Android-Enabled Programmable Camera Positioning System

Sponsor: Dr. D. LeMaster, Air Force Research Laboratory

ECE 480 Design Team 3

Austin Fletcher
Jeremy Iamurri
Ryan Popa
Yan Sidronio
Chris Sigler

Facilitator: Dr. K. Oweiss

Michigan State University

April 25th, 2012

Executive Summary

The purpose of this project was to design and construct an automated infrared camera positioning system utilizing a commercial Android smartphone. The system requirements include the abilities to sense its current location and orientation, and slew between GPS designated targets based on a pre-programmed script. A smartphone was used for its multitude of sensors, processing power, and low cost.

Acknowledgements

Special thanks to Google and their Android team, the Air Force Research Laboratory and Dr. LeMaster for providing us with the support we needed, Dr. Oweiss for taking the time to perform the duties of a facilitator, and Michigan State University for providing us with all of the experience and knowledge required to complete this project.

Table of Contents

Chapter 1 - Introduction and Background

Introduction

Design Team 3 was tasked with developing an Android enabled GPS-based camera positioning system for the Air Force Research Laboratory (AFRL). This entailed constructing a motorized camera mount for an approximately 19lb infrared camera, provided by AFRL, that is controlled by an Android smartphone. This system needed to be able to point at any target GPS coordinate (latitude, longitude, and altitude) on a pre-programmed script to a high degree of accuracy (as high as a smartphone's sensors permit) and reliability. In order to do this, the system must be able to detect its own location and orientation, calculate the necessary orientation so that the infrared camera focuses on its target location, and control the motors in the mount to rotate to that orientation. The phone then signals a host laptop to trigger the infrared camera to capture an image, and take one itself as context imagery. Thus, the phone's camera must be oriented in the same direction as the infrared camera. A typical electrical grid connection will be available to connect the device to, so conserving battery charge is not a primary concern. This project is highly experimental, so the exact requirements were highly variable and dependent on the phone's capabilities.

Background

The AFRL will use this system to automate the positioning of an infrared camera to perform field testing of infrared imaging technology and processing algorithms. Such devices already exist, but the purpose of this project is to develop one at a reduced cost using the sensing and processing power from a smartphone (specifically, we will be using an Android smartphone). Automated motorized mounts similar to this can be used for surveillance cameras or telescope systems. These systems typically do not have a built-in GPS receiver and require this data be input manually. Additionally, in order to obtain the orientation of the device, it must be calibrated by pointing at specific GPS locations (landmarks of some kind) and inputting these coordinates into the device, which it can use to calculate its orientation. After this, it can control the motors precisely or use gyroscopes to calculate its relative orientation to the known setup. Using our design successfully avoids many of these steps, leading to reduced work for the user of the system, and saves money given the low cost of our commercial, mass-produced components.

The Android SDK has shown that an Android smartphone provides the necessary sensors and predefined functions for calculating the system's exact location and orientation. This would make calibration unnecessary, thus improving the automation of the testing equipment. An accelerometer can be used to calculate the direction of gravity, in order to obtain the pitch of the phone. A magnetometer can obtain the direction of Earth's magnetic field, and in conjunction with the accelerometer the phone is able to calculate the azimuthal angle. The GPS receiver on the phone provides the latitude, longitude, and elevation of the system. The Android SDK provides the necessary calculations for obtaining the target orientation, and a phone provides the necessary channels for communicating with the motor controllers and the infrared camera, while still being able to obtain power to stay charged. Additionally it has been shown that an audio jack can be used for data communication and for powering a small circuit. This research reinforced the idea that the motors can be controlled by the audio jack thus saving our very limited wired I/O capabilities.

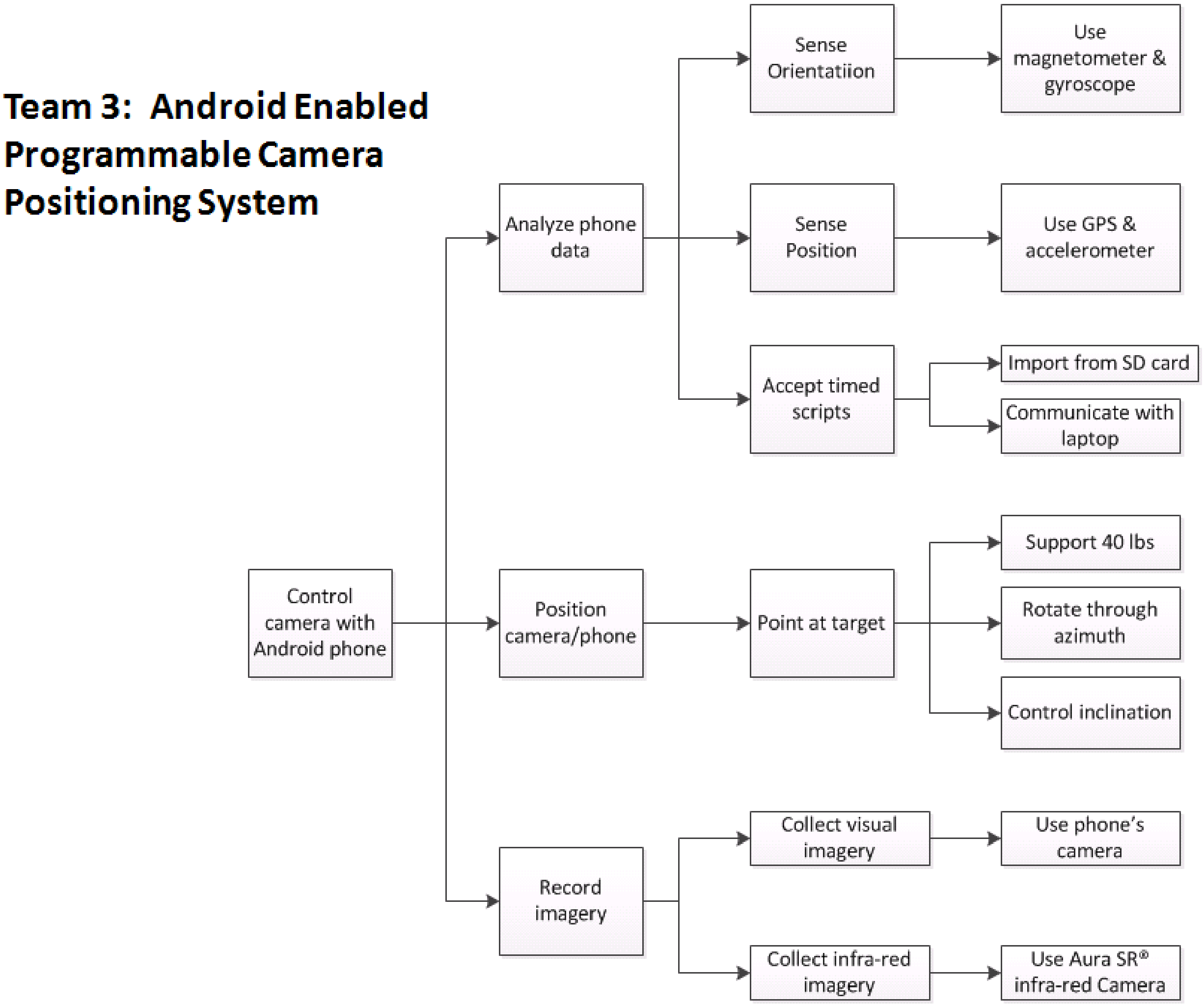
Our group estimates that the following costs should be allocated for a new construction based on our design, but following the suggestions outlined in Chapter 5:

- Android Phone $200
- Power circuitry $50
- Logic circuitry $35
- Hardware and supplies to construct the camera platform $105
- Stepper motors $60
- For a total cost of $450

Chapter 2

Fast Diagram

This Fast Diagram illustrates the desired capabilities of this system:



Design Specification

Sponsor's Requirements

For this project, the AFRL required the team to fully develop and implement an automated camera position system.

1. Positioning - Slewing between target points defined by GPS coordinates at various times according to a pre-programmed test script
    - ○ Azimuthal rotation range of at least 0 to 360 degrees around the camera's position..

    - Polar elevation range of 0 to 90 degrees.
    - Capable of securely rotating and managing a 30 lbs load
    - Capable of being securely mounted above and below the horizon.
2. Coordinate sensing and calculation -Automatically sensing the camera's own position and orientation in order to calculate the required target acquisition geometry.
    - GPS, compass heading, and elevation is to be detected using an Android based smartphone.
    - Taking as much advantage as possible of the computing power available to commercial Android smartphones
3. Imaging - Generating an image acquisition command signal to the attached infrared camera.
    - (Suggested) Using the phone's camera to acquire context imagery in the visual spectrum.
4. Optimizing cost vs. performance.
5. Provide a base for the mount that can be mated to the sponsor's tripod mount, or used as a standalone platform.

Team 3 - Goals

The team took the above as minimum requirements to be achieved and if time allowed, make the following addendums:

1-b. Polar rotation range that exceeds 90 degrees; possibly eliminating the need to physically re-mount the camera to acquire data both above and below the horizontal.

1-d. Producing a mount sturdy enough not only for the camera and it's weight, but the environment as well (ie. rain and wind). This will require a design that can be self-contained or easily covered by a plastic shell (plastic would be for future implementation since it is outside the scope of the equipment available to us).

1-e. (Added requirement) - Produce a positioning mechanism sturdy enough to keep the camera at a designated position, allowing for video acquisition.

2-b. Avoid, if possible, the use of additional micro-controllers and rely as much as possible on the electronics of the android phone for signal generation and handling. This does not include controllers necessary for motor operation.

3-a. Complete suggested requirement.

4. Completing the aforementioned goals within a $500 budget including:

- Android smartphone
- Electric - amplifier components, band-filter components, motors and their control boards (if necessary), transformers
- Mechanical components - gears, bearings, metal-supports.
- Misc. materials - wood, sheet-metal, screws/bolts/washers/etc., wires

House of Quality

See Appendix 4 for the House of Quality. This diagram illustrates the most important design measures for this project. As shown, the most important measures were found to be: Use of the Android smartphone for as much of the required functionality as possible, accuracy of the final picture taken, and reliability of the mount. Using the smartphone's capabilities severely decreases the cost of the system, and accuracy and reliability of the system allow the AFRL to use this mount consistently for their stated purposes.

Solution Selection Matrix

This project presented several aspects which required design decisions. The matrix, shown in Appendix 5, presents the possible solutions for four major design considerations: selecting which type of motor to use, how to communicate between the phone and laptop, how to control the motors with the phone, and how to calculate the orientation of the phone. OrientationSensor and RotationVectorSensor are Android-provided "software" sensors that do the calculations for orientation by combining the accelerometer and magnetometer data automatically.

The bottom row of the matrix shows the total benefit from each possible solution. The highlighted cells show the design solutions we selected, which all had the highest rating from this matrix. In summary:

- Stepper motors were selected for their high holding torque, low error per step, and zero cumulative error.
- USB communication was selected due to its reliability and compatibility compared to bluetooth, and zero cost.
- A PIC microcontroller via the audio jack was selected due to its zero cost (free, provided by ECE 480 lab), the fact that it allowed laptop communication through the USB port on the phone, and increased accuracy compared to analog circuitry.
- OrientationSensor was selected for its lower complexity and better accuracy compared to manual calculations, and due to the RotationVectorSensor having large amounts of noise (>10°), and being very slow (> 1 second between sensor events).

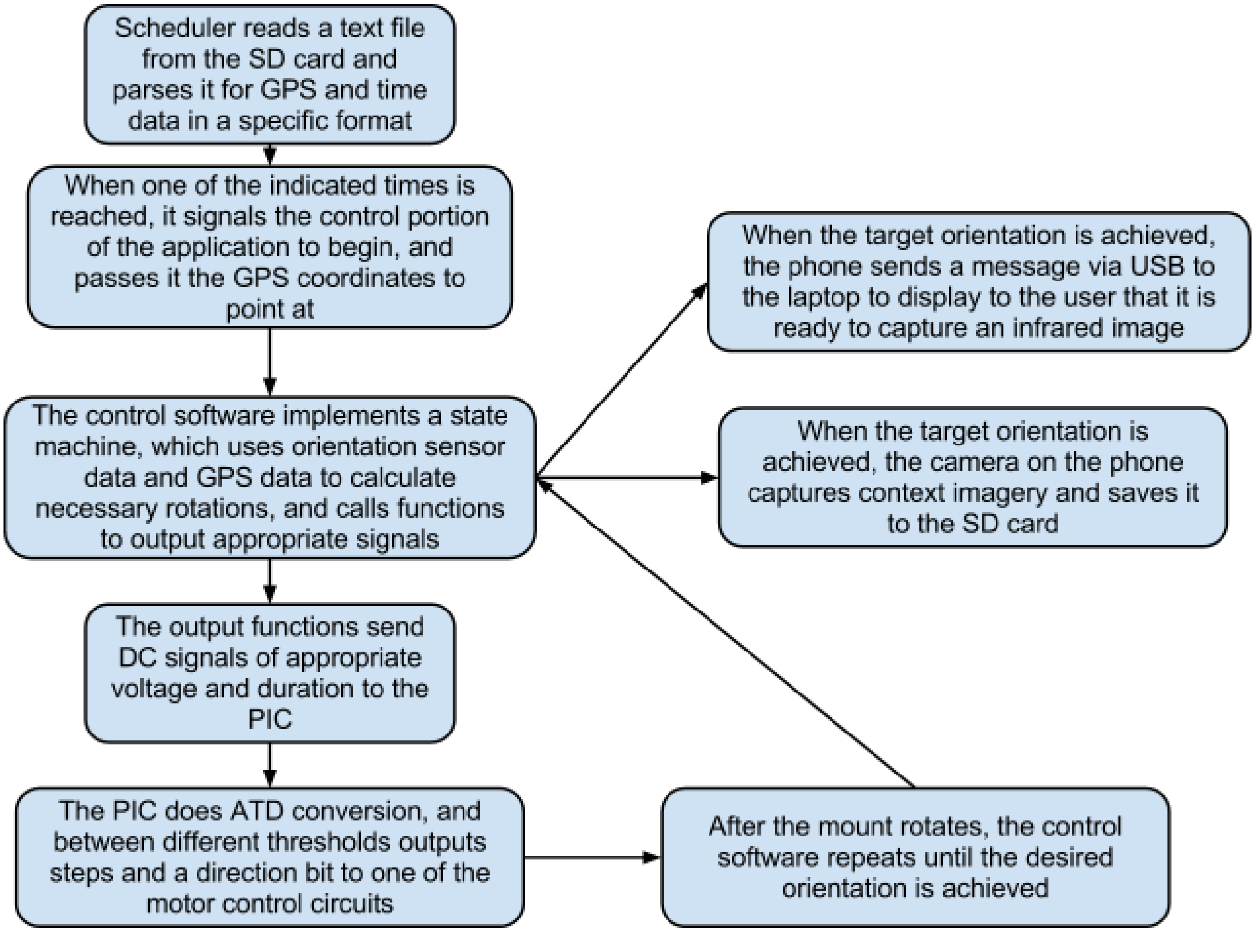
Design iterations throughout the project are described more in detail throughout the rest of the report.

Conceptual Design Description

The project was split up into three distinct design sections. The first is the mount assembly. This section consists of the components needed to hold the camera in place and allow it to rotate both up and down, as well as side to side. The initial design for the assembly can be seen in Appendix 3.

The next design section is the motor circuitry. This section consists of the motors the team chose, the motor controllers necessary to control them, any processing circuitry required to transfer input signals to the motor controllers, and the interface for communicating with the Android phone. The initial design was to try to control the motors through the headphone jack on the phone and use the frequency and amplitude of the AC wave to control the motor's movement and direction. This design went through several iterations, as the team decided to use a PIC microcontroller to interpret the output signals from the audio jack (after they were put through a rectifier) using ATD conversion and to output appropriate signals to the motor control circuitry instead of using analog circuitry. After the phone was tested and found to be able to output DC signals, the design was changed by removing the rectifier circuit.

The final section in our design is the software for the Android phone. The phone first reads from a supplied schedule of coordinates and waits until the time the first scheduled point arrives. When the first scheduled event is ready it calculates the angles of rotation required to point the camera at the desired coordinate. Once the rotation is complete the phone sends a signal to the camera telling it to take a picture. The phone then takes a picture with its own built in camera and stores it to the SD card for reference. The phone then goes to sleep until it is time for the next scheduled event. This process occurs continuously. A flowchart demonstrating our proposed program operation is below.


Scheduler reads a text file from the SD card and parses it for GPS and time data in a specific format
When one of the indicated times is reached, it signals the control portion of the application to begin, and passes it the GPS coordinates to point at
The control software implements a state machine, which uses orientation sensor data and GPS data to calculate necessary rotations, and calls functions to output appropriate signals
The output functions send DC signals of appropriate voltage and duration to the PIC
The PIC does ATD conversion, and between different thresholds outputs steps and a direction bit to one of the motor control circuits
When the target orientation is achieved, the phone sends a message via USB to the laptop to display to the user that it is ready to capture an infrared image
When the target orientation is achieved, the camera on the phone captures context imagery and saves it to the SD card
After the mount rotates, the control software repeats until the desired orientation is achieved

Risk Analysis

While considering various design solutions, some concerns arose. These concerns were broken down into three major categories: the Android phone, power supplies, and the rotating camera platform.

In regards to the Android phone, the primary concern is that few I/O channels are offered by these devices. These channels are further limited by the requirement of only using wired connections for I/O. This limitation leaves two options, the mini-usb port and the 3.5mm audio jack. The reasons for avoiding wireless communications are two-fold. The first reason is to avoid unwanted security risks that are inherent with current wireless protocols. The second is to avoid reliability problems due to range and interference from the motors and other nearby devices.

The power requirements for this system also present a concern, as most off the shelf power supplies are insufficient. The motors selected in the design offer enough torque to move the camera platform, but this comes at a huge power cost. Each motor requires 4 Amps at 3.6 Volts at rest. This means that the power system must be able to provide 8A at 3.6V continuously. To combat this, the power supply and regulation circuitry selected are able to output 15A at 4V (to allow some voltage drops across other circuit components) at maximum. Additionally these supplies require extreme power conditioning circuitry to handle the immense EMF feedback from the motors, making custom solutions too complex for this project.

The motor and mount is another area of the design that posed several concerns. Firstly the torque produced by the motors needs to be enough to rotate the entire 19lbs of the camera as well as the mount, but still rotate at a decent speed. On the other hand, if the mount moves too quickly the gears could have skipping issues. There needs to be a balance between speed, stability, and accuracy. Second, the motors may also generate a significant magnetic field, which could cause interference with the phone’s sensors or other electronics. Due to the stepper motor design, the motors are always powered, meaning this problem may always be present although it is likely the motors are shielded well enough to prevent this. Lastly, the camera itself costs $40,000, so building a robust and secure mount is a top priority.

Gantt Chart

See Appendix 4. The only major difference between the proposed Gantt Chart and the final chart is that testing the motor control circuitry took longer than expected, due to the commercial circuitry purchased not being able to handle the high power. Thus, after burning out four boards, a custom board was built with higher power-handling capabilities. This delayed the rest of the timeline by several weeks and decreased the available time to test the completed system.

Chapter 3

Mechanical Design

The mechanical design was prototyped entirely in Autodesk Inventor ® using 3D CAD models for each component. As a team we gathered in the first week of our design and brought together all of our ideas and sketches in order to decide on a design for our mount. This final design is detailed in the CAD drawings of appendix 3. These CAD drawings show the entire mount as it was planned. Each component was detailed separately so that Computer Aided Manufacturing simulations could be made in order to simulate our mount's motion as well as to confirm calculations about center of mass, moment of inertia and material stress and strain; reassuring this was an adequate design with appropriate materials. The Android phone, nor the circuitry is detailed in these CAD drawings as we were still deciding on these components. However, regardless of what phone we chose, our intention was always to place the logic on the right vertical support (in regards to the infrared camera's point-of-view), the android phone onto the main polar axle to the right of the infrared camera, and the power supply on the back in order to better balance the mount and keep the entire project's center of mass concentric with the azimuthal axis.

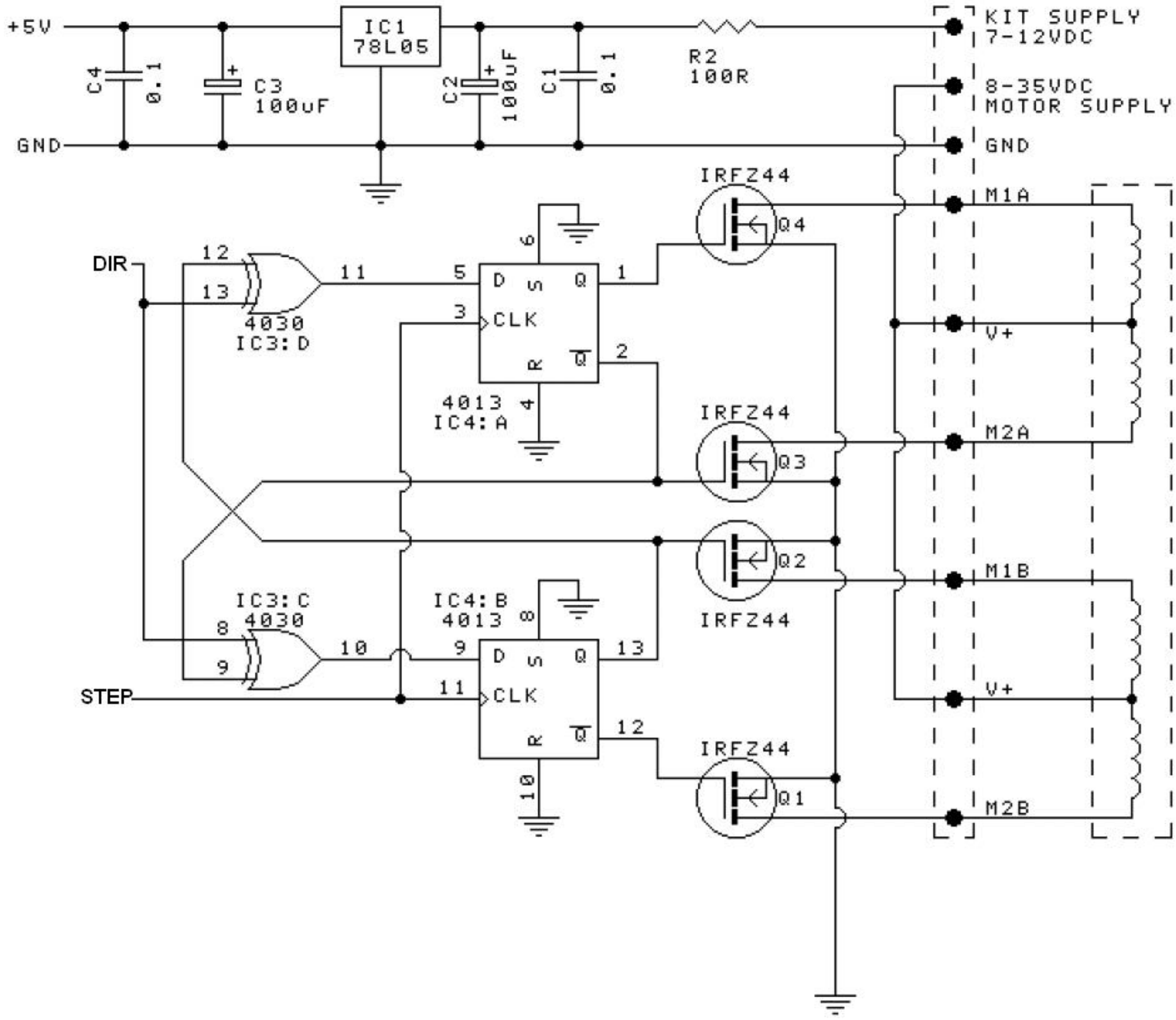
During the entirety of the mechanical engineering design, the circuit and phone costs were taken in consideration. The entire mount was planned to be, and indeed was, built in a \$200 budget; leaving \$300 for our android phone and necessary circuitry/logic. This budget included the motors. Stepper motors were chosen instead of servo motors due to their low cost and reliable, and accurate positioning. The entire system aims to achieve a minimum accuracy of 5^0 in both axis of rotation. We were able to achieve this goal; however some changes were made to the design in the final implementation of our mount.

In the final implementation, which can be seen in Appendix 4, we opted to replace the flange bearing which the mount rotates upon to a "lazy-Susan" turntable bearing. The turntable bearing is much thinner and wider, allowing for less resistance and greater stability of movement. This further assisted us during the semester as or sponsor specified an extra requirement during week six. This requirement was to make a base to our project instead of using a drive shaft, which would be inserted into a camera tripod.

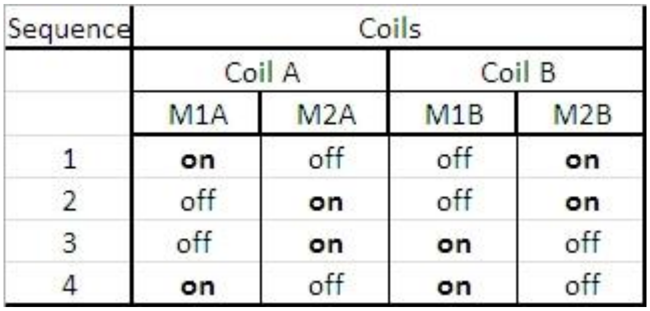
Another design consideration that is observable in our final product deals with cable management. In the bottom of the base, there are two collinear holes, one on the left and another on the right of the base. Through these, up the main shaft and through the azimuthal gear comes both the USB that connects the phone to a host laptop and the main power line. They are placed in different holes so the mount wiring may be mostly self-contained, allowing us to achieve many revolution of rotation without causing cable issues.

Hardware Design

The following figures represent the logic behind the step rotations of the stepper motors. The circuit relies on XOR gates, D Flip-Flops, and N-Channel CMOS transistors to control the current flow of the motors.



The schematic above shows the 5V logic power regulation circuit at the top. This circuit is used to power the Flip-Flop and XOR logic ICs as well as the PIC microcontroller. We can additionally see that due to the configuration of these logic elements that they follow a 4-state sequence. This sequence is outlined in the table below for the Direction bit set at 0.

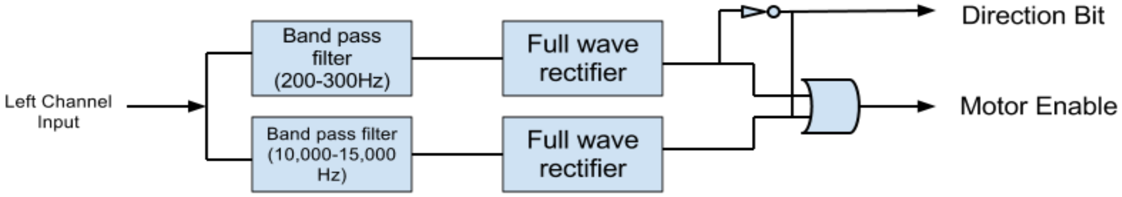


Sequence	Coils			
	Coil A		Coil B	
	M1A	M2A	M1B	M2B
1	**on**	off	off	**on**
2	off	**on**	off	**on**
3	off	**on**	**on**	off
4	**on**	off	**on**	off

This table shows that there are only two coils active at any given step position. Each step only has one coil changing at any given time. If at any time the direction bit changes, at the next step the coil states are inverted, to produce a rotation in the opposite direction.

Hardware Implementation

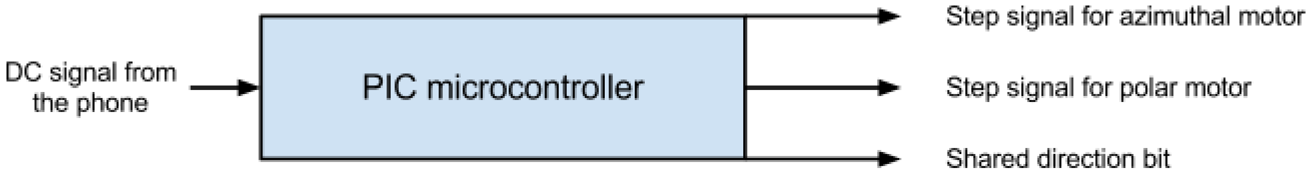
Our first attempt to design the motor control circuitry was designed to be entirely analog. The design was to output different frequency audio signals from the phone's headphone jack that represent different control signals for the motors. Originally we intended to use the left channel of the headphone jack for azimuthal rotation and the right channel for polar rotation. These two channels were connected to two band pass filters which in turn were connected to full-wave rectifiers. The purpose of the filters was to split the incoming audio signal into two separate signals, one to indicate clockwise rotation and one to indicate counter-clockwise rotation. This design is shown as a block diagram below.



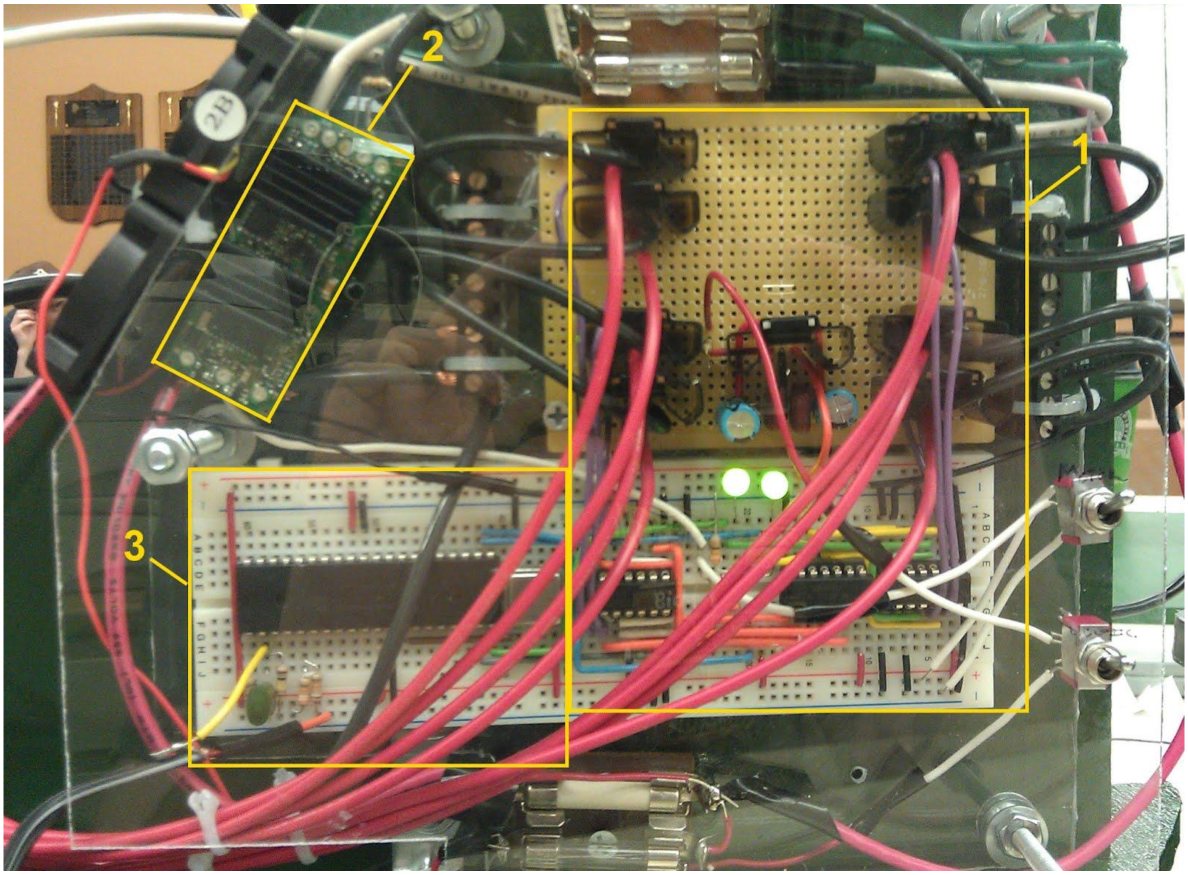
The outputs of this circuit connect to enable pins on a stepper-motor controller. The motor controller was to be powered by a separate clock that was constantly sending it a step signal, therefore the only way to stop a motor from rotating was to turn off the enable pin by ceasing audio output on the channel.

This design proved to be hard to implement due to the band-pass filter design not providing the level of accuracy we were expecting. This design was ultimately abandoned for a design using a PIC microcontroller to do the audio filtering. The new design used the voltage of the audio signal rather than the frequency. This means the PIC microcontroller was acting as a basic voltmeter and basing the motor control signals on the different voltages that it sensed. The full-wave rectifier was attached to the sense pin of the microcontroller and converted the AC audio signal into a DC voltage the microcontroller could measure. The main advantage of this design over the previous design is that more of the logic could be implemented in a digital microcontroller which allowed for much more predictable results than we were able to achieve with the analog design. The second advantage of this design is the accuracy in which we could measure the DC voltage. Before, the solution for large pass bands on the filters was to split the two rotations into two different channels of audio. With the accuracy of the voltage measurement used in the new design we were able to use only one audio channel for all 4 different motor control signals.

The phone can output a DC value between about -1.3V and 1.3V. Since ATD conversion is being done on this signal, voltages below 0V could not be used, as the reference voltages are 0V at minimum. For better resolution, a voltage divider was used to obtain about 1.5V as the upper reference voltage (instead of 5V from the power rail). At even spacing between digital values for 0V and 1.3V, four thresholds were set. This created five possible interpretations for signals: Do Nothing, Left, Right, Up, and Down. For each of the last four options, the PIC outputs a 25.6Hz square wave to act as a stepping clock to the appropriate motor, and asserts or de-asserts the direction bit as needed. This new design merged the stepping clock with the control logic, as opposed to the external clock that was in the first design. We later discovered that the phone could output a DC signal and there was no need for the rectifier. This relieved the final issue we were having with the control circuitry, which was the rectifier’s capacitance holding onto the DC voltage after the phone stopped outputting any signal. With the phone connected to the microcontroller directly the response time between the phone signals and the motor movement is almost instantaneous. The block diagram for the final design is greatly simplified.



The figure below shows the final control circuitry for our project. It is broken down into three distinct areas.



Area one consists of the power N-channel MOSFETs, the 5 volt power regulator for the logic circuitry, the XOR gates, and the D Flip-Flops. Also seen in this area are two status LEDs showing that both the logic and motors are currently powered. This circuit accomplishes the tasks presented in the previous section.

Area two comprises of a 5v Hammerhead™ EHHD015A0A DC-DC converter that is capable of providing up to 15A to power the motors. The power from this converter is routed through two 8A fuses, in parallel, which is then routed to each motor’s control MOSFET logic. This converter has had its output voltage modified to be in the range of 4v, by attaching a 15K ohm resistor between its sense pins. This is set such that the motors, when stationary, will draw approximately 3.15A for each coil phase of coils that is connected, ignoring any other voltage losses. This is due to the 1.3 ohm resistance that each phase of the motor coils presents.

Area three is the circuit necessary for the PIC microcontroller. On the bottom edge there is an RC filter circuit to minimize the noise coming from the headphone jack. This output is then fed into the analog-to-digital converter so that the PIC may control the step and direction output to feed into the motor switching logic in area one.

Android Smartphone

The design team selected a Motorola Droid® for this project. This phone was selected because it contains the necessary sensors for this project and could be obtained at a very low cost due to being an older model.

Software Design and Implementation

The software was broken down into a variety of different tasks. The overall software flow was described in Chapter 2; this section describes each of the components in more detail.

Location and Orientation Sensing

The Android SDK provides easy functionality for finding the GPS location (latitude, longitude, and elevation) of the phone. For calculating the orientation of the phone, there are multiple options. Raw accelerometer and magnetometer data can be used to manually calculate the orientation of the phone, however this option is prone to programmer error and was discarded after finding Android’s “software sensors.” Android provides two such sensors for finding orientation: OrientationSensor (OS) and RotationVectorSensor (RVS). OS was the original orientation sensor provided in the SDK and was deprecated in version 2.2, in part due to high error when the phone is tilted sideways. RVS was introduced in version 2.3, and is intended to have more accurate calculations than OS. However, after being tested on the phone purchased for this project, it was found that RVS’s calculations are too complex for the phone to do in a reasonable time. RSV updated at lower than 1 Hz, while OS was over 10 Hz. Additionally, the phone is mounted upright on a mount which is to be placed on a tripod designed to be level at all times, so the phone will never tilt sideways, negating OS’s weakness. For these reasons, OS was selected as the source of the phone’s orientation.

Audio Jack Output

The Android SDK provides a class (AudioTrack) which can be used to output a user-defined signal through the audio jack, which is precisely the requirement of the chosen design. To implement this, an array of bytes is written to a buffer, which then is output through the audio jack at a user-defined sampling rate. This class can operate in two modes: static and streaming. Static is designed for low duration signals, and only allows a single array to be written to the buffer at a time and then this single array is output continuously. Arrays in java take type int as their index, which only allows a maximum size of 32767 bytes. Due to the low rotational speed of the mount, the required array size to output the maximum duration possible exceeded this maximum. This left streaming mode as the other option. In this mode, data may be continuously written to the buffer for output; however, the output is not guaranteed to be continuous. When this operation is done in the main application thread, the output is continuous, however the output function blocks during output, freezing the application long enough to cause Android to forcefully close the application. In order to prevent this, the output is done in a separate thread, which can be blocked without causing any issue, however in this case the worker thread does not always run enough to keep the buffer full and so sometime the output has discontinuities. This results in the correct duration in total being output still, but split over many discrete times.

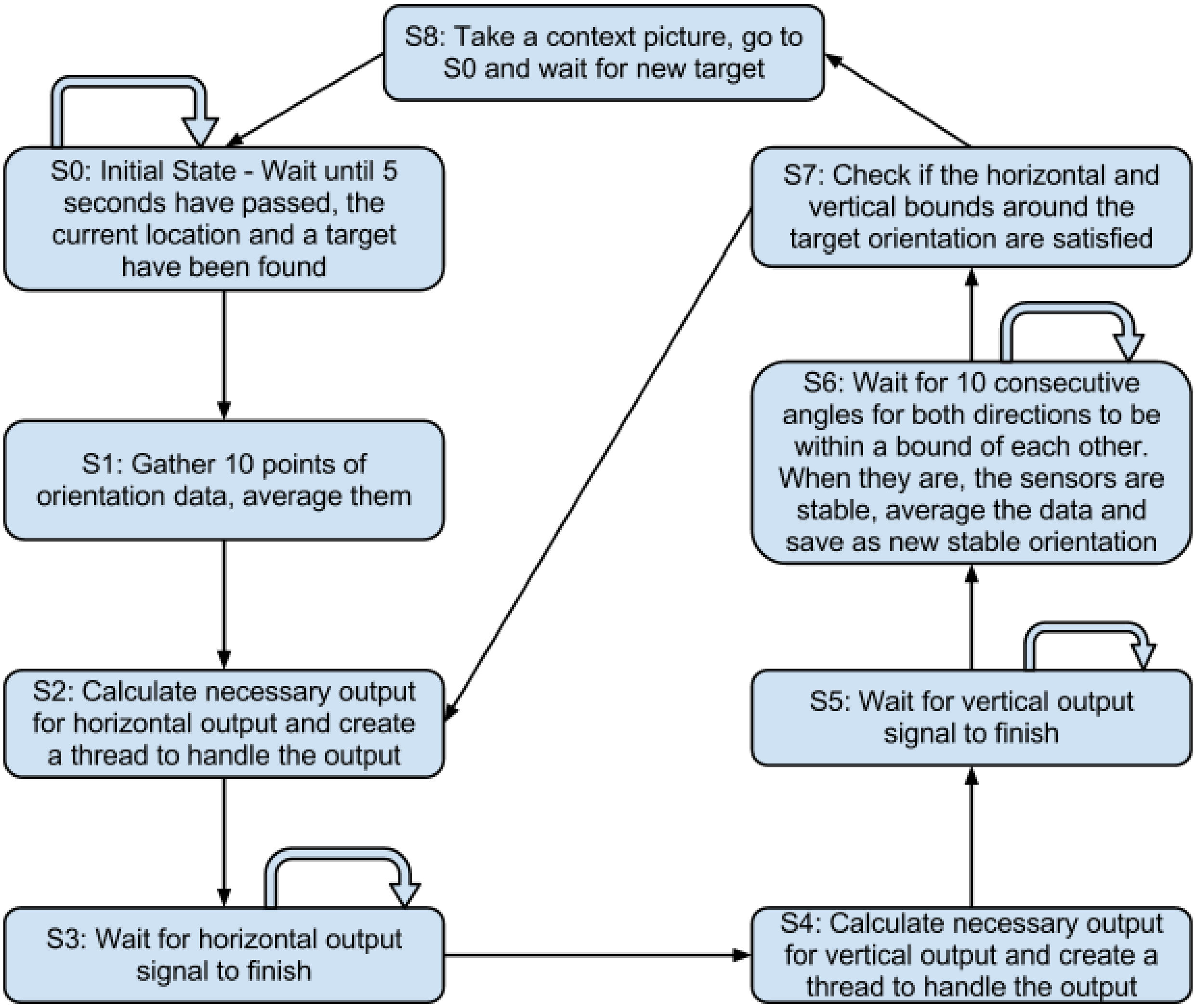
This causes either the motor to lock up and have no net rotation, both motors to receive input steps and rotate simultaneously, or to severely undershoot the target due to a start-up time on each individual output. Due to this, static mode ultimately was chosen. With this mode, it can only rotate 180° azimuthally or 70° vertically in a single output signal. This causes the total time to reach its target to increase when it must rotate through large angles, but severely increases the mount's reliability.

Camera

The UI for this application shows a preview of the camera's view in the center of the screen at all times. This is in part due to Android requiring a camera preview to prevent spy applications, which take pictures without the user being aware. When the final target orientation is achieved, the camera captures an image, and the preview freezes until the next target is loaded from the scheduler. The captured image is stored onto the SD card in the Pictures folder with a timestamp in its name. When capturing the image, many phones (including the one being used) require a shutter sound as another precautionary measure against spy camera applications. This is a problem for the current design, due to the audio jack being used as output to the PIC. The shutter sound is played through both the speakers and audio jack, which would cause the mount to rotate while capturing the image. To prevent this, CyanogenMod 7.1 was installed on the phone, which provides the capability to mute the shutter sound.

State Machine

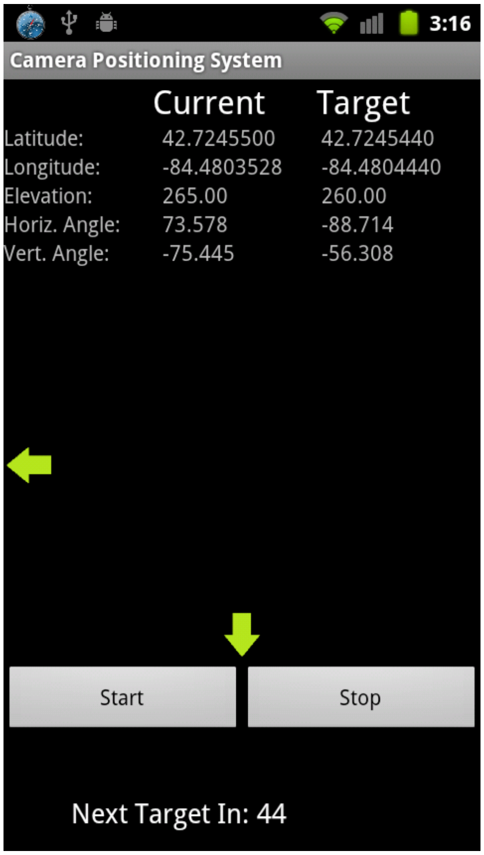
The state machine below represents the main portion of the control software for the phone.


S8: Take a context picture, go to S0 and wait for new target
S0: Initial State - Wait until 5 seconds have passed, the current location and a target have been found
S7: Check if the horizontal and vertical bounds around the target orientation are satisfied
S6: Wait for 10 consecutive angles for both directions to be within a bound of each other. When they are, the sensors are stable, average the data and save as new stable orientation
S1: Gather 10 points of orientation data, average them
S2: Calculate necessary output for horizontal output and create a thread to handle the output
S5: Wait for vertical output signal to finish
S3: Wait for horizontal output signal to finish
S4: Calculate necessary output for vertical output and create a thread to handle the output

The horizontal error angle calculation requires more logic than the vertical calculation. It must first calculate the smallest angle between the current orientation and the desired orientation (either by rotating clockwise or counterclockwise). Then, it checks what the final orientation would be if it rotated by this angle. The mount is restricted to rotating by a total of 190° from the starting orientation in either direction, to prevent the wires through the center pipe from twisting, while still allowing the mount to point directly opposite its starting orientation. If the suggested angle exceeds this limit, it rotates in the opposite direction by the appropriate amount.

Graphical User Interface

A screenshot of the application is provided here:
The screenshot does not capture camera previews, but in the center of the GUI there is a preview of what the camera is viewing at all times (except directly after an image capture, when it has a preview of that image). The GUI also shows the current GPS location, the target GPS location, and the horizontal and vertical angles for the phone's current orientation and the target orientation (with axes defined by OrientationSensor). The green arrows indicate what directions the camera must be rotated in to achieve the target orientation. The Stop button pauses the state machine (and hence rotation), but keeps the schedule running, and the Start button resumes normal operation. Beneath these, the text "Ready for Image" appears when the target orientation is achieved (within bounds) in addition to capturing an image and signaling the laptop. The text at the bottom is a countdown until the next target is read from the schedule. Android smartphones have a menu button beneath the screen, and pressing this pulls up three more options: reloading the schedule from the start, defining the center position (which it cannot move more than 190° from), and returning to the center position.



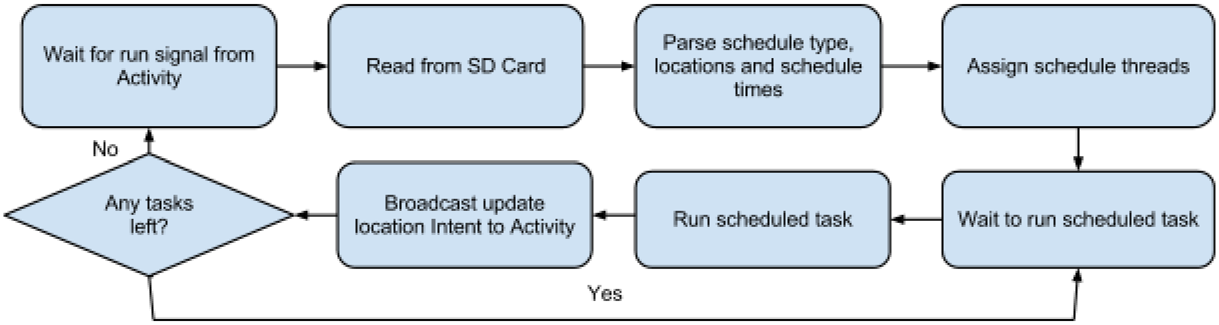
Camera Positioning System

	Current	Target
Latitude:	42.7245500	42.7245440
Longitude:	-84.4803528	-84.4804440
Elevation:	265.00	260.00
Horiz. Angle:	73.578	-88.714
Vert. Angle:	-75.445	-56.308

Next Target In: 44

Scheduling

The basic flow chart for scheduling is below.


Wait for run signal from Activity
Read from SD Card
Parse schedule type, locations and schedule times
Assign schedule threads
Wait to run scheduled task
Run scheduled task
Broadcast update location Intent to Activity
Any tasks left?
No
Yes

Multiple approaches were considered to fulfill the automated schedule requirement, but the design that was eventually settled on was using Android OS's built-in Service and Intent classes, along with the native Java Timer class.

The Service class is similar to an Android Activity, which is the main execution class of an Android application. The main difference between the two is that the Service is designed to run completely in the background, where the Activity class is in the foreground. We decided to use the standard Service class to maintain the schedule in lieu of some of the more advanced

Service subclasses due to the relative simplicity it provides in design and implementation, while still performing the necessary functionality.

One of the downsides of the regular Service class that we discovered, however, is that it is bound inside of an Activity, which limits the function call pathways. Calling functions from the Activity to the Service is trivial, because the Service is an object of the Activity. However, the converse is not so simple, and due to the multithreaded implementation the Timer class utilizes, a class called an Intent must be broadcast from the Service to the Activity.

By using the Intent class, individual threads or subprocesses in an application gain a way to either communicate with each other. In our implementation, these Intents are sent through a scheduled event, by way of Java's built-in Timer class, and then detected by another class, the BroadcastReceiver to update the target coordinates.

The last part of handling the scheduling is the native Java Timer class. Events in the Timer can be scheduled to run either at a designated date and time, or after a delay period in milliseconds. These scheduled events are completely threaded, which introduced a few complications and unexpected results in the application. When using a date and time is to schedule the Timer, a TimerTask object, the threaded set of code, is put into a schedule that can be halted. This was the expected result. However, if using a delay period, the TimerTask thread is started immediately, but pauses for the specified amount of time before executing, which caused overlapping of multiple TimerTask threads and unusual update times for the target locations.

Our desired design was to purge the schedule when the application is no longer in focus, and due to the unexpected behavior described in the previous paragraph, the Timer would continue to run its schedule, despite the Timer.cancel() and Timer.purge() functions being called. We ended up saving the TimerTask thread objects to a vector in order to maintain the references to those threads. This also provided us with a consistent way of purging the schedule, regardless if using dates or delay periods.

Signaling the Laptop

Due to the fact that the Android phone is not directly communicating with the infrared camera, and to the design choice to not use wireless communications, our last option was to use the USB cable to communicate with the laptop. This required networking, and, as stated above, the TCP socket protocol in tandem with the Android Debug Bridge. The ADB handled port forwarding in order to simplify the process.

Similarly, the Timer's threaded processing and the internal communication from the Service to the Activity, communication from the phone to the laptop had complications. This ended up becoming even more complex due to using both the Android Debug Bridge and the TCP socket protocol. Our inexperience with sockets only furthered this difficulty. However, we were able to solve the problems and implement the signaling successfully.

A problem that arose while working on this functionality was recovering the socket connection between the phone and the laptop after a failure. Due to a quirk with older Android phones, the phone has to act as the socket server, and the laptop as the client. However, if either of the devices lost connection, neither would handle reconnecting properly, especially on the client/laptop side. Originally, we thought it was due to some strange issue with the socket connection. However, it was discovered that the ADB was continuing to maintain the connection, and was preventing either side from being able to reconnect. In order to make sure the client side would properly reconnect, the ADB had to be killed and then restarted from within the desktop application. On the phone, a careful process of closing, destroying, recreating, and then reconnecting the server and its sockets needed to be done when necessary.

Chapter 4

The testing of our project went through much iteration. We tested each piece individually as well as together as a whole unit. The testing process centered around four areas: the motor controllers, the audio jack to motor control circuitry, the Android software, and the camera mounting system.

The motor controllers had the least success all through the project. We had repeated failure with parts that claimed to be tolerant to high operating currents, but failed to withstand the currents that the stepper motors draw. In the first two iterations, we used ready-made motor controllers from online vendors that achieved, but did not successfully sustain their advertised current capacity. With later iterations, we refined our own design using large high power parts in a custom-built circuit to accomplish our goal. These controllers will reliably cause the stepper motors to turn properly with all components being able to sustain power requirements 50% in excess of our needs. This extra step was taken both to increase quality and reliability and did not greatly affect our budget. In the case of our power MOSFETs, upgrades had a 5 cent difference out of a total price of $1.50 per unit and provided an increase peak current from 40 A at 30V to 56A at 60V. Upgraded terminal blocks had a 35 cent difference out of a total price of $6.50 per unit and provided an increase peak current limit from 6A to 16A.

Prior to our circuit's integration with our Droid ® phone, a function generator was used in order to test our logic circuitry by providing square waves as step inputs to the circuit. This produced the expected behavior of one step equal to 1.8 degree motor rotation. When the direction bit was flipped from low to high, the rotation reversed direction. These tests were followed up with connecting this circuitry to the output coming from the PIC microcontroller. After some modification of the PIC code, the PIC was properly able to change the rotation and direction of the motor based on the DC voltage being presented to its analog-to-digital converter. With the PIC microcontroller outputting steps at full Vdd, this indicated the polar motor to rotate counter-clockwise, increasing our camera's elevation view. At 75% Vdd, the polar motor rotates clockwise, forcing the camera to point down. At 50% Vdd, the azimuthal motor rotates clockwise and at 25% Vdd, it rotates counter-clockwise. This proved to be the simplest and most effective way to distinguish motors and their rotation during operation.

The audio jack to motor controller circuitry also suffered some design challenges early on. As mentioned in chapter 3, this went through many design revisions. Early versions used a full-wave rectifier to provide the necessary signals to the PIC microcontroller, or even to discrete components. However, this method was unreliable due to the capacitance introduced in the ripple filter. The group instead opted to output a DC voltage from the phone's headphone jack instead of using AC frequency, once it was found that the phone could output DC signals. This allows the use an RC filter to get rid of noise to the PIC. After making these adjustments tests were made to ensure that extra delay would not be an issue for the PIC to control the motor controllers. This is both successful and reliable.

The camera mounting system's base, including the azimuthal gearing, motor and bearings secured to the base platform has experienced differing degrees of failure. During the tooling of the main-drive shaft, the azimuthal gear broke. This was temporarily repaired using a 3D printed gear since our gear supplier; SDP-SI did not have our gear in stock. Our printed gear is not as hard as the original material and the decreased contact with the motor's gear in certain parts accelerates the printed gear's wear. This printed gear will need to be replaced, preferably with mineral doped nylon, as we originally planned, or metal gears. The original flange bearing also failed as it did not fully support the weight of our mount in the way we wanted. It would cause a significant wobble of more than 10 degrees. This was replaced by a lazy-susan turntable bearing which is both cheaper (at $2.00 it was 1/7 the price of the original flange bearing), and provides for a much smoother operation with about 4 degrees of deviation from center. This change did require the creation of a different base since we no longer could rely on the main drive shaft and its matching tripod for support. This change however became beneficial as our sponsor, Dr. LeMaster later required us to have a mounting platform that could be used independently of a tri-pod.

Testing of the phone's GPS capability has also been slightly tricky. Due to the need for a standard wall outlet for our power source, we are currently unable to take our design far enough away from the exterior walls of the engineering building to get a clear GPS signal in many orientations. Additionally, this cannot be tested inside. This means that to test the software, we have had to hold the camera in our hands to achieve its desired orientation, which means that as a unit our design has not been capable of being fully tested with actual GPS data. For our current testing, we are providing a false set of this GPS data. Additionally, the phone's magnetometer has been difficult to test correctly, as the interior of the Engineering building; itself is enough to give some unexpected readings, causing undesired rotations of the mount. These problems should be minimized if the mount is being used outside, with a power source and is far away from anything that can affect its magnetometer data.

Our team's goal was to have an accuracy of 2° in both the polar and azimuthal rotations, matching what our sponsor requested and what his camera's field of view can account for. Despite needing to iterate our design, the mount is capable of achieving this goal, assuming the orientation sensor on the phone delivers accurate data.

Chapter 5

The team found that creating a camera positioning system using an Android phone as a sensor array is possible, but has a few problems. We found that we could rotate the mount on a schedule and point the camera towards a given location with a reasonable amount of accuracy. The tests performed on the mount showed that it normally achieves an accuracy of within 5 degrees. This, however, is highly dependent on a number of variables. During our tests, there were many locations where we had trouble getting a consistent GPS signal on the phone. In this scenario, the phone does its best to estimate its location and this estimation can be inaccurate resulting in unpredicted results. We also found that the accelerometers and magnetometers on the phone behaved inconsistently and would frequently report inaccurate orientation readings resulting in the mount not turning enough or turning too much. The final issue with the accuracy of the project is the gears used for azimuthal rotation. The gear on the base of the mount is prone to skipping teeth. Occasionally while trying to rotate the two base gears do not mesh properly and a step is skipped, this can throw off the phone calculations and make the movement slow or jerky. In some extreme cases, the mount refuses to rotate to a position due to the gear not being able to rotate the required amount, and will skip that entry on the schedule. This problem could be solved with gears made out of better material and having the two gears pressed together more firmly. Chapter 4 details some more design issues encountered with the project.

The final schedule is shown in the second Gantt chart in Appendix 4. As stated in Chapter 2, the largest difference between the initial Gantt chart and this one is the time taken to test multiple motor controller boards and build a custom board.

The final costs of this project are as follows:
Total budget - $500

Item	Cost
Stepper Motors	$48
Power Supply	$25
DC-DC converter	$35
Mount Materials (Gears, Bearing, etc.)	$43
Components for Logic	$20
Components for Motor Controllers	$50
Four burned-out Motor Controller Boards	$92
Android Smartphone - Motorola Droid®	$50

Miscellaneous Parts	$12
Total Shipping	$60
Total	$435
Actual Cost of Parts for Mount	$283

In total, this project came in under-budget by $65. Additionally, the actual material cost of the mount was only $283, far less expensive than other possible solutions for the AFRL.

In consideration of the outcome of our project, design team three would suggest the following changes be made to the project in any future revision:

- Use all metal gears
- Use servo motors instead of stepper motors
- Use a more appropriate power supply
- Build the mounting platform out of a solid block of aluminum
- Use a bearing that does not deform or lean when weight is applied
- Use a newer phone with more accurate sensors and processing power

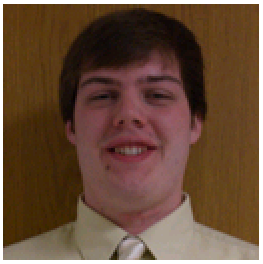
In conclusion, the design team met these design criteria:

1. Positioning - Slewing between target points defined by GPS coordinates at various times according to a pre-programmed test script
    - Azimuthal rotation range of at least 0 to 360 degrees around the camera's position..
    - Polar elevation range of 0 to 90 degrees above and below the horizon
    - Capable of securely rotating and managing a 20lb load
2. Coordinate sensing and calculation - Automatically sensing the camera's own position and orientation in order to calculate the required target acquisition geometry.
    - GPS, compass heading, and elevation is to be detected using an Android based smartphone.
3. Imaging - Generating an image acquisition command signal to the attached infrared camera.
    - Using the phone's camera to acquire context imagery in the visual spectrum.
4. Optimizing cost vs. performance.
5. Provide a base for the mount that can be mated to the sponsor's tripod mount, or used as a standalone platform.
    - Producing a mount sturdy enough not only for the camera and it's weight, but the environment as well (ie. rain and wind).
    - Produce a positioning mechanism sturdy enough to keep the camera at a designated position, allowing for video acquisition.

Limitations of the mount:

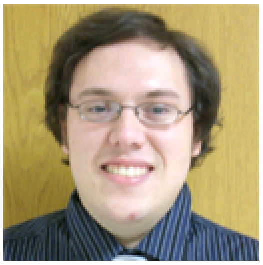
1. Accuracy of the phone’s sensors – The phone’s magnetometer is suitable for navigation applications, within 10°, but not for the degree of accuracy (< 2°) required for this project.
2. Due to the materials of the gears, the mount could not reach a speed of rotation that exceeded our minimum design specifications, and may become a problem as wear and tear smooth out the gear teeth.

Appendix 1 – Technical roles, responsibilities, and work accomplished



Austin Fletcher

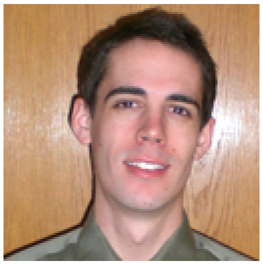
The first task I worked on was a collaboration with Ryan. It involved taking the signal from the headphone jack and making it into a useable signal. The first iteration of this idea used band pass filters to turn on the enable pins of our motor controllers. This failed due to the precision of the band pass filters needed, and the destruction of the first motor controller. The second iteration utilized a full-wave rectifier and a PIC microcontroller to measure the voltage provided by the audio jack, which in turn enabled an AND gate and allowed a step signal, generated with a 555 timer, to pass. This failed due to the imprecise timing that the full-wave rectifier introduced, as it took too long to charge and discharge. The final iteration of this was to use a straight DC signal from one channel of the audio jack to control the PIC, which then generated its own stepping signal. After we destroyed our final motor controller, my primary objective was to find a replacement circuit that could handle the massive current that we now knew we were dealing with. I discovered a schematic and a parts list, which lead us to our solution that is functioning in the final prototype. After that, I was left to find a power source capable of powering everything that we needed. I found a cheap power supply on Ebay that fit our specifications, at the time, for minimum current and voltage. Additionally, I found the appropriate DC-DC converter to use on our final prototype, this was necessary to limit the current draw caused by our motor windings. In-between all of these things I helped with the physical construction of the mount. On several occasions, my contributions to the physical mount were to solve problems that were caused by damaged or defective parts. All through the semester, I kept tabs on all of the other group members accomplishments and direction in order to make sure that all of the group contributions would mesh together correctly at the conclusion of the project.



Jeremy Iamurri

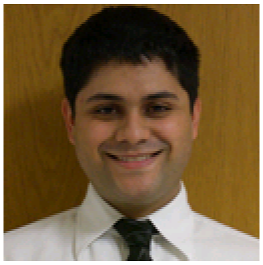
I was in charge of the scheduling and communication aspects of the Android application. This involved using multi-threaded coding with Java's built in Thread and Timer classes, implementing background processing in the activity using the Android Service class, and implementing communication aspects using the TCP socket protocol and the Android Intent system. The schedule, in its final iteration, is completely automatic and, depending on the schedule type input in the schedule.txt file on the SD card, will loop between the selected points endlessly. In order to get the cross device communication working, I had to create a console Java application that uses the Android Debug Bridge to ease the process of communication between the laptop and the phone.

As discussed above, this involved some odd complications due to the somewhat unpredictable nature of network communications, and because the entire process was threaded to prevent stalling on the phone. Eventually I implemented a way to let the laptop and phone auto-recover if the socket connection is ever lost by killing and restarting the ADB sub-process in the laptop and restarting the server socket on the phone, which was possibly one of the more confusing aspects. We were not aware of the ADB continuing to hold the socket until I did some random tests. I also solved the issue with the camera not being unbound when the application is paused. This was done by explicitly destroying the camera object and its layout panel on pause and then recreating everything when the application resumes.



Ryan Popa

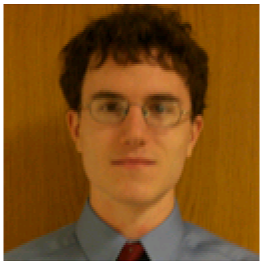
Most of my work has been on the circuitry for controlling the motors from the phone. I started out by designing the analog circuitry we were using in the first design iteration of the circuit with help from Austin. He and I built prototype circuits for that design on breadboards using outputs from a frequency generator to test. I did most of the designing of the rectifier as well as the physical construction of the circuit. When we abandoned that solution and turned to using a microcontroller I wrote the original code for measuring the DC voltage from the phone and outputting signals for the motors. At the time I was using LEDs to be placeholders for what would eventually be wires to a motor controller. After that Austin and I figured out how stepper-motors worked and tested the motor controllers we ordered. I also did research into what power supply to purchase after reviewing the stepper-motor datasheets. When we designed our own motor controllers I assembled all of the motor control circuitry and microcontroller circuitry on the breadboard and did the necessary soldering and wiring for full operation. This required laying out everything in a way that would fit on our mount and routing the wires in a way that would be both attractive and functional. After that I, with the assistance of Austin, replaced a broken bearing in the camera mount and redesigned the stationary base of the mount for use with the new bearing. When it came time to test everything I made changes to the PIC code as requested by the Android software team to change the speed of rotation of the motors. During final assembly I fabricated a base plate for the circuitry and an acrylic face plate to protect the wires. I helped with the final wiring of the circuitry to the phone and the motors. Finally I helped with the final testing and all of the documents and presentations required throughout the semester.



Yan Sidronio

I focused primarily on the mount's mechanical design and integrating its movement with the motors and its controllers. This involved all of the necessary mechanical calculations in prototyping the final design (ie, center of mass, torque, inertia, material stresses), 3D CAD modeling and CAM simulation of the mount, deciding which mechanical (bearings, gears, motors) components to buy and constructing the mount. All CAD modeling was done using Autodesk Inventor ® and is located in Appendix 3. This was a time saving step as it allowed us to clearly communicate with our sponsor, Dr. LeMaster, what our design looks like, how it will function and what his thoughts about where. It also allowed me to make 1:1 scale blue-prints that facilitated machining of parts. Manufacturing of our mount also included some CNC machine programming in order to properly fit our polar motor to its vertical support and 3D printing to replace the azimuthal gear which was accidently broken during the machining of the azimuthal drive shaft.

Since I constructed the mount, I became in charge of acquiring parts. Due to this I worked closely with Austin Fletcher and Ryan Popa in deciding how to control our stepper motors, which components to buy in order to safely meet our power requirements and assisting with its design and wiring. The first motor controller iteration used pre-made controllers from Spark Fun ® that failed to meet our stepper motor power requirements. The second iteration used a circuit that I made based on Hobby Engineering’s Unipolar stepper motor driver board which used 7400 series logic ICs and TIP32 power MOSFETS. With this circuit we were able to finally test our Android software with our physical motors and mount. It was a reliable circuit and was very useful for testing, however it did not produce the necessary amount of torque for our camera. Our team’s third and final iteration uses 4000 series ICs and IRFZ44 power MOSFETS. I assisted with the final circuit’s wiring and placing it into our mount.



Chris Sigler

My portion of the overall project was the part of the Android application which controls the mount, including collection GPS and orientation sensor data, calculating the target orientation, outputting appropriate control signals from the smartphone, and creating a graphical user interface. Additionally, I worked on a variety of other aspects of this project, including implementing the camera functionality of the app, finding bugs in Jeremy's code (and fixing some), assisting in designing and building a prototype motor controller circuit, assisting in programming the PIC microcontroller to receive signals from the phone appropriately, and assisting in the construction of the final mount circuitry.

My portion of the Android application could be written separately from the rest of the team's work, once we decided on how to communicate with the phone. I had the problem of not having an Android phone to test with until we chose a phone and received it, and even then, although I could write the app, testing could not be done for many features until the mount or control circuitry was built. Prior to receiving the final phone, I borrowed and tested a variety of phones, and found that only the newest and most expensive phones could output a DC signal through the audio jack. The output from others decayed exponentially. After receiving the final phone and testing it, I found that one channel on this phone could hold a DC value without decaying (the other channel could not).

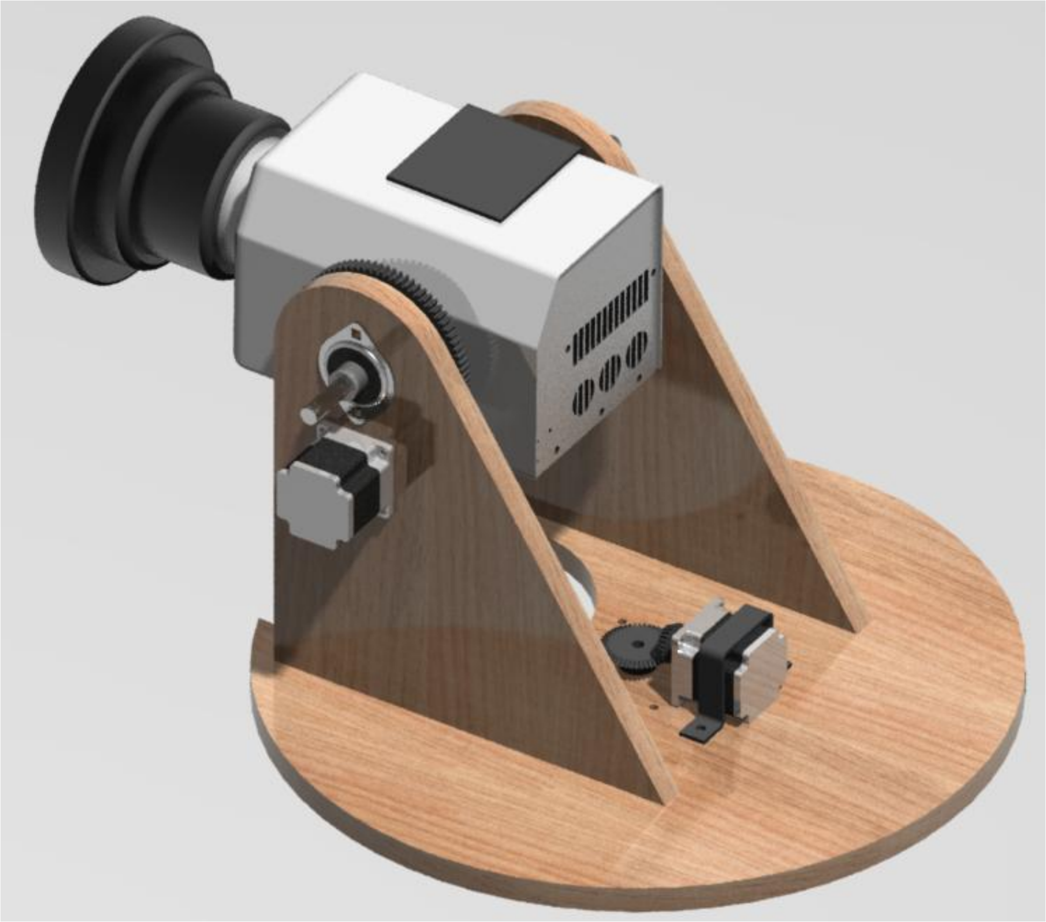
I completed most of the app well before the mount was completed enough to test it properly, or before the design changed to another iteration. So, during these periods I spent my time working on other parts of the project. I wrote the camera portion of the application, which Jeremy assisted with in debugging. I also helped Ryan and Austin diagnose issues with the circuitry whenever they cropped up, and aided them in building the final circuitry on the mount whenever they needed help. Despite the fact that my responsibility on the project was coding, I spent large amounts of time in the lab helping my teammates throughout the semester on other aspects.

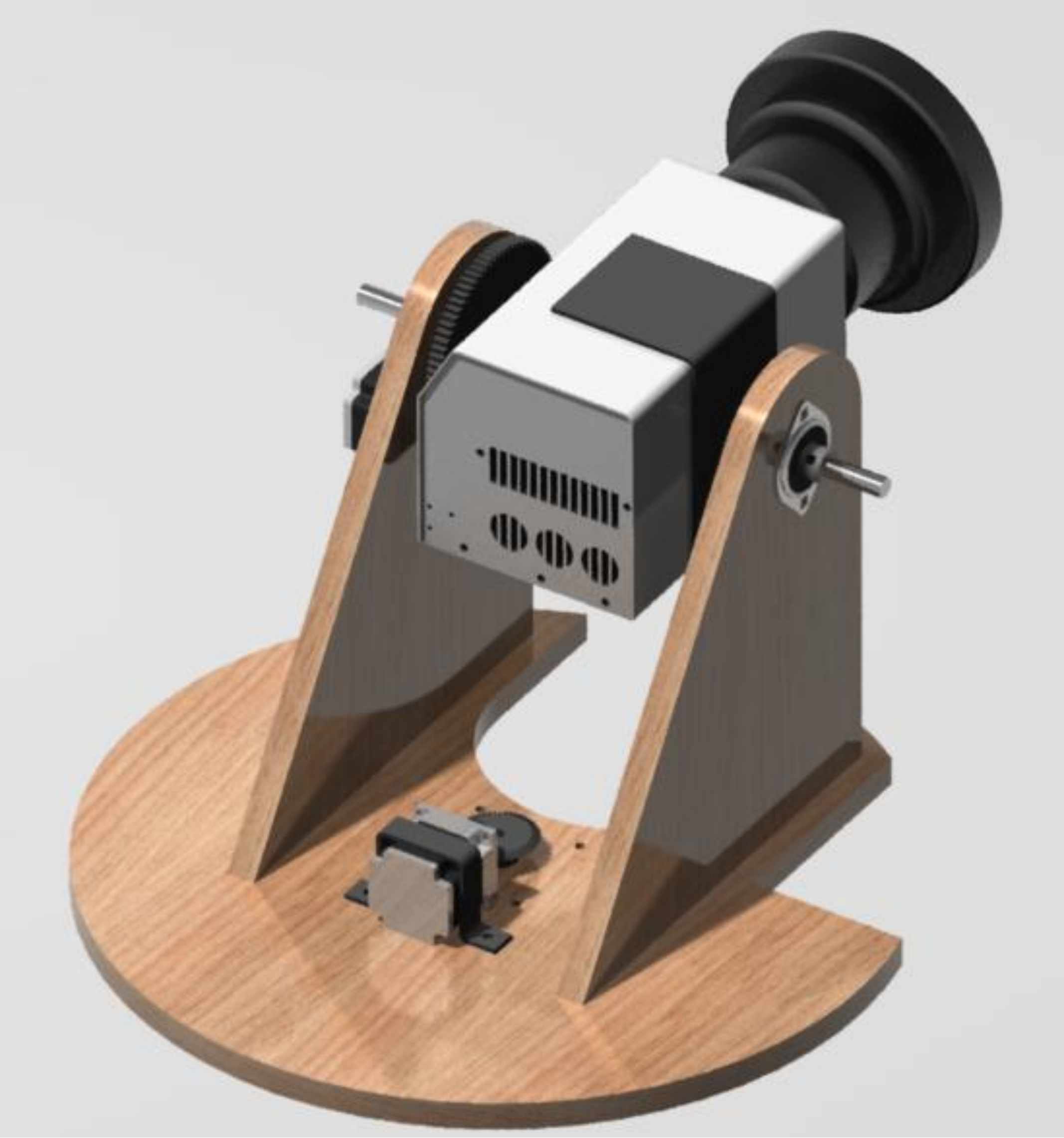
Appendix 2 – Literature and website references

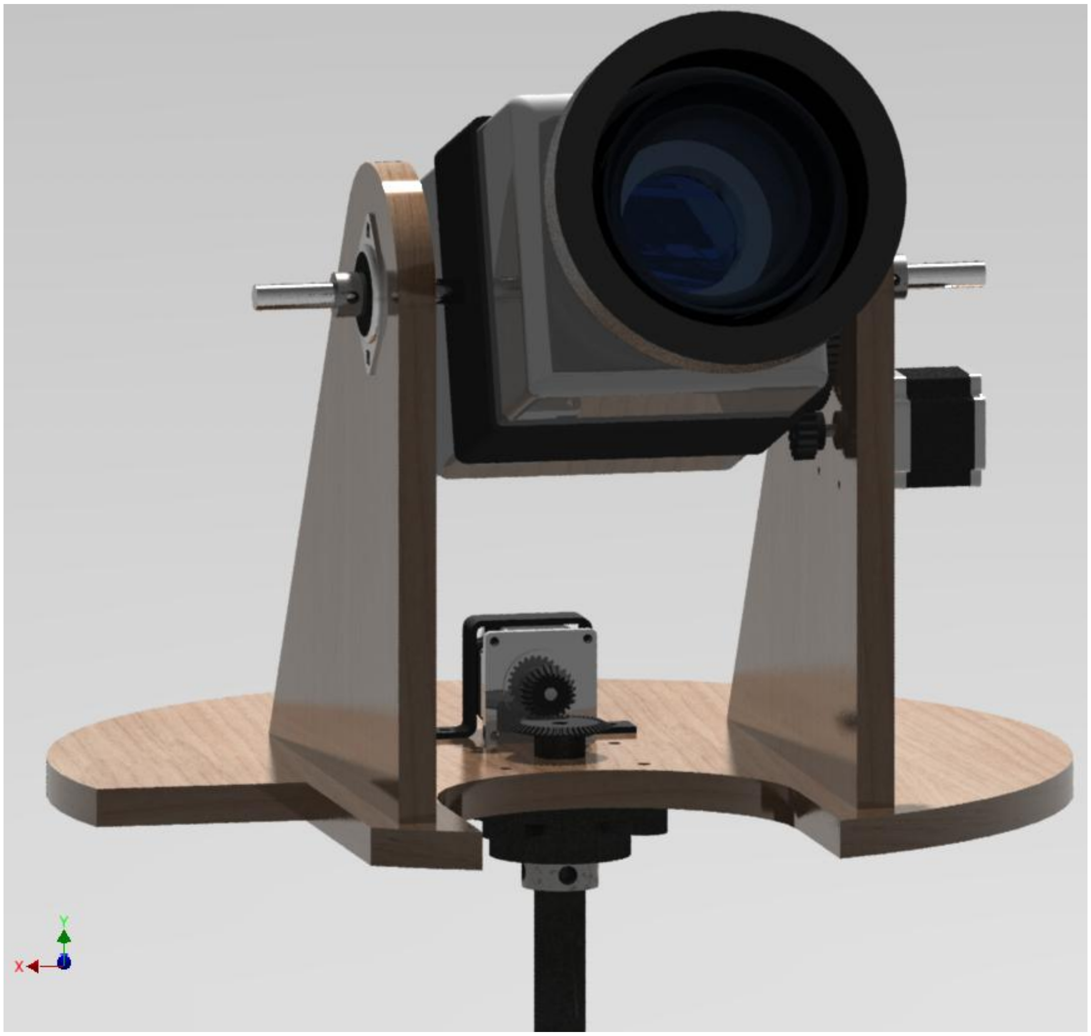
This appendix should list references of any books, data sheets, web sites, manuals, etc. used to research, design, and implement your project.

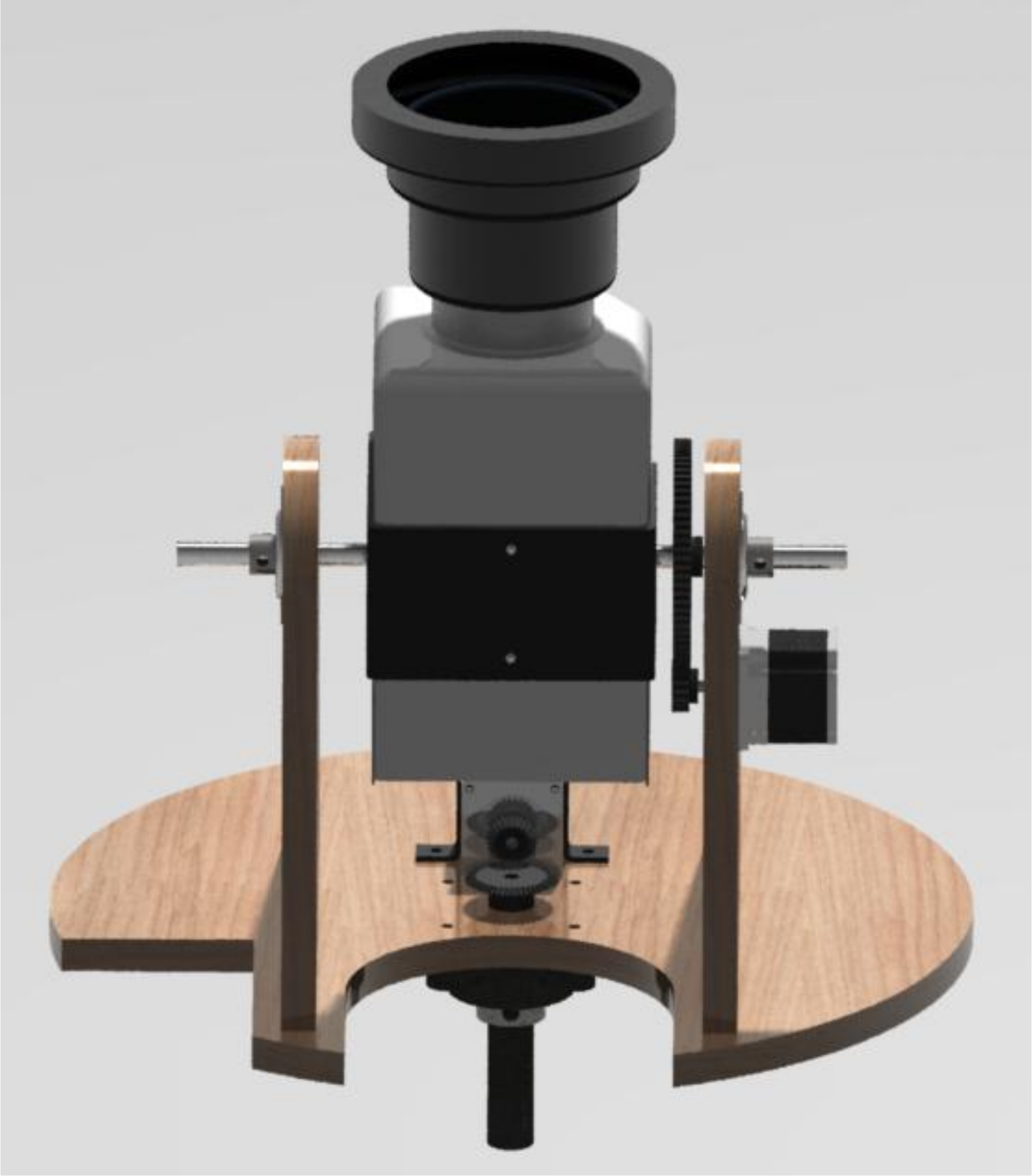
http://www.eecs.umich.edu/~prabal/pubs/papers/kuo10hijack.pdf
http://www.creativedistraction.com/demos/sensor-data-to-iphone-through-the-headphone-jack-using-arduino/
http://developer.android.com/sdk/index.html
http://www.flir.com/mcs/products/advance-features/Geo-Pointing-Module/index.cfm
http://www.anothem.net/archives/2010/10/15/android-usb-connection-to-pc/
https://github.com/commonsguy/cw-android/tree/master/Service
http://hobbyengineering.com/H1259.html

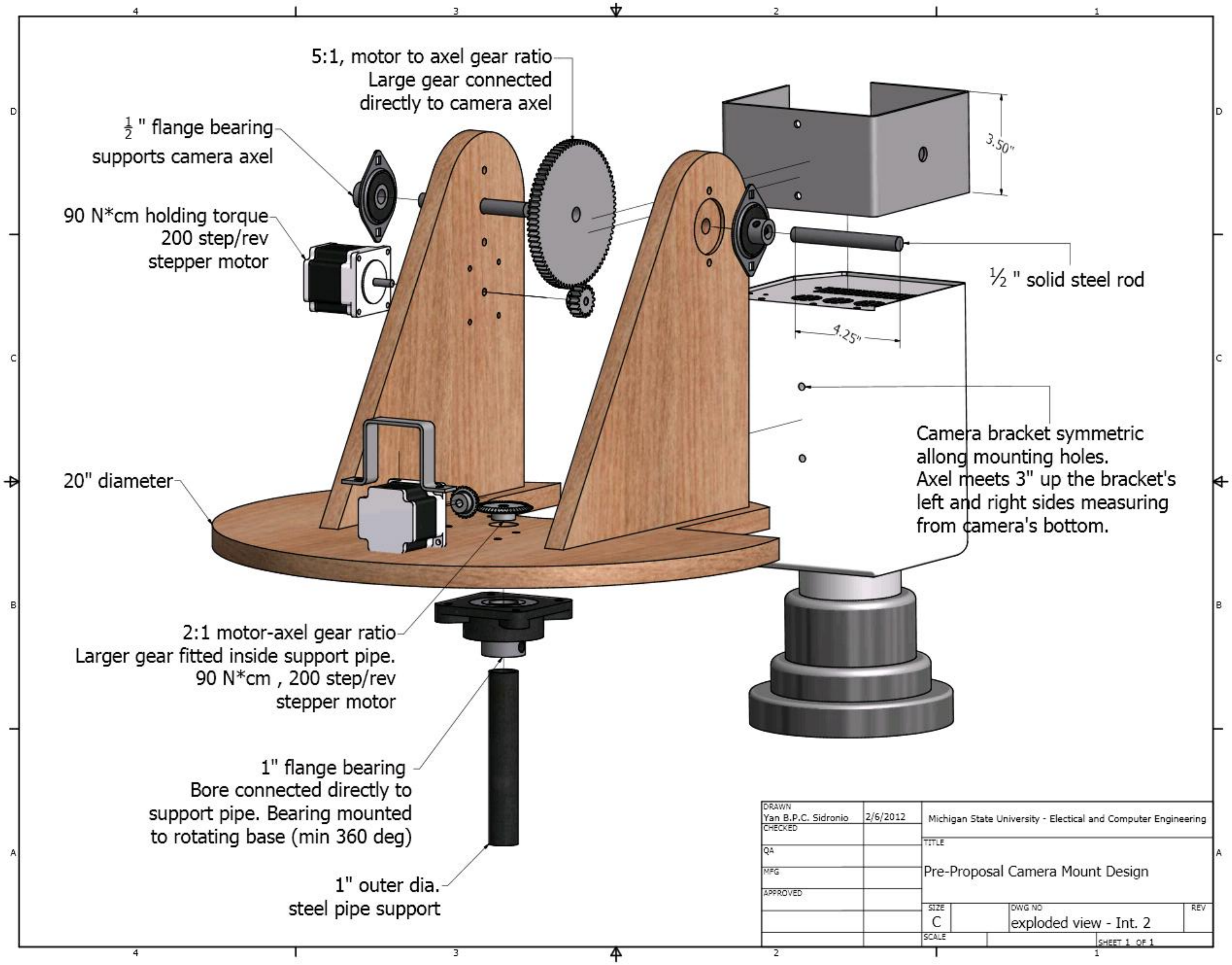
Appendix 3 - CAD Images of Camera Mount



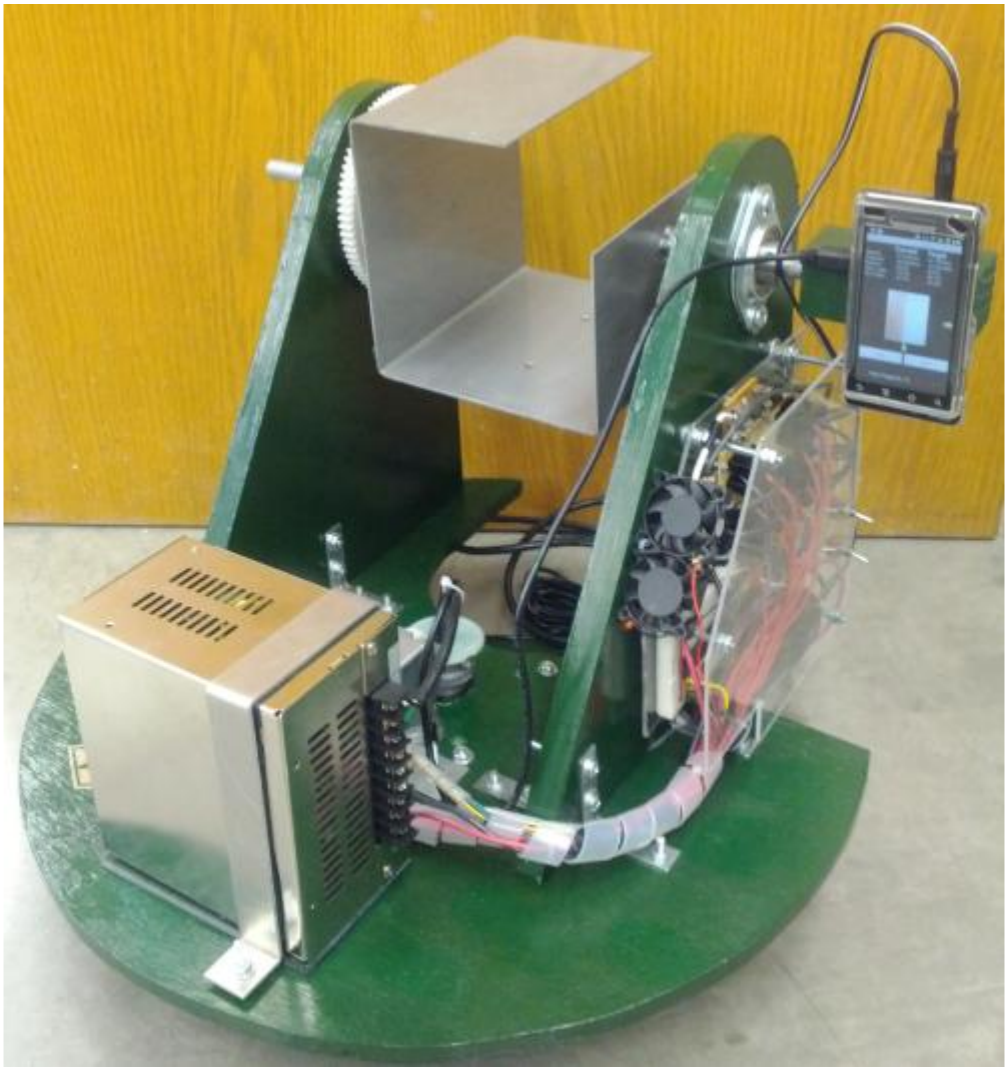


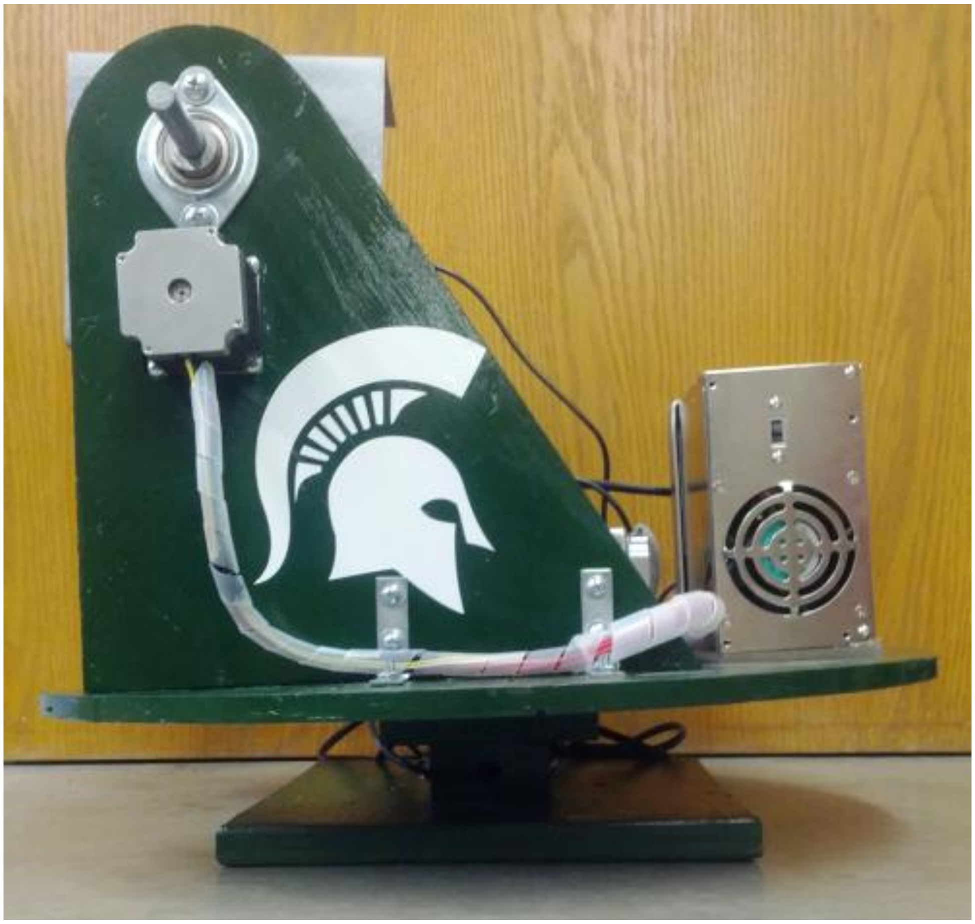

Y
X




5:1, motor to axel gear ratio
Large gear connected
directly to camera axel
½ " flange bearing
supports camera axel
90 N*cm holding torque
200 step/rev
stepper motor
3.50"
½ " solid steel rod
4.25"
Camera bracket symmetric
allong mounting holes.
Axel meets 3" up the bracket's
left and right sides measuring
from camera's bottom.
20" diameter
2:1 motor-axel gear ratio
Larger gear fitted inside support pipe.
90 N*cm , 200 step/rev
stepper motor
1" flange bearing
Bore connected directly to
support pipe. Bearing mounted
to rotating base (min 360 deg)
1" outer dia.
steel pipe support
DRAWN
Yan B.P.C. Sidronio
2/6/2012
CHECKED
QA
MFG
APPROVED
Michigan State University - Electical and Computer Engineering
TITLE
Pre-Proposal Camera Mount Design
SIZE
C
DWG NO
exploded view - Int. 2
REV
SCALE
SHEET 1 OF 1
4
3
2
1
D
C
B
A

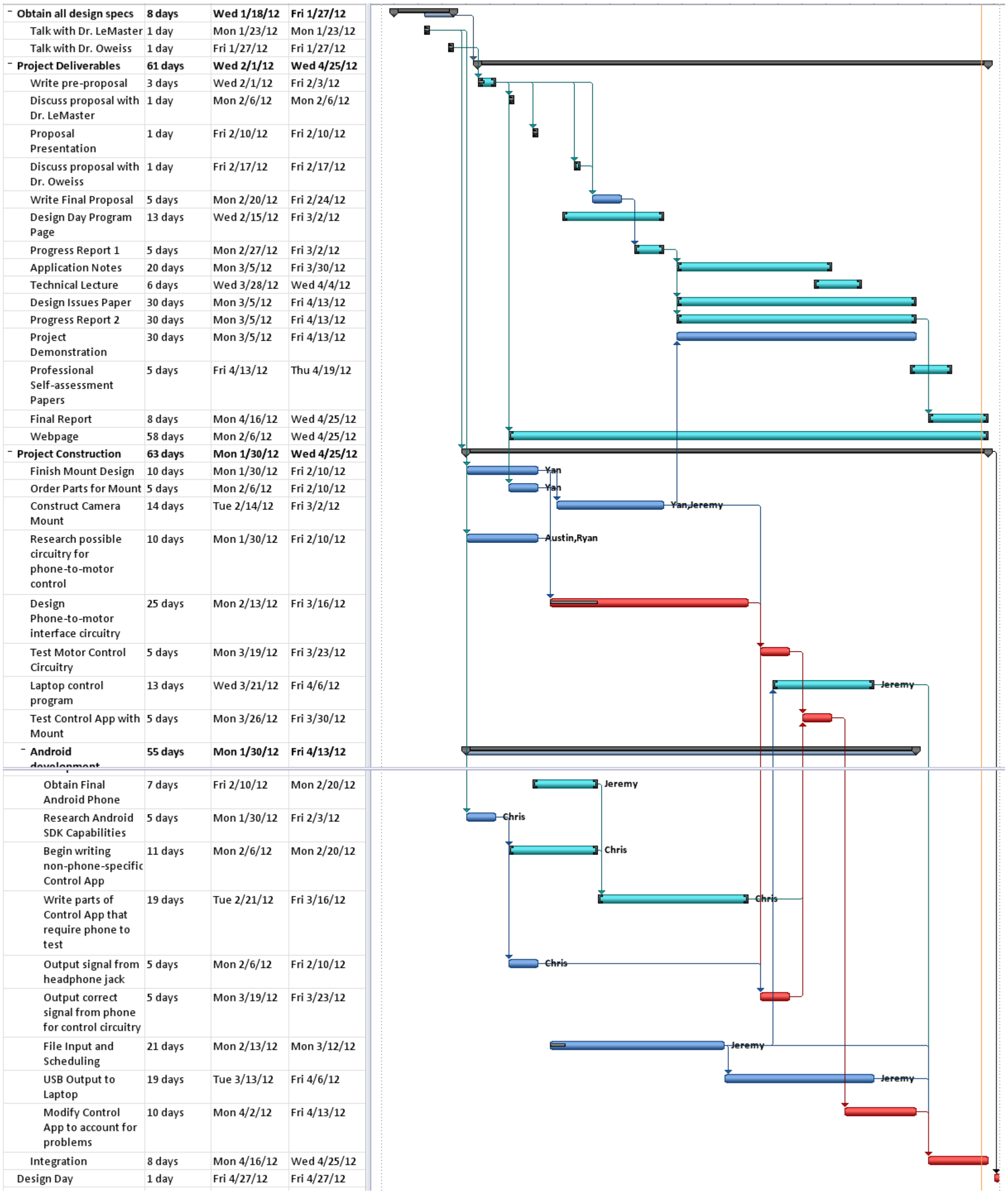
Appendix 4 - Images of Final Mount





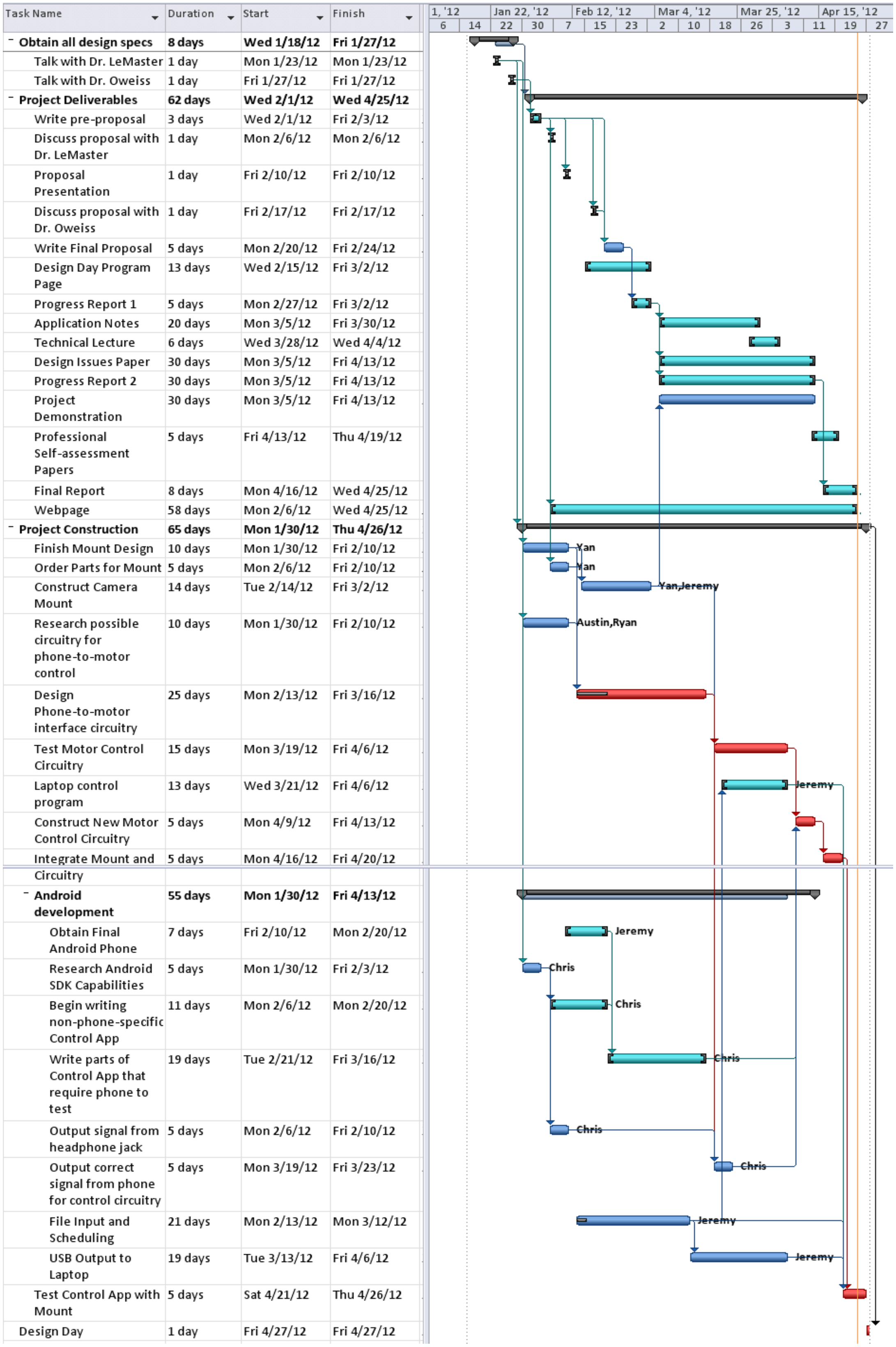
Appendix 5 - Gantt Charts

The first Gantt chart shows the proposed project plan throughout the course of the semester as it was created in Week 4 of the semester.



Obtain all design specs	**8 days**	**Wed 1/18/12**	**Fri 1/27/12**
Talk with Dr. LeMaster	1 day	Mon 1/23/12	Mon 1/23/12
Talk with Dr. Oweiss	1 day	Fri 1/27/12	Fri 1/27/12
Project Deliverables	**61 days**	**Wed 2/1/12**	**Wed 4/25/12**
Write pre-proposal	3 days	Wed 2/1/12	Fri 2/3/12
Discuss proposal with Dr. LeMaster	1 day	Mon 2/6/12	Mon 2/6/12
Proposal Presentation	1 day	Fri 2/10/12	Fri 2/10/12
Discuss proposal with Dr. Oweiss	1 day	Fri 2/17/12	Fri 2/17/12
Write Final Proposal	5 days	Mon 2/20/12	Fri 2/24/12
Design Day Program Page	13 days	Wed 2/15/12	Fri 3/2/12
Progress Report 1	5 days	Mon 2/27/12	Fri 3/2/12
Application Notes	20 days	Mon 3/5/12	Fri 3/30/12
Technical Lecture	6 days	Wed 3/28/12	Wed 4/4/12
Design Issues Paper	30 days	Mon 3/5/12	Fri 4/13/12
Progress Report 2	30 days	Mon 3/5/12	Fri 4/13/12
Project Demonstration	30 days	Mon 3/5/12	Fri 4/13/12
Professional Self-assessment Papers	5 days	Fri 4/13/12	Thu 4/19/12
Final Report	8 days	Mon 4/16/12	Wed 4/25/12
Webpage	58 days	Mon 2/6/12	Wed 4/25/12
Project Construction	**63 days**	**Mon 1/30/12**	**Wed 4/25/12**
Finish Mount Design	10 days	Mon 1/30/12	Fri 2/10/12
Order Parts for Mount	5 days	Mon 2/6/12	Fri 2/10/12
Construct Camera Mount	14 days	Tue 2/14/12	Fri 3/2/12
Research possible circuitry for phone-to-motor control	10 days	Mon 1/30/12	Fri 2/10/12
Design Phone-to-motor interface circuitry	25 days	Mon 2/13/12	Fri 3/16/12
Test Motor Control Circuitry	5 days	Mon 3/19/12	Fri 3/23/12
Laptop control program	13 days	Wed 3/21/12	Fri 4/6/12
Test Control App with Mount	5 days	Mon 3/26/12	Fri 3/30/12
Android development	**55 days**	**Mon 1/30/12**	**Fri 4/13/12**
Obtain Final Android Phone	7 days	Fri 2/10/12	Mon 2/20/12
Research Android SDK Capabilities	5 days	Mon 1/30/12	Fri 2/3/12
Begin writing non-phone-specific Control App	11 days	Mon 2/6/12	Mon 2/20/12
Write parts of Control App that require phone to test	19 days	Tue 2/21/12	Fri 3/16/12
Output signal from headphone jack	5 days	Mon 2/6/12	Fri 2/10/12
Output correct signal from phone for control circuitry	5 days	Mon 3/19/12	Fri 3/23/12
File Input and Scheduling	21 days	Mon 2/13/12	Mon 3/12/12
USB Output to Laptop	19 days	Tue 3/13/12	Fri 4/6/12
Modify Control App to account for problems	10 days	Mon 4/2/12	Fri 4/13/12
Integration	8 days	Mon 4/16/12	Wed 4/25/12
Design Day	1 day	Fri 4/27/12	Fri 4/27/12

The second Gantt charts shows the final Gantt chart after delays.



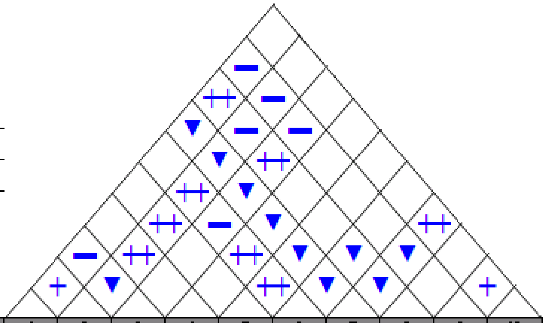
Task Name	Duration	Start	Finish
Obtain all design specs	**8 days**	**Wed 1/18/12**	**Fri 1/27/12**
Talk with Dr. LeMaster	1 day	Mon 1/23/12	Mon 1/23/12
Talk with Dr. Oweiss	1 day	Fri 1/27/12	Fri 1/27/12
Project Deliverables	**62 days**	**Wed 2/1/12**	**Wed 4/25/12**
Write pre-proposal	3 days	Wed 2/1/12	Fri 2/3/12
Discuss proposal with Dr. LeMaster	1 day	Mon 2/6/12	Mon 2/6/12
Proposal Presentation	1 day	Fri 2/10/12	Fri 2/10/12
Discuss proposal with Dr. Oweiss	1 day	Fri 2/17/12	Fri 2/17/12
Write Final Proposal	5 days	Mon 2/20/12	Fri 2/24/12
Design Day Program Page	13 days	Wed 2/15/12	Fri 3/2/12
Progress Report 1	5 days	Mon 2/27/12	Fri 3/2/12
Application Notes	20 days	Mon 3/5/12	Fri 3/30/12
Technical Lecture	6 days	Wed 3/28/12	Wed 4/4/12
Design Issues Paper	30 days	Mon 3/5/12	Fri 4/13/12
Progress Report 2	30 days	Mon 3/5/12	Fri 4/13/12
Project Demonstration	30 days	Mon 3/5/12	Fri 4/13/12
Professional Self-assessment Papers	5 days	Fri 4/13/12	Thu 4/19/12
Final Report	8 days	Mon 4/16/12	Wed 4/25/12
Webpage	58 days	Mon 2/6/12	Wed 4/25/12
Project Construction	**65 days**	**Mon 1/30/12**	**Thu 4/26/12**
Finish Mount Design	10 days	Mon 1/30/12	Fri 2/10/12
Order Parts for Mount	5 days	Mon 2/6/12	Fri 2/10/12
Construct Camera Mount	14 days	Tue 2/14/12	Fri 3/2/12
Research possible circuitry for phone-to-motor control	10 days	Mon 1/30/12	Fri 2/10/12
Design Phone-to-motor interface circuitry	25 days	Mon 2/13/12	Fri 3/16/12
Test Motor Control Circuitry	15 days	Mon 3/19/12	Fri 4/6/12
Laptop control program	13 days	Wed 3/21/12	Fri 4/6/12
Construct New Motor Control Circuitry	5 days	Mon 4/9/12	Fri 4/13/12
Integrate Mount and Circuitry	5 days	Mon 4/16/12	Fri 4/20/12
Android development	**55 days**	**Mon 1/30/12**	**Fri 4/13/12**
Obtain Final Android Phone	7 days	Fri 2/10/12	Mon 2/20/12
Research Android SDK Capabilities	5 days	Mon 1/30/12	Fri 2/3/12
Begin writing non-phone-specific Control App	11 days	Mon 2/6/12	Mon 2/20/12
Write parts of Control App that require phone to test	19 days	Tue 2/21/12	Fri 3/16/12
Output signal from headphone jack	5 days	Mon 2/6/12	Fri 2/10/12
Output correct signal from phone for control circuitry	5 days	Mon 3/19/12	Fri 3/23/12
File Input and Scheduling	21 days	Mon 2/13/12	Mon 3/12/12
USB Output to Laptop	19 days	Tue 3/13/12	Fri 4/6/12
Test Control App with Mount	5 days	Sat 4/21/12	Thu 4/26/12
Design Day	1 day	Fri 4/27/12	Fri 4/27/12

Appendix 6 - House of Quality

Title: Android Enabled Programmable Camera Positing System

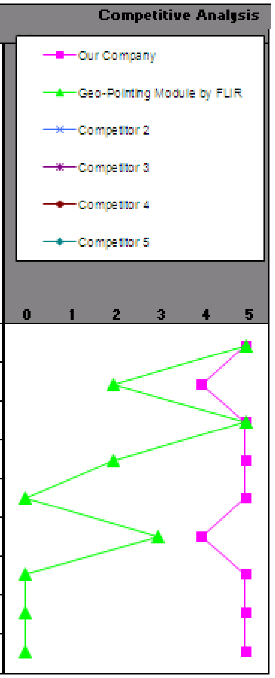
Author: ECE 480 Design Team 3

Date: 4/25/2012



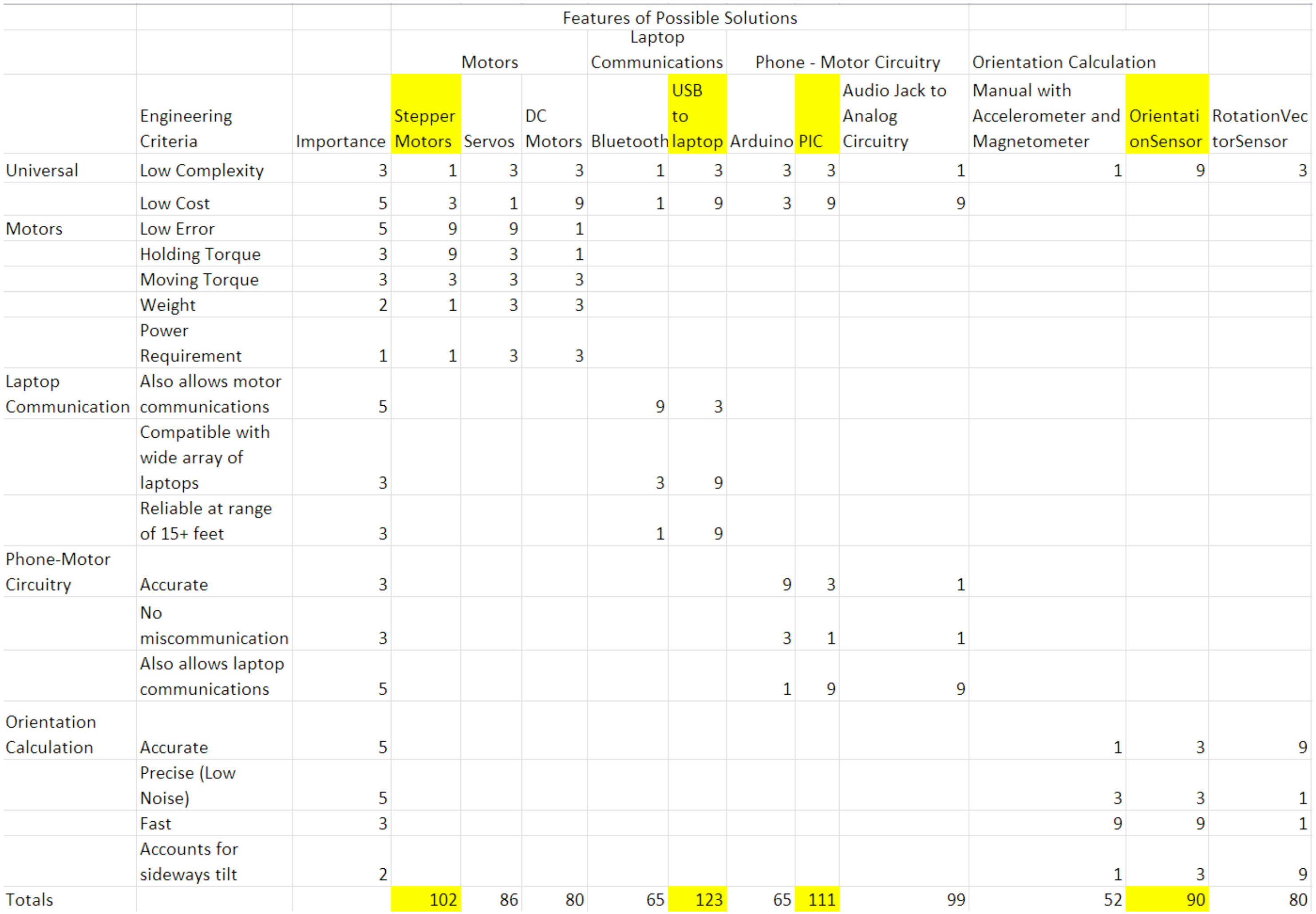
Legend		
⊙	Strong Relationship	9
○	Moderate Relationship	3
▲	Weak Relationship	1
╪	Strong Positive Correlation	
+	Positive Correlation	
▬	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

			Column #		1	2	3	4	5	6	7	8	9	10	**Competitive Analysis**	
			Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)		▲	▲	▲	▲	▲	▲	▼	▼	▼	▲		
Row #	**Max Relationship Value in Row**	**Relative Weight**	**Weight / Importance**	**Quality Characteristics** (a.k.a. "Functional Requirements" or "Hows") / **Demanded Quality** (a.k.a. "Customer Requirements" or "Whats")	180 Degrees either direction from center	Reliability	Rotation Speed	Accuracy of mount'ss Orientation	Accuracy of calculated bearings	Accuracy of photo in relation to target position	Total Cost	Time to achieve target orientation	Weight	Utilize a smartphone's sensros and compting power	Our Company	Geo-Pointing Module by FLIR
1	9	14.7	5.0	360 Azimuthal Range	⊙	○						⊙	○		5	5
2	3	5.9	2.0	0 - 90 Polar Range		○						○	○		4	2
3	9	14.7	5.0	Securely Holding and Rotating 30lb load		⊙	⊙			▲		▲	⊙		5	5
4	9	11.8	4.0	Capable of rotating above and below horizon		○				⊙		○	○		5	2
5	9	11.8	4.0	Automatically sense own location		○			⊙	⊙				⊙	5	0
6	9	11.8	4.0	Automatically sense own orientation		⊙		⊙		⊙		▲		⊙	4	3
7	9	2.9	1.0	Acquire context imagery										⊙	5	0
8	3	14.7	5.0	Generate an image acquisition command signal to attached camera/laptop										○	5	0
9	9	11.8	4.0	Not exceed budget of $500							⊙			⊙	5	0
				Target or Limit Value	190 degrees in either direction	Repeatable tests	>20 deg/second	<2 deg	<.5 deg	<2.5 deg	<$500	< 30 sec	< 15 lbs (not including infrared camera0			
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)												
				Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9		
				Weight / Importance	132.4	370.6	132.4	105.9	105.9	332.4	105.9	211.8	229.4	388.2		
				Relative Weight	6.3	17.5	6.3	5.0	5.0	15.7	5.0	10.0	10.8	18.4		



Powered by QFD Onli

Appendix 7 - Solution Selection Matrix



			Features of Possible Solutions										
			Motors			Laptop Communications		Phone - Motor Circuitry			Orientation Calculation		
	Engineering Criteria	Importance	Stepper Motors	Servos	DC Motors	Bluetooth	USB to laptop	Arduino	PIC	Audio Jack to Analog Circuitry	Manual with Accelerometer and Magnetometer	OrientationSensor	RotationVectorSensor
Universal	Low Complexity	3	1	3	3	1	3	3	3	1	1	9	3
	Low Cost	5	3	1	9	1	9	3	9	9			
Motors	Low Error	5	9	9	1								
	Holding Torque	3	9	3	1								
	Moving Torque	3	3	3	3								
	Weight	2	1	3	3								
	Power Requirement	1	1	3	3								
Laptop Communication	Also allows motor communications	5				9	3						
	Compatible with wide array of laptops	3				3	9						
	Reliable at range of 15+ feet	3				1	9						
Phone-Motor Circuitry	Accurate	3						9	3	1			
	No miscommunication	3						3	1	1			
	Also allows laptop communications	5						1	9	9			
Orientation Calculation	Accurate	5									1	3	9
	Precise (Low Noise)	5									3	3	1
	Fast	3									9	9	1
	Accounts for sideways tilt	2									1	3	9
Totals			102	86	80	65	123	65	111	99	52	90	80