
Your permission will be greatly appreciated.

Thank you,

Laura Rubio-Perez
University of Central Florida
Electrical Engineering and Computer Engineering Student

FW: Permission to use Website Alphabet and Number charts

DELETE REPLY REPLY ALL FORWARD ...

mark as unread

Wray, Christina Creech <ccwray@indiana.edu>
Mon 12/2/2013 8:19 AM

To: lalis.rubiete@knights.ucf.edu;

Hi Ms. Rubio-Perez,

Please feel free to use the sign language chart in your project. It is released with a creative commons license that only requires that it not be used for commercial purposes and that you provide attribution to us in your project.

Sincerely,

Christina C. Wray
Librarian
Center for Disability Information and Referral
Indiana Institute on Disability and Community
812-855-0077
www.iidc.indiana.edu/cedir

mark as unread

braintroili
Mon 12/2/2013 11:25 AM

To: webmaster@ti.com;

Hello,

My name is Brian Troili and I am an Electrical Engineering student at the University of Central Florida. I am requesting permission to use several of your images for multiple products Texas Instruments produces.

The images will not be published and are for educational purposes only.

Below are the links to the requested images.

Figure 3 of:
http://www.ti.com/lit/ds/symlink/adc081c021.pdf
Figure 8 of:
http://www.ti.com/lit/ds/symlink/cc2540.pdf
Figure 2 and 3 of:
http://www.ti.com/lit/ug/swru271f/swru271f.pdf
Figure 22 of:
http://www.ti.com/lit/ds/symlink/lm2576.pdf

Respectfully,
Brian Troili

braintroili

Mon 12/2/2013 11:34 AM

mark as unread

To: pr@invensense.com;

Hello,

My name is Brian Troili and I am a student at the University of Central Florida. I am requesting permission to use an image from a document on one of your products.

The image will not be published and is for educational purposes only.

The image is the block diagram at section 7.5 with the link below:
http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/PS-MPU-6000A.pdf

Thank you for your time.

Respectfuly,
Brian Troili

APPENDIX B | References

Agile development
http://GUIde.agilealliance.org/subway.html
http://support.planbox.com/knowledgebase/topics/34844-agile-project-management-resources
http://www.agilemanifesto.org/principles.html

Roboto
http://developer.android.com/design/style/typography.html

Batteries
http://batteryuniversity.com/learn/article/Nickel_based_batteries
http://batteryuniversity.com/learn/article/whats_the_best_battery
http://playground.ATMega.cc/Main/IntWithHW-PwrSup#PwrStore
Battery Management
http://www.ti.com/lit/an/snva557/snva557.pdf
Microcontrollers
http://www.quora.com/What-are-the-advantages-of-using-an-ATMega-over-just-using-the-underlying-microcontroller
http://ATMega.cc/en/Main/ATMegaBoardLeonardo
http://www.glitovsky.com/Tutorialv0_2.pdf
http://homepages.ius.edu/RWISMAN/C335/HTML/MSP430/Chapt4.pdf
Microcontroller's Architecture
http://www.mikroe.com/chapters/view/65/chapter-2-8051-microcontroller-architecture/
http://www.g-w.com/pdf/sampchap/9781590702079_ch26.pdf
http://www.circuitstoday.com/basics-of-microcontrollers
http://www.robotplatform.com/electronics/microcontroller/microcontroller.html
Sensors
http://www.education.rec.ri.cmu.edu/content/electronics/boe/ir_sensor/1.html
http://www.wisegeek.org/what-is-a-temperature-sensor.htm
http://www.seeedstudio.com/wiki/Grove_-_UV_Sensor
http://www.clear.rice.edu/elec201/Book/sensors.html
http://www.analog.com/library/analogdialogue/archives/40-10/cap_sensors.html
http://www.engineersgarage.com/articles/sensors?page=6