Google Glass Remote Monitoring

EE 496 Final Report
Fall 2014

Department of Electrical Engineering
University of Hawaii

Kevin Hu

December 21, 2014

Faculty Advisor: Professor David Garmire

1. Project Objectives and Criteria

Technology is changing every day. As future computer engineers, we must be able to adapt any and all new forms of technology as the years advance. This project is the quintessence of computer engineering, not only in adapting to new technologies but also covering a plethora of computer engineering topics, including hardware, software, client-server wireless communications, and user interface.

This project works as part of a large quadcopter project consisting of two other teams. One team has been working on the hardware, including antennas and power distribution of motors; the other has been focusing on the software portion of motor control including takeoff, hovering, and landing. Our team looks into hands-free remote monitoring using the new *Google Glass*, a wearable technology based off the Android operating system (OS), and a *GoPro* camera module, which will be mounted onto said quadcopter. Using sensors in the Google Glass hardware, changes in head position are to be transmitted to an Arduino, which will move motors on a Gimbal. Streaming video will be transmitted through WiFi to give the pilot a first person view of the quadcopter, emphasizing the need for ease on user interface or UI.

Our work consists of a unique combination of both hardware and software, having the software physically control the GoPro via circuitry. This project has multiple real-life applications, including uses for military, police, and firefighters. This project would allow implementing a fast and easy method to view inaccessible areas.

2. Previous/Related Work and Differentiation

This project is different from other senior capstone projects in its usage of new technology, while still relying on experience from previous technology, projects, and classes. Google Glass is a new wearable

technology that had only been released in 2013 to developers (to the general public in May 2014). Due to its price of $1500 per pair and its recent release, answers to common questions concerning programming for the Glass are still scarce; very few questions with answers can be found on *Stack Overflow*, a website forum open to answering computer programming questions, and its questions still involve simply becoming accustomed to the Glass such as running the standard HelloWorld app [1]. Even additions to Google's documentation are still being release on a regular basis on their website [2].

However, programming apps for the Glass, i.e. Glassware, is still convenient because Google Glass actually uses the same OS, Android, as mobile smartphone applications. The developer workflow has not changed, and access to Android libraries including usage of its sensors is the same; the Glass simply has an add-on of APIs, GDK or Glass Development Kit, to the general Android SDK or Software Development Kit. Fortunately, all three of us members of this group project have had experience developing our own Android applications, so adapting to the Google Glass was not too demanding. This GDK add-on package simply covers gestures used for the touchpad, voice recognition, and its user interface via cards and immersion, described in section 3.2. Because this wearable technology is still new, this forces us to adapt to changes as more tools are being developed – a necessary skill for any engineer, but especially a computer engineer.

This project is also similar to the renowned Micromouse project in this university, an autonomous robot mouse which attempts to reach the center of a maze. Its demand for building hardware from scratch and programming to control motors is consistent with our project. However, this project has been consistently done throughout several decades and thus the internet is saturated with all the information needed [3]. Working on this project does have its benefits however; given all this information provided, engineers can focus on the more important matters such as innovation for obtaining the fastest and most

efficient mouse, the reason for annual competitions within the school, regionally, nationally, and even internationally.

3. Final Design, Design Methodology, and Alternate Solutions

To effectively complete this project, we first divided up the project into four goals: create an effective UI, establish a wireless connection between the Google Glass and the GoPro, control the GoPro's angling direction via the Google Glass, and effectively stream video from the GoPro to the Google Glass. While I worked on the UI and the video streaming, the other two members of our team worked on the other two goals individually.

3.1. Materials and Design Tools

Several materials besides the Glass and GoPro are needed for success of this project. The total materials needed include, in addition: 3-axis Handheld Gimbal, RN-XV WiFly Module, and an Arduino processor. The Gimbal is used not only to hold the GoPro camera but also to receive voltage signals from the Arduino. The WiFly is used to establish a TCP/IP access point for both the Arduino and Glass to connect to, so the Glass can send its sensors' data to the Arduino. Currently the WiFly works with connections only for up to 30 meters, which is sufficient for our testing purposes, but if desired, an antenna can easily be attached to extend the range up to 300 meters.

In addition to these hardware components, several design tools were also needed to get the job done. *Android Studio* is an integrated development environment (IDE) which is most recommended for Glassware programming for its ability to instantly create files needed for a new Glass project. This IDE is similar to *Eclipse*, another IDE ubiquitously used for Java programming, which the Android (OS) is based off of. Eclipse is also commonly used to program Android smartphone applications via a downloadable add-on. However, Google is now attempting to lean toward creating a new environment specifically for

Android programming, and thus released Android Studio instead of releasing another Eclipse add-on for the Glass.

Github was also used as the main source of version control. Many industries use Github not only to be able to revert back to older versions of code but also for individual programmers to merge their code together into one main project repository. Similarly, the three of us in our team were easily able to share and edit code by pushing and pulling code via Github.

3.2. User Interface

The UI for the Glass was not difficult; Google actually had great documentation concerning effective UI [4]. Understanding how we can most effectively implement the UI requires understanding of the general use of the Glass, also well-documented on their website. Because this is a new product, reading the documentation was a requirement to success, without being able to ask other students or faculty on their own experiences.

Because this is a wearable product that is meant to be worn about all day, the Glassware is normally designed to be as minimal as possible, "engaging functionality that supplements the user's life without taking away from it" [4]. However, because the intention of this Glassware is for one to stay stationary while being fully immersed in controlling the quadcopter and GoPro, multiple activities would still work well.

The Google Glass runs on individual screens per Glassware called "cards," lying on a timeline which the user can scroll left to right with the touchpad. In the middle of the timeline consists of a home card simply displaying the present time. The other cards consist of either "static" cards, which simply display text, images, and videos, or "live" cards, which deals with real-time information. Older information (the

static cards) appear farther to the right of the home card, while updated information via live cards appear farther left. Most Glassware requires only one card per app.

Our Glassware however will require multiple activities, for turning on sensors, controlling the GoPro, etc. Google Glass fortunately provides the ability for Glassware to temporarily take over the timeline, simply returning to it by swiping down on the touchpad (similar to pressing the home button on a smartphone). This is called "immersion" and is perfect for Glassware that requires more attention.

For our immersion-based Glassware, we begin with the command, "Ok glass, show client." "Show Client" is the voice command selected – one that must be picked among about a hundred of preset options. Alternatively, one can swipe via the touchpad to the card that says "Show Client." Tapping the Show Client card or saying the voice command leads to the splash screen which waits for the user to say or tap the card "Start Client," as shown in Figure 1.

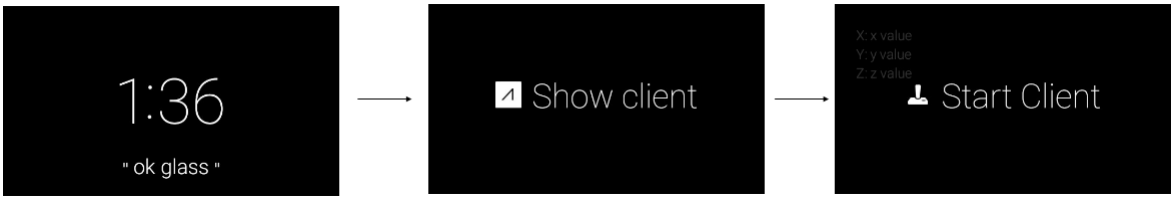

Figure 1. Initialization of Glassware.

After tapping "Start Client," the Glassware will attempt to connect to the Arduino via WiFly. If successful, the user will have the option to turn on and off the sensors. The default sensors turned on will be both pitch and yaw, or in the x and y axes. Menu options will also allow the user to utilize the pitch or yaw individually. Tapping the touchpad will allow access to this menu.

The decision to turn on and off sensors, and selectively choose axes, was based on the fact that the Glass heats up extremely quickly with standard use, after about two minutes. The user him/herself can easily feel the heat, and the Glassware will indicate a warning saying to stop wearing the Glass. In

addition, the heat of the Glass would cause extreme latencies in sending data over to the Arduino and Gimbal. While the Glass does send data and the Arduino receives data in the correct order, the timing can be difficult to work with. The GoPro stops changing angle for a few seconds, then it attempts to utilize all the data at once, trying to catch up and return to normal.

The workflow of the Glassware is shown in Figure 2.

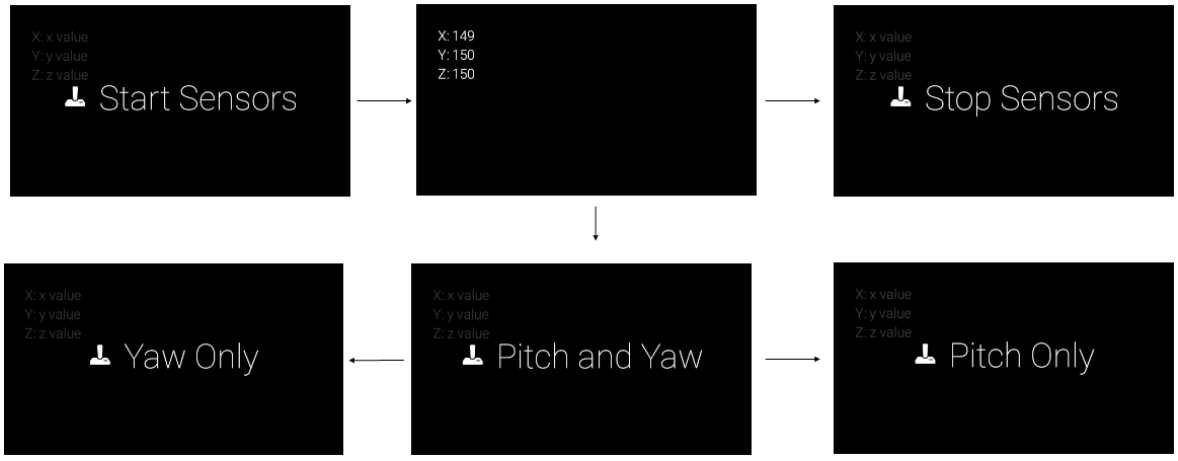

Figure 2. Glassware workflow, starting from top left.

3.3. Data Transmission

The Android APIs provide a very efficient library for wireless connections, including the "java.net.Socket" package. Using this, we were able to connect via the Transmission Control Protocol, or TCP.

TCP is most commonly applicably used as TCP/IP, or Transmission Control Protocol/Internet Protocol, as the basic communication language of the internet. TCP is the most common communication protocol used because it guarantees an established connection, allowing all data not only to arrive at their destination but also to arrive in order. This most effectively is done in bunches of data

called packets, due to fewer bits to resend when an error is detected (detecting in each packet as opposed to each complete file).

Because we had already established what we wanted to do with the user interface as mentioned in section 3.2, we knew that we had to run multiple tasks at once – one task to actively and continually collect data, and one for interacting with the menu. This required us to utilize threads in the Glass, also conveniently provided in the Android APIs' "Thread" class.

To continually collect data without the user interaction interrupting the collection, we created a class called "ClientThread," extending the "Thread" class provided. While this thread is running, it would also continually check if the sensors are still on, which can be turned off at any time in the other UI thread (along with connection to the Gimbal).

Both an IP address and a port number were passed to the "ClientThread" thread to create a socket connection from "java.net.Socket." This allows connecting to the WiFly module containing the given IP address. The connection is then constantly checked; if the connection is interrupted, "Thread.interrupt()" detects this discontinuity and proceeds in error-handling.

Only near the end of the semester did we consider the User Datagram Protocol, or UDP. This is less commonly used than TCP as an alternative. UDP is a best-effort protocol which does not establish a connection before sending data packets. This means that this protocol will not care for errors nor order; if there is an error, the data will not be resent. The main tradeoff for this is speed, not only in not resending but also not establishing the initial connection. UDP is most commonly used applications which do not require sending much data. Because we realized that the Glass heats up so quickly, utilizing UDP may not have been that bad of an option. Our data arriving in a different order or not arriving at all will not leave that large of an impact (especially if the average probability of an error is the standard bit error rate

of 10^{-6}), because the data sent is the average of the sensors' values over time. More will be discussed in the next section 3.4.

3.4. Head Tracking

The accelerometer and gyroscope of the Glass were both initially utilized together to obtain the most accurate information in head tracking, in all three axes. The Android's "android.hardware.Sensor" library provides easy accessibility to these sensors, particularly the "android.hardware.Sensor.TYPE_ACCELEROMETER" and "android.hardware.Sensor.TYPE_GYROSCOPE" libraries.

Firstly, raw data was obtained per change via the method "onSensorChanged()." Every time the sensors' values change, this method would instantly be run. These values would be then stored into a variable. Trigonometry would then be used via a "getAngle()" method to obtain actual angles for roll, pitch, and yaw. These three new values would be added to lists of values every time the method is called, and the average of each list would be calculated. This average would finally be sent to the Arduino for utilization of the Gimbal.

Our first problem encountered in doing this was that when moving the head in one direction and mostly altering one of the roll, pitch, and yaw, then changing the head direction to alter a second one of the three, the data would become skewed. Values appeared random, and the Gimbal would thus shift in angle randomly as well. We originally thought that maybe too many axes were becoming in play, so we decided to temporarily remove the use of roll, implementing only pitch and yaw (the x and y axes). This actually was a success, and is the reason why only Pitch and Yaw are used as shown in Figure 2.

The second problem we had encountered was dealing with the heat of the Glass. We realized that the Glass would become especially hot only when the sensors were on. Since the Glass was using both the

accelerometer and the gyroscope, we knew removing one would essentially half the time the Glass takes to become heated. Removing the accelerometer was our initial thought; if one were to keep their head in an exact position, the methods would return a zero, and the Gimbal would read the zeroes and back to center position. Using only the gyroscope, with only two axes, turned out well after all, although with minor adjustments in the methods' calculations.

3.5. Video Streaming

The WiFly was also intended to, in addition to connecting the Glass to the Arduino, connect the Glass to the GoPro for streaming direct video. We had realized however that the GoPro acts as its own access point. In addition, the Glass, like most electronic devices, can only hold one wireless connection at a time. Therefore the Glass has to either connect to the GoPro for streaming, or the Glass has to connect to the Arduino to control the Gimbal's angle. Because the GoPro is such an accessible item that many consumers use, however, we have been trying to figure out how to figure out a way in still using the GoPro and doing both streaming and remote controlling, but we have been unfortunate in that regard.

If we decided to use a camera that would solve our wireless connection problems, and if we were able to successfully stream video and control the camera at the same time, turning on and off the sensors will be even more essential. Because the Glass already heats up quickly as is, streaming simultaneously would heat up dangerously quickly, and delaying the heat as much as possible would be a priority. However, the mere hardware limitations of the Glass deems streaming to be an unsatisfactory option, so in the meantime simply omitting streaming was deemed to be the best option.

Even simply attempting to stream video in itself was a difficult task. We had wanted streaming as an option now merely for future implementations when a more effective Glass is released by Google. We have discovered that directly streaming video from a uniform resource identifier (URI), essentially the

name of a device, results in an extremely heavy delay of 8-12 seconds. For practical use, this is very inefficient, and may even make the difference between life and death in life-threatening situations. To offset this delay, we wanted to try shortening the video into fewer frames per second, perhaps as low as one to three frames per instead of 30. This required either editing settings to lower the frames per second, editing the video itself as it is streamed, or alternatively automatically capturing multiple images and sending back immediately instead of streaming video. Documentation within the Glass was not provided to determine how to approach any of these three solutions, and because of the current problems with the Glass heating up already due to the mere use of sensors, we abandoned this goal to focus on the others.

In the meantime,we simply utilized the official GoPro Android app for smartphones. This, on a consumer level, easily creates a wireless connection between the phone and the GoPro, using the GoPro's access point and streaming video immediately. While the video does have random latencies outside our control, it is sufficient for our current video needs.

3.6. Hardware

Although this is a software-based project, we still needed to connect all the hardware pieces together. The most difficult part of this was connecting the Arduino to the Gimbal's motors, controlling the exact axes that we want, pitch and yaw. What made this task difficult was the lack of documentation for the Gimbal. We have gone as far as contacting the developers of the Gimbal for documentation, but even that was unsuccessful and they were not able to help. The only solution we could think of then was simply trial and error in connecting wires, which can be extremely dangerous, possibly connecting power to ground for example, creating short circuits and destroying the Arduino or Gimbal.

After trial and error, however, we managed to get the desired circuit, shown in Figure 3.

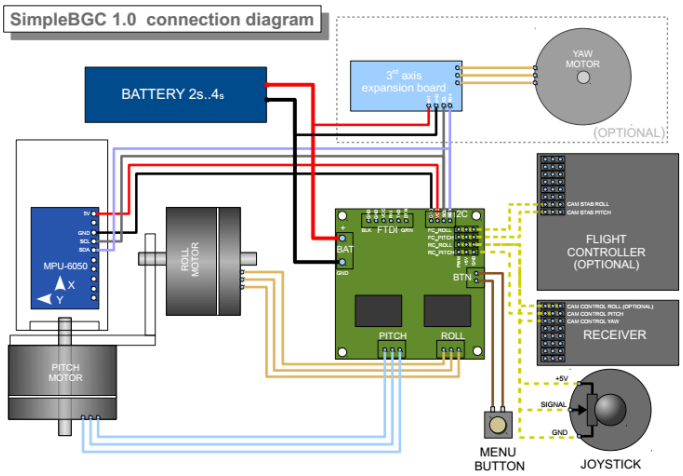

Figure 3. Circuit connecting all hardware components together.

4. Future Work

We have actually managed to reach all of our goals except for streaming video. This would be our main focus to develop in the future, if possible. Otherwise, we will have to rely on the GoPro app, which is inconvenient given the emphasis on UI design in the Glassware.

Although we have met our goal in being able to head track and remotely control the camera, heating is still a major problem, preventing applicable use of this project. Unfortunately this is a hardware limitation that is out of our control, so we will simply have to wait until Google releases a sturdier version of the Glass. They admit that the Glass is still a prototype, so there is still potential for the Glass to do exactly what we want to do without problems.

5. Engineering Standards and Practical Constraints

Everything in engineering has its limitations, even in all fields of engineering. Energy, materials, and safety are always of primary concern. But there are many other standards and constraints to consider when tackling on any engineering project, including economic, environmental, sustainability, manufacturability, ethical, health and safety, social, and political constraints.

5.1. Economic

Quadcopter drones are becoming a great project for many electronic hobbyists primarily due to its economic reasons. Building the quadcopter from scratch simply involves building its wings and chassis, purchasing motors, and an Arduino and a battery. The wings and chassis can nowadays simply be 3D printed for a few dollars at most. Flight motors can cost between $15 to $30 each, estimating about $100 for four total. An Arduino costs about $25, and a battery that will support all this equipment can range from $25 to $50. In summation, about $150, no more than $200, is needed to completely create your own quadcopter from scratch.

If one wishes to record video just like in our project, a GoPro camera would cost at least $130. Effectively mounting a 3-axis Gimbal however costs an additional $300. Even if head tracking were not used, the Gimbal would provide its own stabilization which would be paramount to usable video footage.

The most expensive equipment would undoubtedly be the Google Glass, currently as of December 2014 $1500. If the Glass were to work perfectly with both video streaming and without heating up, then this would be a fantastic deal for practical use. For hobbyists however, they may want to forgo the Glass and perhaps simply use the GoPro smartphone app while using a remote control to angle the quadcopter itself instead of the Gimbal.

For practical uses, the $1500 may be worthwhile for the Glass due to much easier control of the quadcopter. Instead of forcing the quadcopter into an awkward position for the perfect angle, simply change the angle of the Gimbal. This will result in fewer crashing, saving money over time on quadcopter repairs.

In addition, having the streaming video right in front of you makes sure you get the exactly angle you want for the video. This saves on having to repeatedly shooting videos over time, saving money on labor.

Money on human resources is one of the greatest savings for this project as a whole. Instead of requiring people to enter a hard-to-reach location, possibly multiple people at once for safety reasons, simply send the quadcopter out to do the hard work while on person controls the quadcopter remotely. In addition, driving the quadcopter is completely safe for the driver. Because medical bills can be extremely taxing, emphasis on safety also means emphasis on economic savings.

5.2. Environmental

Quadcopters in Hawaii can leave a fantastic impact on our environment. A common news story is hearing about people getting lost while hiking the mountains on these islands. Quadcopters could be used instead of helicopters or rescuers intruding the natural life in these areas, simply maneuvering their way around the forest without reaching contact with any wildlife, including nature.

Quadcopters could also be used to detect levels of air pollution by attaching the appropriate sensors. This can inform people of areas that are high in air pollutions, and actions can take place in order to revert it back to safe levels.

The one concern however is its interference with flying wildlife such as birds. While we do have full control of both the quadcopter and the visual, we can try our best to avoid collision, but sometimes the quadcopter's mere presence can have birds crashing into it. This leaves a negative impact on both the birds and the quadcopter itself, requiring more repair than necessary.

5.3. Sustainability

The materials used as mentioned in section 5.1 are cheap because of the amount of materials they use. In addition, only five volts are needed to power the Arduino and motors attached to the quadcopter, compared to a helicopter which may require many gallons of oil and gas. The battery will only need to be

recharged occasionally. Compared to alternative solutions, this quadcopter drone is undoubted the most sustainable option.

5.4. Manufacturability

As mentioned in section 5.1, because the parts needed to build the quadcopter are so cheap, these quadcopters are extremely manufacturable and in fact are done so already by electronics hobbyists. 3D printing allows almost effortless accessibility to the wings and chassis. Motors and the Arduino can be bought at any electronics store, as well as the wires required to connect the battery to them. With the discovery of 3D printing, its easy manufacturability has exponentially increased the number of quadcopter drones built within the past two years.

5.5. Ethical

Ethical issues are of the biggest concern with using this project. Any piece of technology has its own form of misuse and the quadcopter is definitely not an exception. While the quadcopter can be used for government, military, and firefighter purposes, potentially saving lives and bettering society, there will be people who intend to misuse this product. Such examples of misuse include carrying illegal items such as weaponry and drugs, as well as spying on or stalking people or companies and invading their privacy. However, just because a product has forms of misuse does not mean it should or will be banned, such as cameras in general. Instead, laws have been created to inform society to use products properly. With this camera example, people are simply told that they can only use cameras in private areas with the area owner's permission, that public areas are free for photographers to shoot anywhere, etc. The quadcopter drone is no exception, and thanks to its recent popularity laws will soon be created and enforce to ensure misuse does not occur. These laws do have to be taken into significant consideration however, due to most laws controlling human behavior, not robots and technology. If a drone does approach and spy on

an individual, what will the government laws do about it to take immediate action? Only time will tell as laws are being created, but in the meantime this product is dependent on being used for the greater good.

5.6. Health and Safety

There are a satisfying multiple of ways that this project promotes health and safety. As mentioned in section 5.1, a drone can be use in place of human resources in finding people while hiking on the mountains, for example. This lessens the chance of rescuers becoming lost themselves or attacked by wildlife, as they remotely search for victims with the drone. Rescuers can go in once they find the target, and they can then determine the best path before approaching.

In section 5.2, alternative sensors were mentioned to detect a plethora of other possibilities in addition to merely vision, including air pollution. Having an easily controllable, low-cost drone will allow easy gathering of data, and people can analyze the data and make adjustments to societal needs as necessary, such as reducing air pollution in specific locations.

Additionally, in the previous section 5.5, misuse was mentioned in carrying weapons and illegal drugs. Similarly, if used for the greater good, these drones can carry useful items such as medicine or hand tools, possibly at a great speed without using many resources. Delivering items to someone in a multi-story building, for example, would be ideal for use with this quadcopter drone.

The only concern regarding health and safety would be with collisions. Colliding into buildings should not be a problem because the driver has full control of the quadcopter as well as the camera, but inexperienced drivers may have problems. Additionally, if the drone does run out of battery, it may unexpectedly fall in undesirable places, possibly hitting people or affecting the environment. Both concerns can be avoided if approached with care, however; like in section 5.5, misuse simply depends on the driver.

5.7. Social

Social standards and constraints apply to this project both positively and negatively. Section 5.5's discussion of spying and stalking others marks a strong negative impact in our society socially, especially since its manufacturability and accessibility allows almost any hobbyist to be able to make this project, and even neighbors can be potential criminals in this regard. However, if used for the greater good, drones can be used to deliver necessary items such as medicine to others in dire need, or first aid in hard-to-reach places in post-disaster times. Having the option of being able to help each other out makes this project worth doing for anyone.

5.8. Political

While the primary use is for military and police, the government may also use this project to their advantage. The government may take pictures of areas as necessary and use the data for whatever purpose they may need, such as surveying areas that need more improvement than others or advertising for the next election.

6. Conclusion

This quadcopter project was the perfect way to close my undergraduate engineering career. It entails most if not all topics of computer engineering, it utilizes the latest technology and requires adapting to changes and lifelong learning, and it also has a practical use in everyday life. From hardware to software to networking to UI, such a wide range of topics are covered to improve one's overall skills as a computer engineer. As technology advances and new products come out, we will have to rely on these fundamental engineering skills as we become more innovate and build for our society.

References

[1] (2014, December 14). Stack Overflow. [Online]. Available: http://stackoverflow.com/search?q=google+glass

[2] (2014, October 29). Google Developers: Glass Platform Release Notes. [Online]. Available: https://developers.google.com/glass/release-notes

[3] (2010). UH Micromouse. [Online]. Available: http://www-ee.eng.hawaii.edu/~mmouse/history.html

[4] (2014, July 9). Google Developers: Glass Interface. [Online]. Available: https://developers.google.com/glass/design/ui