Dynamic Projector Mount Project - Design Document

Andy Kwan, Dante Ye, Siamak Rahmanian, Smaran Karimbil

University of British Columbia

EECE 409/429/419/439/400/469

August 30, 2017

Design Document

ELEC 491-Capstone Proposal
Dynamic Projector Mount Project

Andy Kwan [redacted]
Smaran Karimbil [redacted]
Siamak Rahmanian [redacted]
Dante Ye [redacted]

Executive Summary

Our client, Tim Herron, who works at the BC Hydro Theatre in the UBC CIRS building uses multiple projectors for visualization presentations that require more than one projecting image. The projectors are currently mounted to the ceiling via unistrut fixtures; these hang at a height of 4 meters above the ground. This results in a great deal of inconvenience when adjusting and configuring the orientation of these projectors. To solve this problem, we have designed and created a dynamic projector mount that is remotely controlled. Our projector mount will allow users to adjust the positioning of one or more projectors using touch control on an iPad. This is done via electric motors coupled with mechanisms using belts and pulleys to pan and tilt the projector mount. The motors are controlled using a microcontroller that receives signals from an iPad through wireless technology. Our current prototype has fully functional pan movement and functional tilt movement that can be further refined and optimized for a fully operational prototype. The total cost of this projector mount is $1155.

Table of Contents

Executive Summary 0

1. Introduction 5

2. Mechanical Components 6

2.1 Structural Frame 6

2.1.1 Material 6

2.1.2 Bearing 7

2.2 Mechanism 8

2.2.1 Tooth Profile 8

2.2.2 Gear Ratio 8

2.3 Prototyping 9

3. Electrical Components 10

3.1 Electrical Machine 10

3.1.1 Motor 10

3.1.2 Motor Driver 10

3.2 Sensors 11

3.2.1 Feedback using potentiometer 12

3.2.2 Feedback using Rotary Encoders 13

3.3 Microcontrollers 14

3.4 Connectivity: 15

3.5 Power Distribution 16

4. Software 17

4.1 Bluetooth 17

4.1.1 CoreBluetooth 17

4.1.2 Establishing Connection 18

4.1.3 Read/Write 18

4.2 iOS 18

4.2.1 Graphical User Interface(GUI) 18

4.2.2 Functionality 20

4.2.2.1 Slider 20

4.2.2.2 Stepper 20

4.2.2.3 Label 21

4.2.2.4 Button 21

4.2.2.5 Segmented Control 21

4.2.3 Data 22

4.2.3.1 NSUserDefaults 22

4.2.3.2 Real-Time Movements 22

Conclusion 23

APPENDIX 24

Appendix A- Cost Analysis 24
Appendix B- Motors 25
Appendix B-1 Comparison of motors 25
Appendix B-2 Stepper motor 25
Appendix B-3 Wiring the stepper motor driver 25
Appendix C- Sensors 27
Appendix C-1 Potentiometer 27
Appendix C-2 Rotary Encoder 27
Appendix C-3 Dynamic recalibration by rotary encoders 28
Appendix D-Wireless communication 29
Appendix D-1 Selection of Bluetooth over Wifi 29
Appendix D-2 Data stream convention 29
Appendix D -3 8 Bit data Stream Transmission 30
Appendix E Software- Arduino 31
Appendix E-1 Arduino software Flowchart 31
Appendix F Software- IOS 32
Appendix F-1 Flowchart 32
Appendix F-2 Bluetooth Peripheral Explained 33
Appendix F.4 Home Page of iOS Application 34
Appendix G- Circuit Diagram: 35
Appendix H Relative Comparison of Structural Metals 36
Appendix I Relative Comparison of Structural Metals 37

References 39

Table of Figures

Figure Number	Figure Name	Page Number
Figure 1.1	System Block Diagram	5
Figure 2.1	Gimbal Style Structural Frame	7
Figure 2.2	Pan Dual Bearing Compressin Preload	8
Figure 3.1	Position of potentiometer	12
Figure 3.2	Motor Coupled with the Rotary Encoder	13
Figure 3.3	Power Distribution Circuit Diagram	16
Figure 4.1	Block Diagram of an Example Peripheral & Central	17
Figure 4.2	Projector 1 View with User Interface Elements Identified	19
Figure 4.3	Home View	20
Figure 4.4	Real-Time Movement Block Diagram	22

1. Introduction

Our team's project is to prototype a dynamic projector mount that allows a user to remotely adjust the positions of a projector. This is project involves three broad subsections, the mechanical design, electrical design, and software design. The projector mount will be designed to rotate along a vertical and horizontal axis. The movement of the mount is handled by two electrical machines, which are powered by drivers that are controlled accurately through sensors using microcontroller. The rotation along the horizontal axis will move the projected image from the projector vertically along the wall which is the tilt movement. The rotation along the vertical axis will move the projection horizontally on the wall which is the pan movement. The microcontroller should communicate with an iPad remotely, so an iOS app will be built to interface human interactivity with our design. The app must also provide a simple but effective interface for the user to select and control the desired movement.

In the block diagram below, the iPad is shown to connect to an Arduino via a bluetooth module. This arduino will drive two stepper motors through stepper motor drivers. The motors in turn are connected to belt and pulley systems which will rotate our mount. For precise movement we will be using rotary encoders that provide feedback about the position of the motor to the arduino. This will allow us to ensure that the motor moves accurately to the desired position. With this design a user can control the orientation of projectors via an iPad.

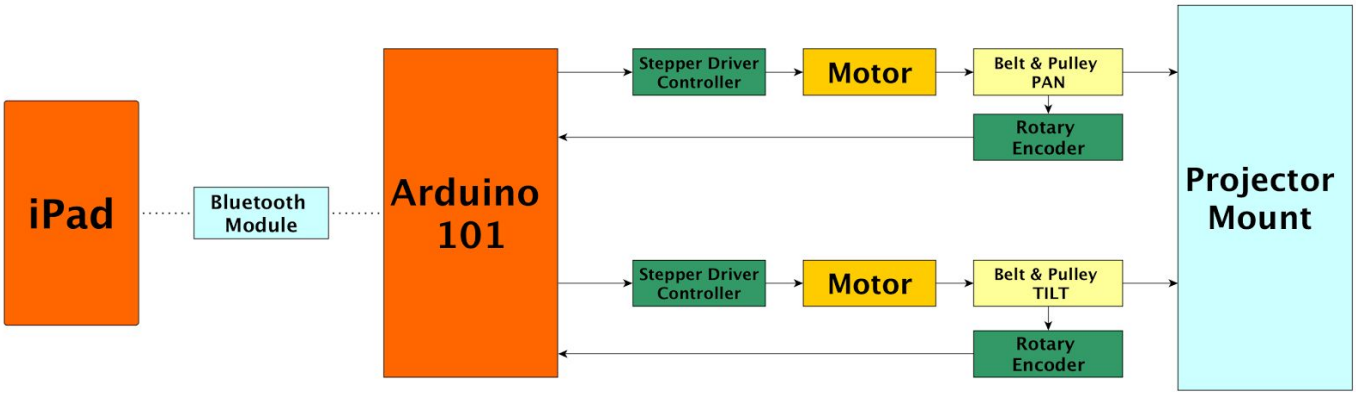

Figure 1.1 - System Block Diagram Project Overview

2. Mechanical Components

The mechanical components of the dynamic projector mount consists of two sections: a structural frame that carries the projector while constraining the relative motion between two parts so that the pan and tilt movements can be achieved to satisfy our requirements (Functional Requirement 1 and 2), and a mechanism to link the rotational movement from the remotely operated motor shafts to the pan and tilt motion of the projector mount to satisfy our requirement (Functional Requirement 3).

2.1 Structural Frame

The dynamic projector mount structural frame was designed using a gimbal implementation with dual pivot support for the tilt axis and a single hanging pivot for the pan axis as shown in Figure 2.1 below. This gimbal style allows for continuous infinite pan and tilt motion without any mechanical restrictions on the motion sweep angles, thus surpassing our requirements (Non-Functional Requirements 4 and 5).

2.1.1 Material

The structural frame is designed to be prototyped and fabricated out of aluminum and thermoplastics. Aluminum was chosen after comparing against other structural metals and a table showing the comparison is in Appendix H. Aluminum has such good structural properties that it is even made into structural pieces commonly available as 80-20 T-Slot. These structural members are made with a 'X' shape to emulate the strength of '**I**' beams while also implementing slotted features for fasteners such as nuts and bolts, making them ideal for structural frames. The dynamic projector mount frame is engineered around using aluminum 80-20 T-Slot. 80-20 T-Slot is low cost, can be designed to be strong enough to carry a payload of 80 kg, easily available, and lightweight, meeting our requirements (Non-Functional Requirements 2 and 7, Constraints 1, 2, and 3). To obtain the gimbal style dynamic projector mount, 7 main 80-20 T-Slots are used as shown in Figure 2.1 below. Aluminum cut plates are used to join the structural frame together and hold the pan bearings in place. Two PLA plastic parts are used to hold the tilt bearings in place. Aluminum cut plates are designed with extruded cut features for the mounting standards satisfying our constraints (Constraint 7 and 8). The main shafts that carry the payload are chosen to be 12mm steel metric grade 10.9 bolts and are commonly available. These have the strength to hold up 180 kg. With the mount weighing less than 50kg, there is slightly more than a safety factor of 3 for a 40kg projector, surpassing our requirement (Non-Functional 2).

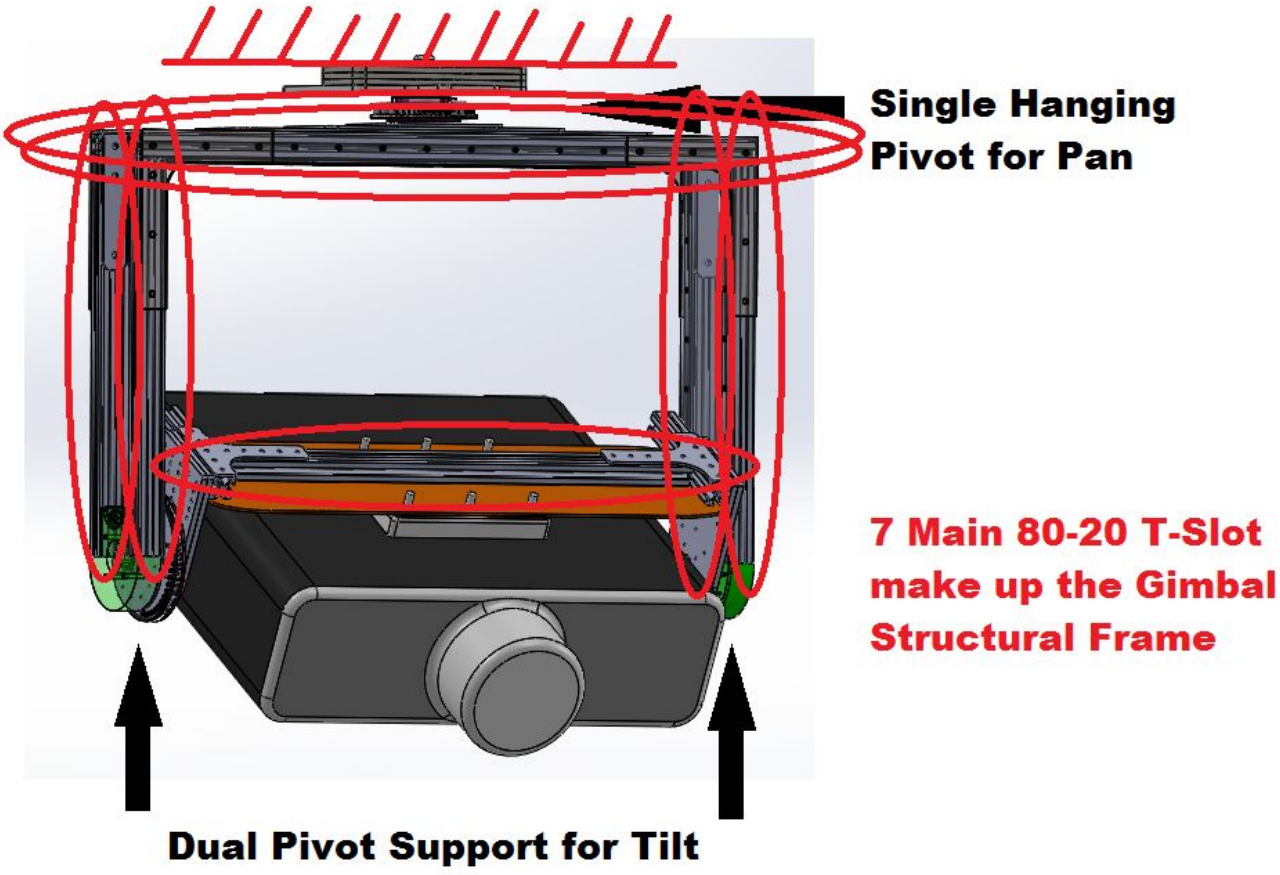

Figure 2.1 - Gimbal Style Structural Frame

2.1.2 Bearing

The Dynamic Projector Mount was designed with bearings, which have lower friction than bushings, to allow for smooth pan and tilt motions, as per our constraint (Constraint 4). The bearings used are 6201RS ball bearings and were chosen to be compatible with the 12mm shaft size. These bearings are a standard size that is used in many drive train applications making them widely available and low cost. This is important in order to satisfy our constraint of receiving bearings in the given timeframe (Constraint 3), keeping cost to a minimum to stay within budget (Constraint 2), and it's already mass manufactured (Non-Functional Requirement 9). Bearings also allow preload in contrast with bushings which don't. The structural frame implements a compressive preload design that uses two bearings as seen in Figure 2.2. This preload is adjusted to meet and maintain the smooth and precise rotational pan movement, as per our requirement (Non-Functional Requirement 3).

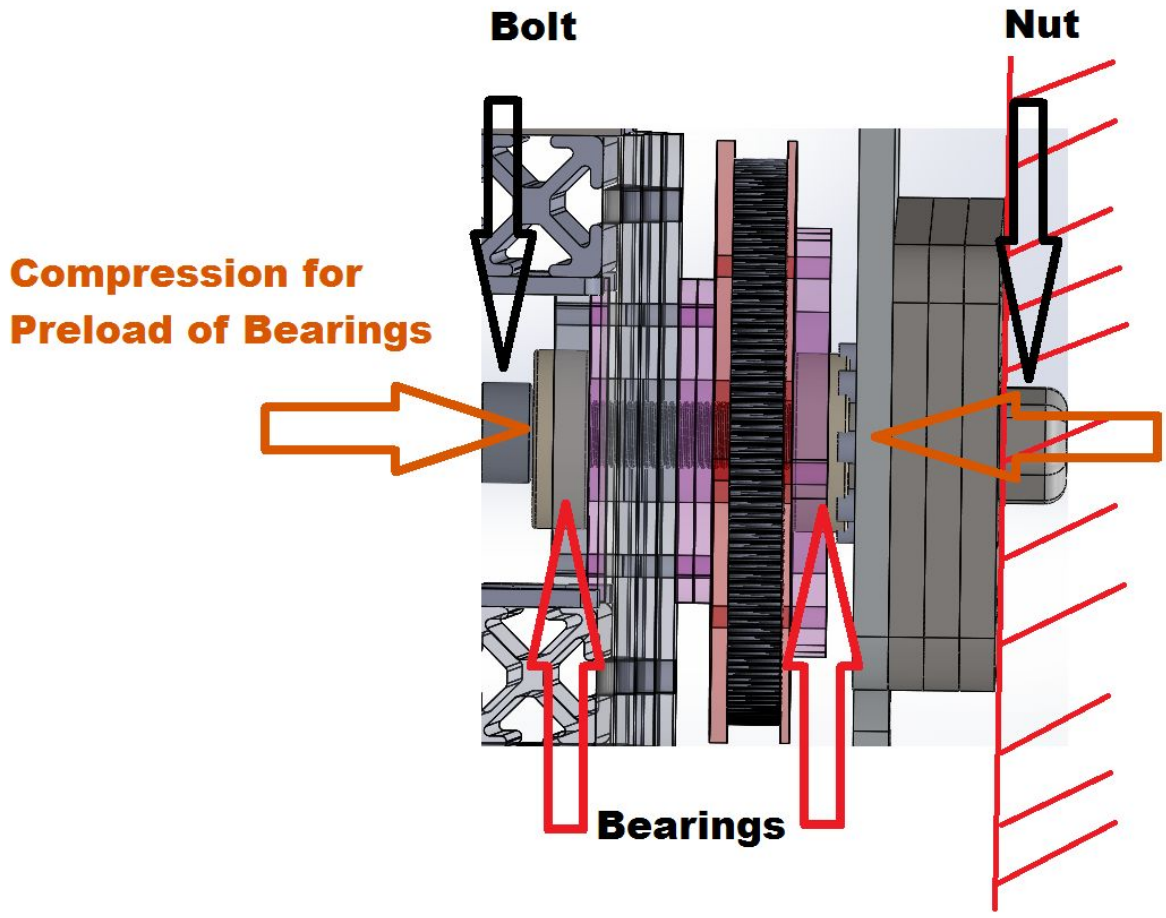

Figure 2.2 - Pan Dual Bearing Compression Preload

2.2 Mechanism

The mechanism for panning and tilting uses a belt driven pulley system. This mechanism allows the transmission of rotating shafts in motors to position rotational movements in the horizontal and vertical axis resulting in powered pan and tilt movements. The belt drive was chosen after comparing the alternatives and a table showing the comparison is in Appendix I.

2.2.1 Tooth Profile

A GT2 curvilinear tooth profile was chosen after comparing it against both the trapezoidal and the HTD curvilinear tooth profile. The semi-round curved tooth shape of the GT2 tooth profile gives higher precision and anti-backlash properties necessary for precision movement, as per our requirement (Non-Functional Requirement 3). This tooth profile is also rated for higher torque and load carrying capacity which is necessary as per our requirement (Non-Functional Requirement 2). [6] The GT2 tooth profile is a standard for positioning timing belts and pulleys so it is commonly

used in various positioning applications, as a result it is widely available and low cost. This is very important in order to satisfy our constraint of receiving belts and pulleys in the given timeframe (Constraint 3), keeping cost to a minimum to stay within budget (Constraint 2), and it's already mass manufactured (Non-Functional Requirement 9).

2.2.2 Gear Ratio

A gear ratio is used for both the pan and tilt mechanism. The gear ratio chosen for the pan mechanism is 8:1. Where the driving pulley used on the motor is 20 teeth and the driven pulley on the structural frame is 160 teeth. The driving pulley was chosen to be 20 teeth because it is the lowest teeth pulley available that fits the motor. The gear ratio for the tilt mechanism is 10:1. Where the driving pulley is also 20 teeth and the driven pulley is 200 teeth. These gear ratios increase the precision of pan and tilt motion. Appendix B-2 shows the angles between the precision of the motor rotation and the structural mount rotation for both pan and tilt. The gear ratios also increase the torque applied to the structural frame to move the mounted projector. Since the max torque from the motors is 4 $kg*cm$ the max torque applied to the pan motion is 32 $kg*cm$ and the max torque applied to the tilt motion is 40 $kg*cm$. The gimbal style structural frame results in a moment of inertia for the pan of 1.4 $kg*m^2$ and a moment of inertia for the tilt of 1.6 $kg*m^2$. This means the tilt motion requires higher torque to achieve the same acceleration as the pan, hence the higher gear ratio in the tilt mechanism. This results in a max acceleration of $0.5°/s^2$, for each movement, and when coupled with controlled acceleration the projector mount moves smoothly and precisely, as per our requirements (Non-Functional Requirements 3, 6, and 7).

2.3 Prototyping

The Dynamic Projector Mount prototype was fabricated with 80-20 T-Slot cut to length using a bandsaw followed by manual grinding to obtain the exact desired lengths. This process was used because of the easy availability of these tools, and given our timeframe constraint (Constraint 3) this was the best process. The aluminum structural frame plates were CNC Waterjet cut for accuracy to achieve the precision movements as per our requirements (Non-Functional 3). The bearing holders for the tilt motion required a more complex part so that the payload requirement is satisfied (Non-Functional 2). These parts were designed for 3D printing to allow fast prototyping in order to fabricate these parts within our given timeframe (Constraint 3), and can be easily converted for injection molding for mass manufacturing (Non-Functional 7).

3. Electrical Components

The electronic components of the projector mount, mainly consist of the microcontroller to communicate with the iPad wirelessly, motors that will act as actuators to bring about movement and potentiometers and encoders for feedback to the controller. The circuit has also been designed to operate from a power outlet so that the user will not have to face the inconvenience of replacing or recharging batteries. For this we will have regulators and converters to get the desired voltages for operating the different components. The description of the protoboard-circuit connection can be seen in Appendix G.

3.1 Electrical Machine

3.1.1 Motor

To bring about changing the horizontal angle and vertical angle of the projector, (which is a part of our functional requirement 1 and 2), we are using motors. After comparing various types of motors, as seen in Appendix B, we have decided that the best motor for our use is stepper motors.

This type of motor is very common for precision positional actuation because the motor design incorporates small discrete steps where the motor can hold its position. In most common stepper motors the commutation is done electrically through controlling the switching of forward and reverse voltages and currents at different states on two phases. This makes driving the motor a little bit more complicated but because there are discrete positions at which the motor stops at. This makes it necessary to use a motor driver which we will control with our microcontroller. The motors that we use have 200 discrete steps.

3.1.2 Motor Driver

To accurately control the stepper motor we have selected to use an A4988 stepper motor driver. This driver allows microstepping for upto 1/16th of a step. This means our motor will have 3200 steps per rotation therefore it is possible to rotate with a precision of approximately 0.01° as calculated in Appendix B-3, which meets our precision movement requirements(Non-functional requirements 3 from our requirements document). For the stepper motor driver that we use there are two logic inputs that we can use from our microcontroller, one for stepping the motor which is driven by rising edge of a signal, i.e when the logic goes from 0 to 5V and another input that determines the direction of rotation. Once we wire the motor driver to the motor as explained in Appendix B-2, we can proceed to control the movement of the motor from the microcontroller.

3.2 Sensors

The stepper motor can be accurately controlled and positioned, but despite this, it is important to detect horizontal and vertical angle through sensors. The two main reasons for having sensors is to allow precise position control, so as to not depend on calculated positions based on the steps of the stepper motor for the projector which would have a possible degree of error due to external oscillations or disturbances. We also need to know the starting position of the projector mount for accurate control. Calculating the starting position will help ensure that there is no attempt to turn on motors that would move the projector beyond its given scope of rotation thus setting soft limits on the movement. For our design we use a potentiometer to read our initial position and rotary encoders to accurately measure our subsequent movements.

3.2.1 Feedback using potentiometer

One of the most common ways to determine the position of a motor or any small rotating electronic component, is by using a potentiometer. Potentiometers are generally low cost but have a low accuracy. The potentiometer we have selected is the 3590S-2-103L , which is a 10 turn potentiometer and a 5% tolerance[2]. This potentiometer has an output which is a voltage that is linearly dependant on the position of its shaft. The limits of its output voltage are between its ground and input voltage. This output voltage can be read by the Analog to Digital Converter(ADC) of the microcontroller. This will allow the absolute position of the shaft to be determined at any given instant. In our design the potentiometer for pan movement will be fixed between the two relative motion components that is the panning bolt and the horizontal distribution beam assembly, where the bolt is fixed. This can be seen in figure 3.1 As seen in our calculation in Appendix C-1 the potentiometer can give us a reading with a precision of ±1.8°. Note that the tilt mechanism will not be using a potentiometer as we will always return it to the starting position before powering down, because of the way it is balanced.

Figure 3.1 Position of potentiometer

3.2.2 Feedback using Rotary Encoders

As per our initial design requirements we needed a movement precision of 0.1°, which is higher than the precision of the potentiometer. For movements of higher precision we will use be using the following rotary encoder, E6B2-CWZ3E which has a precision of 1024 pulses per rotation. This encoder is coupled to the shaft of the motor and to calculate the position of the mount we will need to multiply our results by the gear ratio. Appendix C-2 shows that it can be used detect 0.04375° of movement of the projector, this exceeds our requirement of 0.1° of precision(Non functional requirement 3). This encoder is a relative encoder and has to be regularly monitored with hardware interrupts to calculate any change in position.

Even though the rotary encoder meets our expectations, we will use both the potentiometer and the rotary encoder. This is mainly due to the fact that the rotary encoder is used to measure movement that is relative to our initial position and it is important to know our starting position. The error from our starting position is corrected by using the Z pin from the encoder as seen in Appendix C-3.

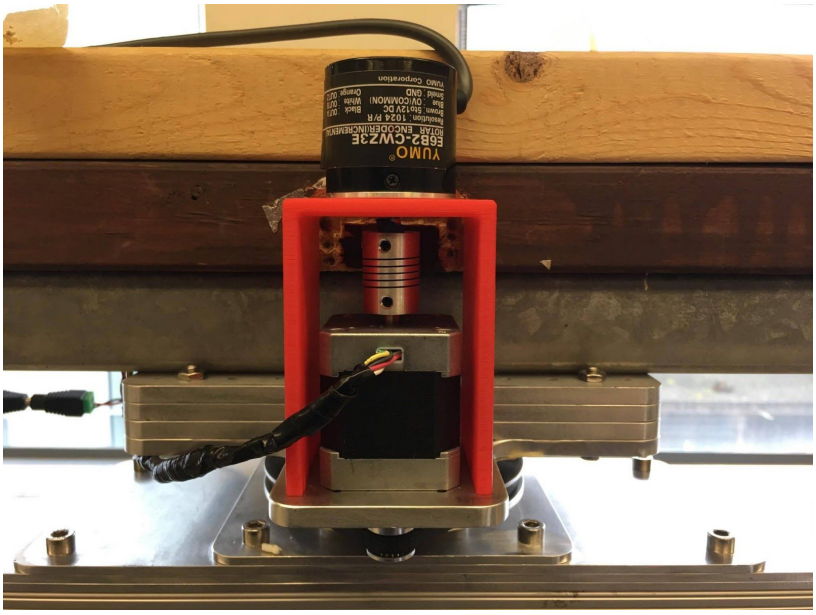

Figure 3.2 Motor Coupled with the Rotary Encoder

3.3 Microcontrollers

One of the most important parts of the electronic system is the microcontroller which controls the positioning of the mount. The microcontroller will be connected to the sensors, the motors as well as a wireless module that communicates with an Ipad. The microcontroller will receive a wireless signal to change the position of the projector mount, it will then determine which motors must be used to reach the new position. It will then read its current position from the sensors and turn on on off the motors sequentially to move the projector mount to the desired position. There are many microcontrollers that are viable options for this project, we have selected the Arduino 101 (marketed as the Genuino 101 outside the North America)[9].

The Arduino 101 comes with an inbuilt programmable bluetooth module which we have decided to take advantage of. This bluetooth chip is programmed with the CurieBLE library[10]. The Arduino 101 comes with a 32MHz microprocessor with an intel 32 bit architecture. These specifications allow us to conveniently implement communication protocols. The other features we use on the board are its input and output pins to control the motors and read encoders, and an inbuilt ADC to read our potentiometer. We also use a timer to control the speed of the motors and four hardware interrupts to read the two encoders precisely.

These specifications allow us to meet all requirements(functional requirements 1, 2 and 3) of movement as well as satisfy our constraints of wireless communication with an iPad.

3.4 Connectivity:

For our microcontroller to receive inputs that control the position of the projector mount, it must communicate with an iPad wirelessly (Functional requirement 3 and constraint 4). There are many methods of wireless communication, the two most commonly being WiFi and Bluetooth. For our project, we are using the Arduino 101 which will have a built-in Bluetooth module. This is a Bluetooth 4.0 Bluetooth Low Energy chip which has a data transfer rate of 0.27MBps and a range of 50m. This module is a UART (Universal Asynchronous Receiver and Transmitter), which sends and receives data not based on a clock, to communicate with other bluetooth devices. This communication protocol is supported by every iOS device. Currently the data is being sent as a stream of packets which is 8 bits wide. The range of the bluetooth module will be limited by the range of the Ipad, which is stated as 10m, but when tested can reliably communicate from distances up to approximately 30 m. To communicate various positions and movements we have come up with a data scheme as seen in Appendix D. Bluetooth communication is sufficiently fast to send and receive signals that will allow the user to see the effects of the movement of the projector immediately after selecting a new position.

3.5 Power Distribution

To design the power system for the components in the circuit we must know the individual power rating of every component. From the motor specifications[5] we know that the motors will operate from 15V-30V DC and the microcontroller will operate from 7V-12V DC [9] and sensors will operate at 5V DC[1][2][3]. For the convenience of setting up our projector mount we designed our system to powered from a common power outlet, just like the projector(Non-functional requirement 6). Our system has only DC components, allowing us to use a simple AC-DC rectifier such as a laptop changer. We have selected one such rectifier which will have an output of 19V DC, 3.75A. Once we have 19V DC, we can use voltage regulators for the different levels of voltages in the system. As seen in the below figure 3.3, we have three different DC Voltage levels in our system, the arduino 101 which runs at 9V DC with the help of the LM7809[8], the control and sensing components which operate at 5V DC with the LM7805[7] and motors which operate at 19V DC which is takes directly from the AC-DC rectifier. It can be noted that if we wanted to cool the system, a small fan to cool the system can be powered from the 9V DC bus. The power distribution diagram refers to Figure 3.3. We tested our system to find that the maximum current is 1.5A, this is less than the 3.42A provided by our rectifier. This translates to 28.5W of power consumption.

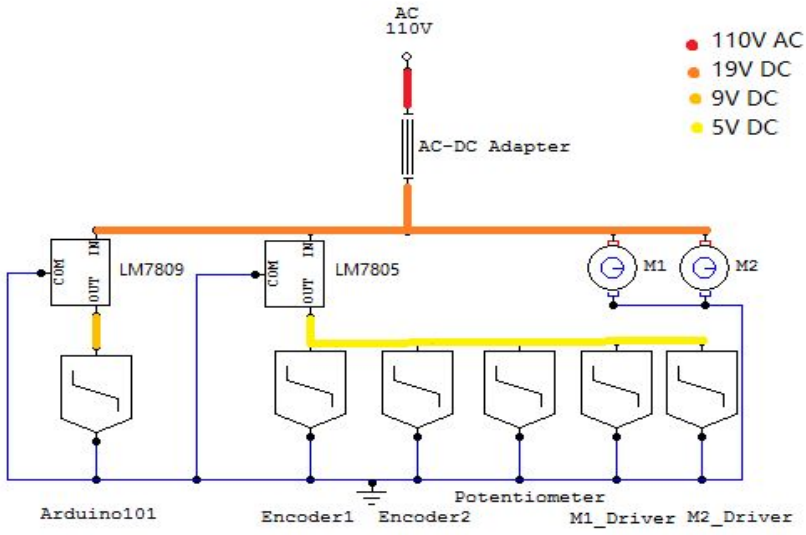

Figure 3.3 - Power Distribution Circuit Diagram

4. Software

As per our constraint of controlling the movements of the projector mount by an iPad, iOS software is to be used to implement the wireless control. The wireless method to be used for communication between the iPad and Arduino 101, is to be Bluetooth. Apple offers various API's that assists our design.

4.1 Bluetooth

Figure 4.2 illustrates the block diagram of the connections between a peripheral, services, characteristics and central. In our design, the Central is the iPad, the Peripheral is the Arduino, with a single Service and two characteristics which are read characteristic and write characteristic. For further understanding of Bluetooth Low Energy, refer to Appendix F-3.

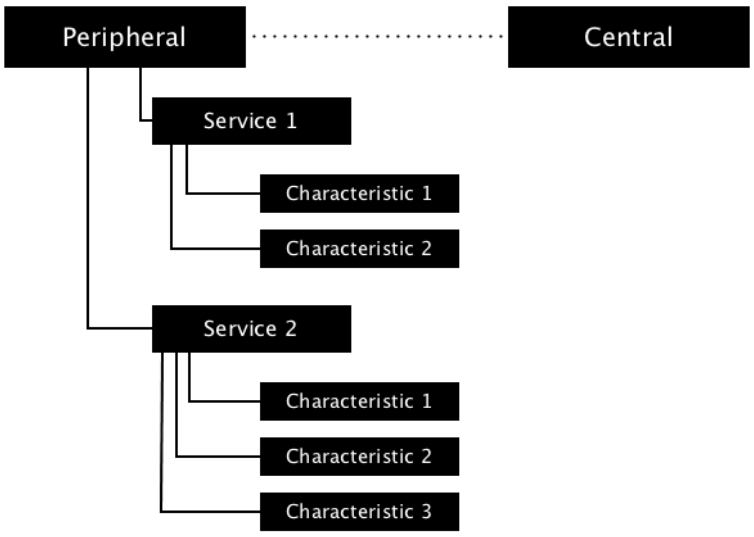

Figure 4.1 Block Diagram of an Example Peripheral & Central

4.1.1 CoreBluetooth

We chose to take advantage of Apple's CoreBluetooth API to set up our two-way communication link. After a connection has been established with the services of our peripheral, we are ready to read and write values to and read from our Arduino 101's characteristics respectively.

4.1.2 Establishing Connection

To establish a connection with the projector, we created a service in the Arduino 101 (which is our peripheral). In order to not mix up the specific services we are trying to connect to with other possible services of other peripherals in the neighbouring area, we assign a Universally Unique Identifier (UUID) to our target service. The Arduino's bluetooth module will now go into *advertising mode* to allow a central who has been given its UUID to be able to find that service and and establish a connection.

4.1.3 Read/Write

There are two simple ways to communicate with the Arduino 101, reading from and writing to our peripheral. Interestingly, these two methods work differently from one another and thus we shall first examine them in detail.

- ***Writing -*** To send values to our Arduino we must first ask the peripheral to give us a list of all of its services, and the way the peripheral gives this information is by sending all of its services Core Bluetooth Unique Identifier (CBUUID) that is attached to each service. By requesting and organizing these ID's, one can then use the CoreBluetooth appropriate methods to write values to the characteristic they are interested in.
- ***Reading -*** To read values from a characteristic, we must firstly turn on the ability to accept notifications from the characteristic we want to read values from. A notification operates in a way that the characteristic will notify its parent service that its value has been changed and therefore the service will notify the central of this change. Thus, the central will then "listen" for this change of value and store it in the appropriate memory location.

4.2 iOS

4.2.1 Graphical User Interface(GUI)

In iOS application development the screen that the application will display its content on is referred to as a *View* and the file that controls the *View* is the *View Controller.* Apple provides many user Interface (UI) elements in order to create simple and familiar applications for the user. The elements provided by XCode* that we utilized are:

- Label (Static/Dynamic)
- Button
- Slider
- Stepper

- Segmented Control

Figure 4.2 shows a screenshot of the projector View Controller and the related elements listed above.

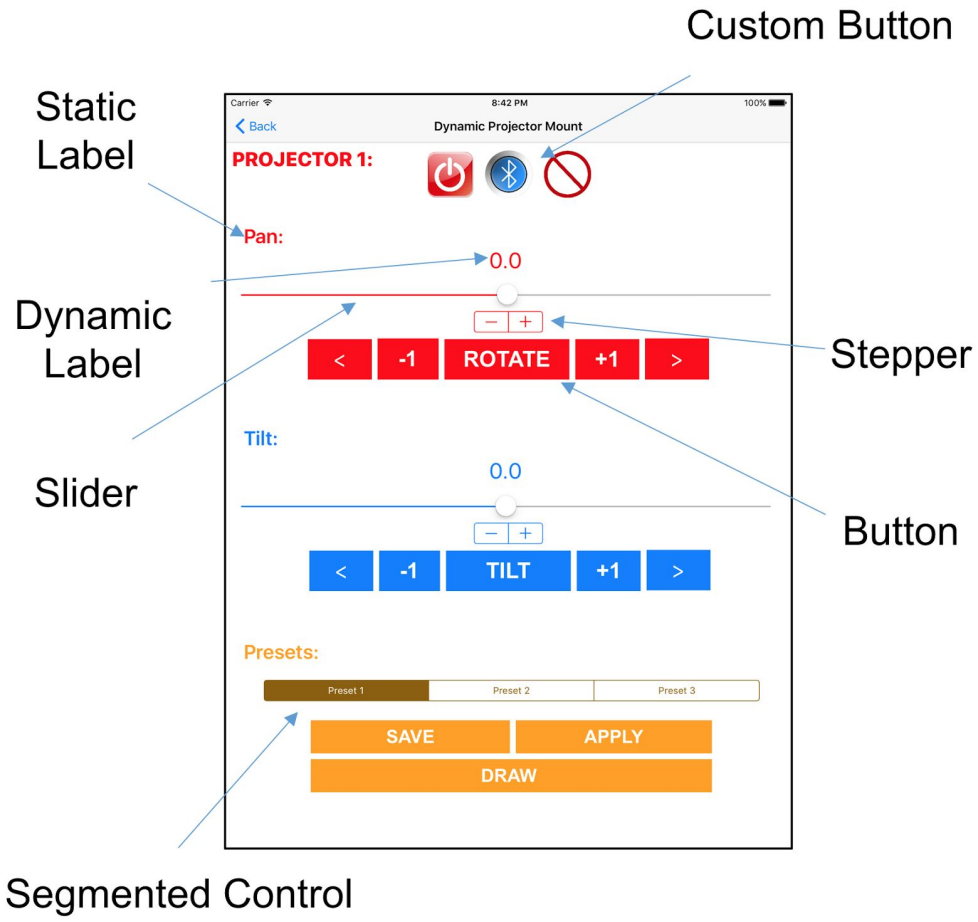

Figure 4.2 - Projector 1 View with User Interface Elements Identified

To further enhance the user experience, we took advantage of multiple Views to separate and organize the application content. Initially, the app opens to a *Home* page - Figure 4.3 - which lists all the available projectors to be controlled using buttons. Furthermore, a *Settings* button is available to the user for changes to names of presets.

Once the desired projector is selected, the application will take the user to the *Projector View* - Figure 4.2 - which contains all of the controls needed to operate the projector mount.

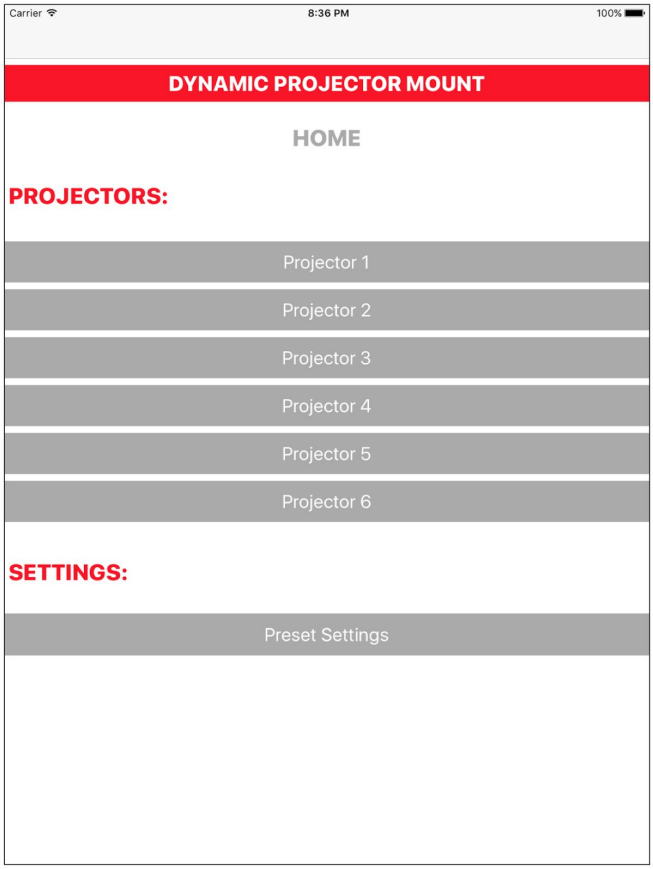

Figure 4.3 - Home View

4.2.2 Functionality

4.2.2.1 Slider

As per the requirements, the selection of the angle of attack of the projector's image onto the wall needs to obtain an accuracy of 0.1º. For fast selection, a slider is used as the coarse angle selection tool with a 1º accuracy. The limits are -180º to +180º for the pan and -90º to +45º for the tilt. The ability to point the projectors image onto the floor (-90º) is a requirement demanded by our client.

4.2.2.2 Stepper

The purpose of the stepper is to give the user the ability to control the fine angle adjustments at an accuracy of 0.1º ,as per the requisite. This UI element is to be used in conjunction with the slider to control either the pan or tilt with the accuracy required. Both the stepper and slider are synchronized and are dependent on each other's values.

4.2.2.3 Label

- **Static Labels -** Static labels are ones that do not change during application operations. They are just informing the user of what other dynamic elements are there to do. They are informative labels.
- **Dynamic Labels -** Dynamic labels are labels that change. The pan and tilt labels displays the current degree value of the projector's pan and tilt respectively and also the desired value that the user will select before deciding to trigger the movement.

4.2.2.4 Button

i - There are many buttons used in our application. The main buttons are labeled as *ROTATE* for pan and *TILT* for tilt. Once these buttons are pressed, a set of instructions will be transmitted through the Bluetooth Low Energy (BLE) to the Arduino 101 to move the projector to the user's desired position. These buttons single-action thus as soon as they user presses these buttons, the instructions will be sent. Also, a *Connect* button is utilized to connect the iPad to the Arduino 101's Service incase it disconnects after initial launch. This button - along with the *STOP* and *Kill* buttons - are using images for their button graphics.

ii - Moreover, multi-action buttons are also utilized in our design. As per the requirement (Constraint 10), real-time movements are designed by taking advantage of dual-action buttons. When the user presses down on a button and keeps finger on it, the projector mount will continuously move at a constant speed. It will continue to move until the user lifts their finger from the button. A dedicated value is sent to the motor when the user presses and holds the button, and another value is sent to the motor when the button is released. Using this method we send the minimum amount of data to satisfy the Constraint 10.

iii - Other buttons are used as segues to change the view of the application. As outlined in Constraint 9, a settings page is designed to allow the user to change the names of position preset.

iv - As per the requirements (Non-Functional Requirement 3) a *+1* and *-1* single-action button is implemented to move the projector by 1º in either pan or tilt direction.

4.2.2.5 Segmented Control

This GUI element is a selectable switch. For our design, a three-element segmented control is utilize to satisfy our requirement (Constraint 9) of have positional presets. The preset function allows the user to save their current pan and tilt position in a segmented control element by pressing the *SAVE* button. Once the user is ready to point the projector to that specific direction that they saved previously, the corresponding segmented control element is chosen and the *APPLY* button is pressed to move the projected to their desired position.

4.2.3 Data

4.2.3.1 NSUserDefaults

In order to pass data between the different views, our design utilizes the NSUserData API to store and read simple data on the iPad. Given the requirement (Constraint 9) to change the positional presets names, which is in a different *View* than the *Projector View,* storing and retrieving data is simplified by using NSUserData. In our design, only Strings are being stored and retrieved using NSUserData.

4.2.3.2 Real-Time Movements

As stated in the requirements (Constraint 10) the real-time movement is designed to not overload the Arduino 101 with data if implemented by sending a continuous stream of data as the user holds the corresponding button. Our solution - as explained in section 4.2.2.4-ii, requires only one value to be sent when the button is pressed, and another value sent when the button is released. This causes an issue for displaying the real-time position in the pan and tilt labels because the iPad has no reference of the projectors position while the button is being pressed.

Using NSTimer, we calculated the period-per-degree (n) of the projector's motion. Using this value a timer is then used to call a label-updating-function every n^{th} second. This gives the illusion that the label is being updated in real time but in fact while the button is being pressed, the Arduino 101 and the iPad are operating independently. A block diagram of this function is shown in figure 4.4 below.

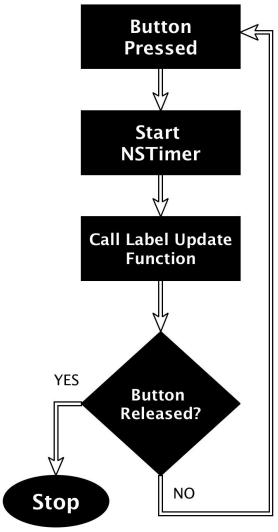

Figure 4.4 - Real-Time Movement Block Diagram

Conclusion

The dynamic projector mount had to been designed as per our requirements while keeping in mind any constraints. To summarise this project, a projector mount has been designed to be controlled by an iPad which will communicate with an Arduino over bluetooth. Through the iPad user interface a user will be able to select an orientation for a projector. This would be achieved through two movements, Pan and Tilt. The range of movement for the Pan motion in our design is 5 revolutions(because of limitations of cable length) but has been limited to -180° to 180° degrees by soft limits. There is full rotational capabilities by the tilt mechanism and has been restricted -90° to 45° degrees. Our design can pan and tilt rotate with an accuracy of approximately 0.1 degrees. Once the Arduino receives this signal, one of two motors will move to arrive at a desired position. This movement is done with a feedback system through rotary encoders.

The details of cost of the project is tabulated in Appendix A. It can be seen that most of the cost of fabrication of materials is the highest. If this projector mount is to be reconstructed for future the overall cost will have to be reduced by buying parts from cheaper sources as well as carefully selecting what features can be optimized. Keeping this in mind, this project has been designed to still be expandable for future development.

APPENDIX

Appendix A- Cost Analysis

Item	Cost	Quantity	Amount
80-20 T-Slot	$20	7	$140
Structural Aluminium	$15	12	$180
Fabrication Cost	$520	1	$520
Structural Bolts	$30	1	$30
Bearing	$10	4	$40
Fasteners	$0.55	100	$55 (siamak make this)
Misc Mechanical			
Belts and Pulleys	$20	2	$40
Motors	$40	2	$80
Motor Driver	$8	2	$16
Rotary Encoders	$55	2	$110
Potentiometer	$15	1	$15
Arduino 101	$50	1	$50
Voltage Regulator 5V	$2	1	$2
Electrical Connectors	$2	6	$12
Misc Electrical			
Protoboard	$5	1	$5
Power Adapter	$20	1	$20
Total			$1315

Appendix B- Motors

Appendix B-1 Comparison of motors

There were many factors to consider to consider while selecting the motor for this project. The stepper motor was the best choice when compared to other motors mainly due to its ability to hold its position when it is being powered. The other factor factors that drove our decision were as follows.

	Stepper	Brushed	Brushless	Servo
Cost(approximate)	$30	$5	$40	$15
Accuracy of control with no Load	Very High	Low	Low	Medium
Customization	Yes	No	No	Yes

Table 2 Motor Comparison

Appendix B-2 Stepper motor

The stepper motor we have selected has 200 discrete steps. This motor is rated for 1.2A. This motor has a torque of 4N.m when it is powered[x], this will allow us to hold the projector at any required position on our tilt mechanism. When we use the microstepping from our stepper motor driver, which can achieve 16 microsteps per step. To calculate the impact of this movement on the projector we can divide the angle per step of the motor by the gear ratio for pan and tilt.

For pan movement, the gear ratio is 8:1

$$Angle\ per\ step = \frac{360}{3200} * \frac{1}{8} = 0.0140625° /step$$

For tilt movement, the gear ratio is 10:1

$$Angle\ per\ step = \frac{360}{3200} * \frac{1}{10} = 0.01125° /step$$

Appendix B-3 Wiring the stepper motor driver

For this stepper motor driver we referred to its datasheet for instructions on which signals are to be connected for its input and output pins. Firstly we will, connect the supply voltage of the motor to Vmot and ground which in our case is at 19V, as seen below it is recommended to use a capacitor across this input as it will ensure a steady voltage in case of power fluctuations. We will power this motor driver through VDD from our 5V bus, which we had used to power some of our other electronics.. Next, will connect the step and dir pins to the microcontroller, as these pins will control the stepping and direction of the motor. We will

not be using the Reset and sleep signals, so we will connect these pins to VDD as they are active low. Since our chip will always be enabled we will tie it to ground as it is also active low. To select the 1/16th microstepping which we wanted for our level of precision, the ms1, ms2 and ms3 are all set to logic 1 by connecting them to Vdd. Finally we can connect the outputs of the driver to 1A, 1B and 2A, 2B to the coils of the motor as we can see in the figure below.

To ensure that the wiring is done correctly, the step pin can be connected to a signal generator. The movement of the motor should be dependant on the frequency of the input. The direction of the movement of the motor should also change when the dir pin is moved from VDD to ground.

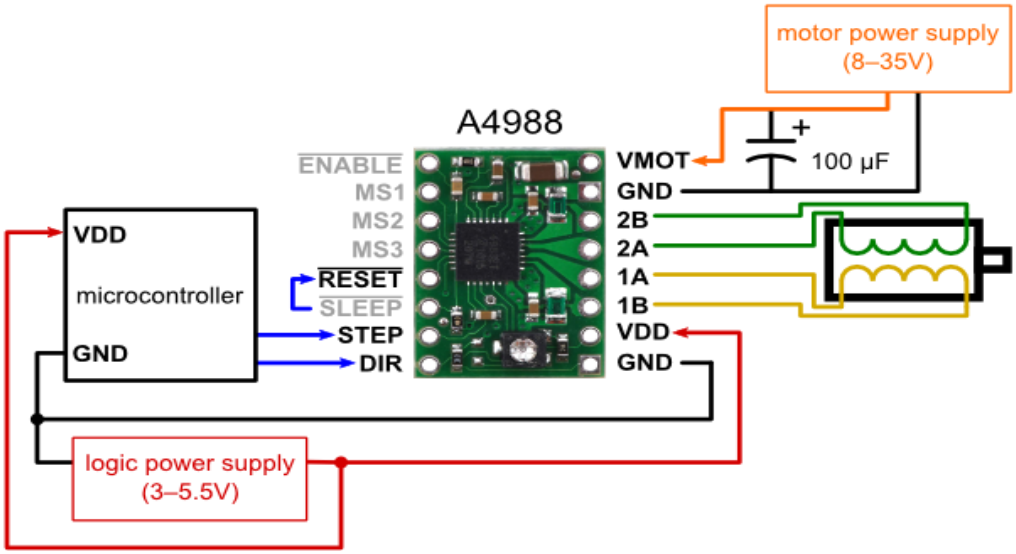

Figure 6.1:Circuit Diagram of the motor driver

Appendix C- Sensors

Appendix C-1 Potentiometer

A potentiometer is useful to find our initial starting position because its analog output pin will indicate absolute position. The 652-3590S-2-103L can be wired from the top as Voltage in, ground and output respectively. The input for the potentiometer will be from the 5V as the output range from 0-5V is safe for the Arduino. This output is an Analog signal but can be interpreted by the inbuilt ADC on the Arduino. Since the potentiometer is connected to the mount directly there will be only 1 turn. Now we can perform the following calculations to determine position as well as precision of the potentiometer. To calculate tolerance we will use the fact that the inbuilt ADC of the Arduino is 10 bit, meaning it will return a result of 0-1023 linearly. We will also use the tolerance from the datasheet of the potentiometer which is 5%

$$Tolerance = \frac{360*10}{1024} * (1 + 5\%) = 3.7°$$

Error in read value=±1.8 °

$$V\ out\ range = \frac{V\ in}{10} * \frac{Angle\ rotated\ in\ degrees}{360} * number\ of\ rotations$$

We need only 360 ° of rotation in our pan movement so we can start at any arbitrary number of rotations. If we start at 2 rotations our output will be from 1-2V.

To calculate the current position of our mount we can use the relation

$$Position\ in\ degrees = (\frac{V\ out}{5V} * 360 * 10) - (number\ of\ rotations\ in\ initial\ position * 360)$$

Appendix C-2 Rotary Encoder

A rotary encoder is a shaft encoder that can convert angular position to an analog or digital code. We have selected the E6B2-CWZ3E which is a relative encoder with one absolute position. This encoder has a precision of 1024 pulses per rotation which means it can calculate up to 0.3°. There are three output pins on the, A B and Z. When the shaft of the encoder rotates pins A and B will sequentially go high based on the direction of movement. The Z output is triggered when the encoder passes a 0 position.

To use the rotary encoder, the power supply is connected to our 5V bus, and its outputs A, B and Z are connected to our microcontroller. These pins change rapidly and have to be regularly checked. For this reason we use hardware interrupts to interpret these signals. One of pin A or B have to be connected to an interrupt which we will select to trigger on a rising edge. When this happens we can read the value of the other pin to know the direction of movement and hence calculate and update the value of the current position of the motor. The second interrupt is connected to pin Z, when it is triggered, it we know we have completed a rotation. With this information, the position of the motor is updated to an absolute value.

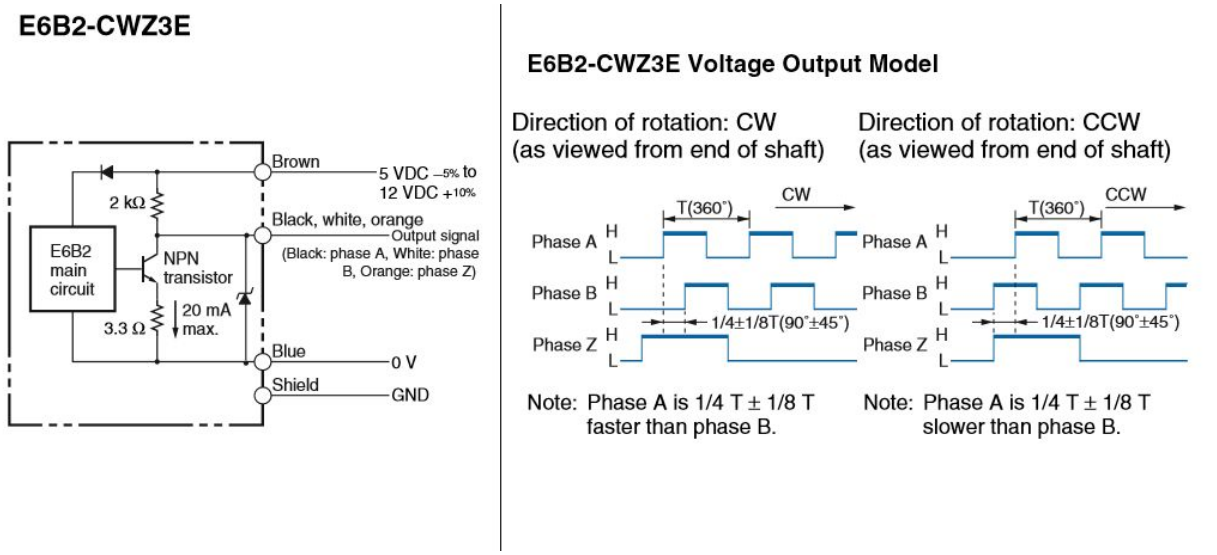

Figure 6.2 - Rotary Encoder Outputs Diagram

Since the rotary encoder will be coupled to the motor, the accuracy will have to be calculated by multiplying the current accuracy by the gear ratio.
For the pan movement:-

$$Tolerance = \frac{360}{1024} * \frac{1}{8} = 0.0439°$$

For the tilt movement:-

$$Tolerance = \frac{360}{1024} * \frac{1}{10} = 0.0352°$$

Appendix C-3 Dynamic recalibration by rotary encoders

In our design we start by knowing the absolute position of the mount and we calculate the following positions relative to this by using the rotary encoder. Since we start movement with a higher degree of error than what we calculate with the rotary encoders, every following position calculated will have the same high degree of error. To minimize this we strategically use the one absolute position, triggered by pin Z, of the encoders to minimize our error. According to the gear ratio for movement, the pan motor will pass this position 8 times and the tilt will pass this position 5 times. Whenever we pass this position the encoder uses this algorithm to calculate our position as:-

$$if(current\ position\ \%1024 > 512)$$

$$encoder\ calculated\ \ position = 1024 * int(current\ position/1024)$$

$$else$$

$$encoder\ calculated\ \ position = 1024 * int(current\ position/1024 + 1)$$

Note:Current position is with respect to encoder and not in degrees
This means that the calculated position when the Z pin is triggered will always give us a fixed position as the closest multiple of 1024. To convert this into degrees we simply multiply our result by 360 and divide by the gear ratio used. This method works because our initial error is less than 22.5°

Appendix D-Wireless communication

Appendix D-1 Selection of Bluetooth over Wifi

There are many methods of wireless communication, the two most commonly being WiFi and Bluetooth. Although it is possible to use both simultaneously, we have decided that the best implementation based on time constraints will be through Bluetooth. Bluetooth communication is easy to implement and has a sufficient range and signal strength for communication. WiFi requires the mount and Ipad to be connected to the same network, whereas Bluetooth directly connects one device to another. The disadvantage of Bluetooth is that an external bluetooth module is relatively more expensive, costing approximately $35, than a WiFi module, which costs approximately $8. Despite this bluetooth communication protocols are easier to program.

Initially we tried using an nRF8001, which uses SPI(Serial Peripheral Interconnect), which is a synchronous communication protocol(based on the clock of a microcontroller) which can simultaneously send and receive data, to communicate with the microcontroller. This module was a UART(Universal Asynchronous Receiver and Transmitter), which sends and receives data which is not based on a clock, to communicate with other bluetooth devices. When we replaced this with the inbuilt bluetooth module of the Arduino 101 we did not have to change the way the data was sent as it was functionally the same.

Appendix D-2 Data stream convention

We use the following schemes in our design while sending and receiving signals. After establishing a connection with the iPad the arduino must wait for data to be sent to appropriately control the movement of the motors. Similarly when we send data back from the arduino to the iPad, we have a scheme for how the iPad must interpret the return feed of data. Below, the currently used data cheme for receiving data is shown. From the arduino can be seen. Refer Appendix D-3 for how to reassemble multiple bytes for larger data values. In general when the iPad sends any data it waits for an acknowledge and a completed signal from the arduino to ensure that the system is working.

Data Sent from iPad	Interpretation by Arduino
1	Wait for two bytes of data to select new pan position
2	Wait for two bytes of data to select new tilt position
11	Start continuous rotation of pan in clockwise direction
12	Start continuous rotation of pan in counter-clockwise direction

19	Stop Pan rotation
21	Start continuous rotation of tilt in clockwise direction
22	Start continuous rotation of tilt in counter-clockwise direction
29	Stop Tilt Rotation
50-58	Saved preset movements
59	Stop movement

Table 6.3 Interpretation of data sent by iPad

Data Sent from Arduino	Interpretation by iPad
0	Ignore/ waiting for next instruction
1	Received data
2	Done movement
10	Error-Unreachable states.

Table 6.4 Interpretation of data sent by arduino

Appendix D -3 8 Bit data Stream Transmission

The bluetooth libraries on the Arduino an iPad allow us to receive and send data as 8 bit(1 byte) data streams. This means we can send any number from -128-127 between the iPad and Arduino. For the simplicity of calculations we have decided to deal only with positive integers, leaving us with a range of 0-127. Since we are dealing with positive integers, we first internally convert our desired values of position from -180 - 180 to 0-360 by adding 180. Similarly we convert -90-45 to 0-135. To send positions with a precision of 0.1, we multiply the desired angles by 10. This means the iPad interprets the positions as 0-3600. One byte of data is not sufficient to send such large numbers.

We have calculated that two bytes of data will allow the sending of any number up to 128*128-1, which is 16,383. This is sufficient to indicate the position of the pan and tilt mechanisms. To convert a number smaller than 16,383 to 2 bytes of 8 bits we perform the below operations.

$$Data\ byte\ A = Input\ data \% 128$$

$$Data\ byte\ B = Input\ data / 128$$

When this data is received by the Arduino, the value can be reconstructed by simply reversing these operations.

$$Read\ Data = Data\ byte\ B * 128 + Data\ byte\ A$$

Appendix E Software- Arduino

The source code for the arduino files can be found at
https://github.com/Smarangk13/ELEC491Capstone2016/tree/master/Arduino

Appendix E-1 Arduino software Flowchart

The Arduino uses has to handle the bluetooth communication as well as motor control simultaneously. After startup, the Arduino will constantly check the position of the mount as well as be ready to receive data over bluetooth. By running its timer to move the motor and its hardware interrupts to check the position it can move to a new position as soon as it is notified of a new movement.

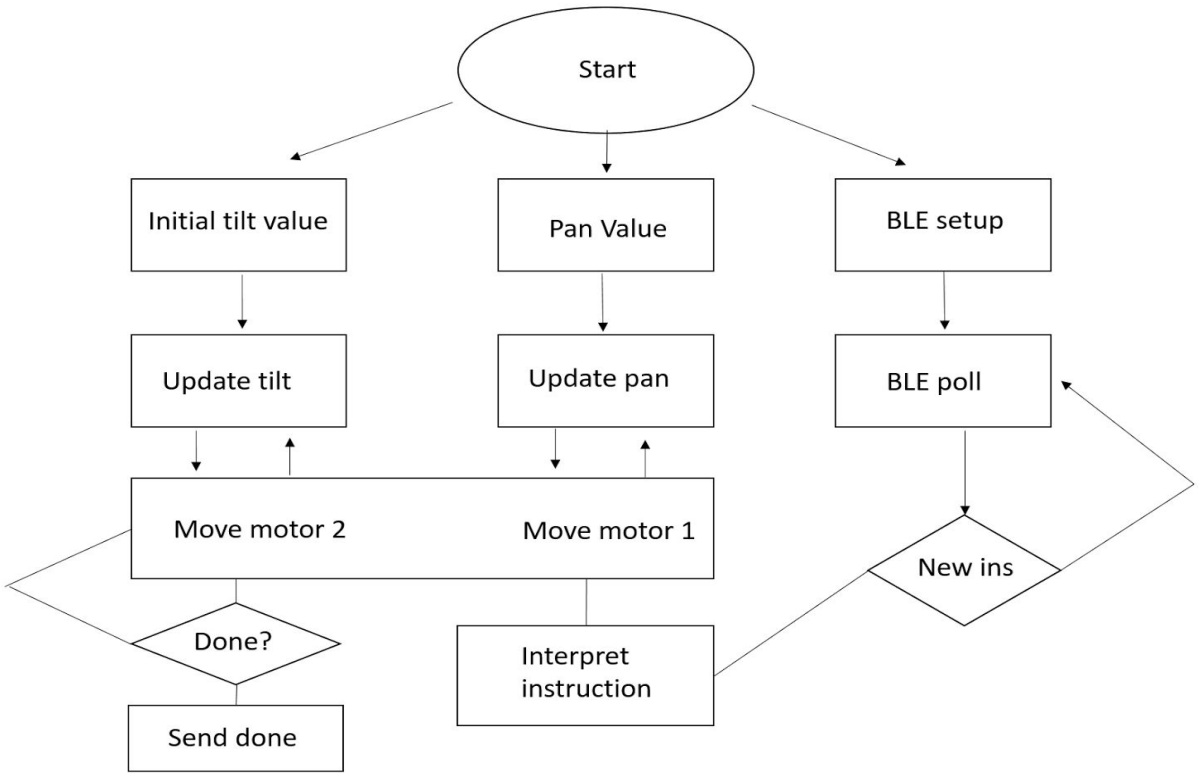

Figure 6.4 Arduino Flowchart

Appendix F Software- IOS

The iOS code can be found at-
https://github.com/Smarangk13/ELEC491Capstone2016

Appendix F-1 Flowchart

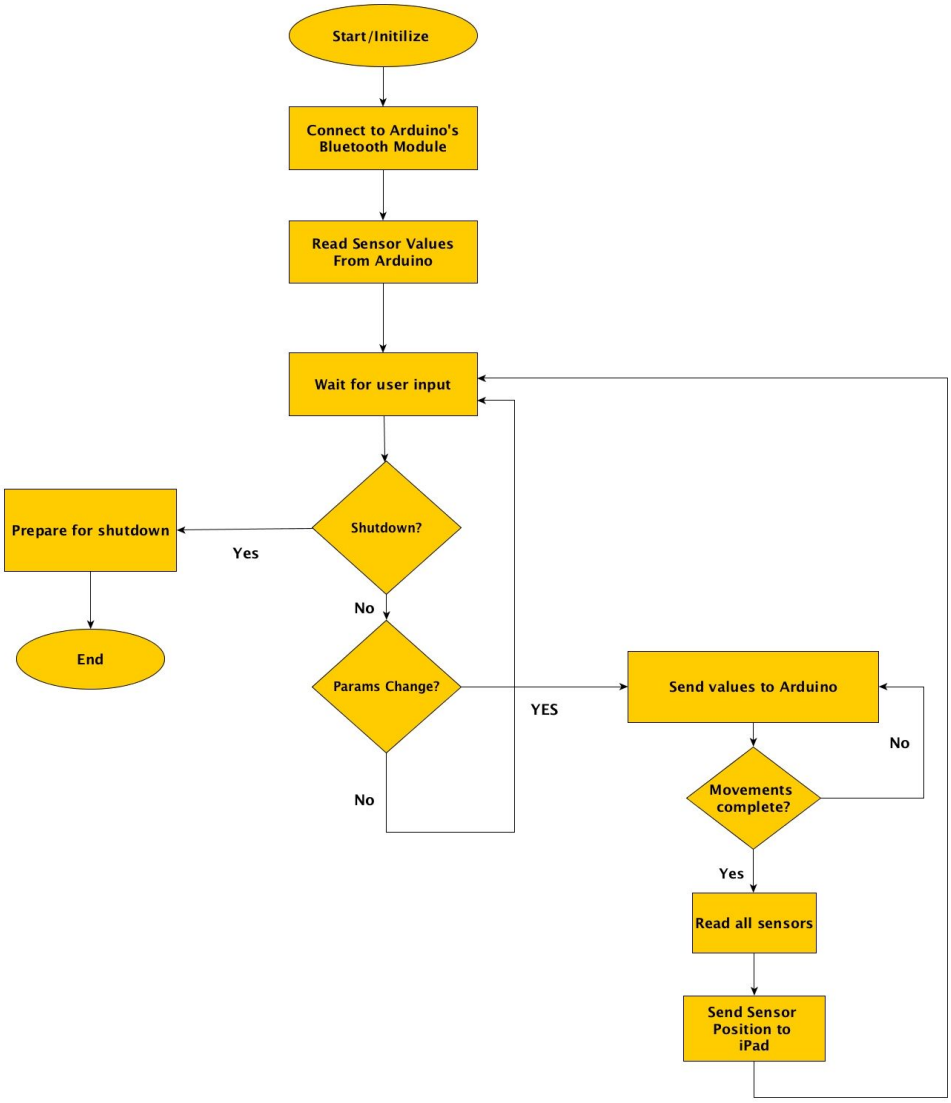

Figure 6.3-System Control Flowchart

Appendix F-2 Bluetooth Peripheral Explained

A peripheral can be understood as a device that the central -usually the controlling device like iPad or iPhone- is attempting to connect to. Each peripheral can have multiple services and within each service lies one or more characteristic. The characteristics are then defined as the node in which we either can write data to or read data from. To better understand this concept and terminology, refer to the documentation of the CurieBLE Library [10].

Appendix F.4 Home Page of iOS Application

Figure 6.4- Protoboard Circuit Diagramff

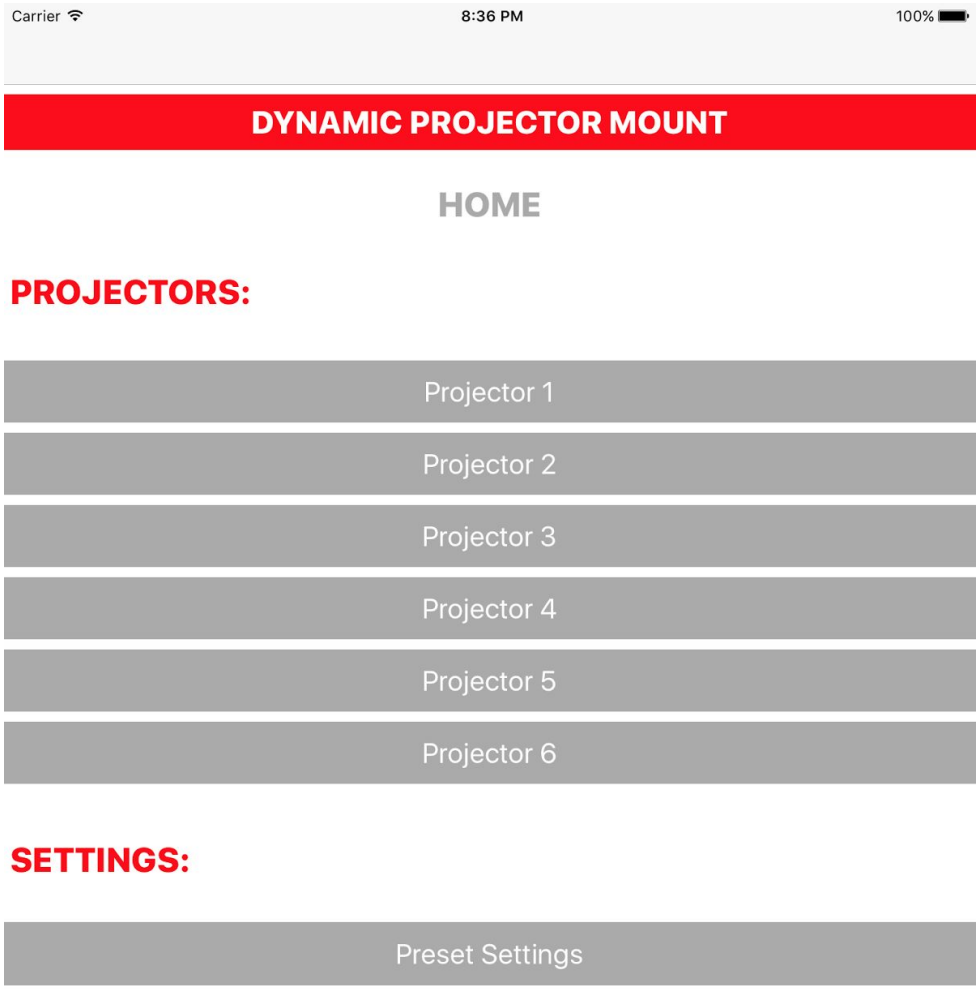

Figure 6.4 App Startup Page

Appendix G- Circuit Diagram:

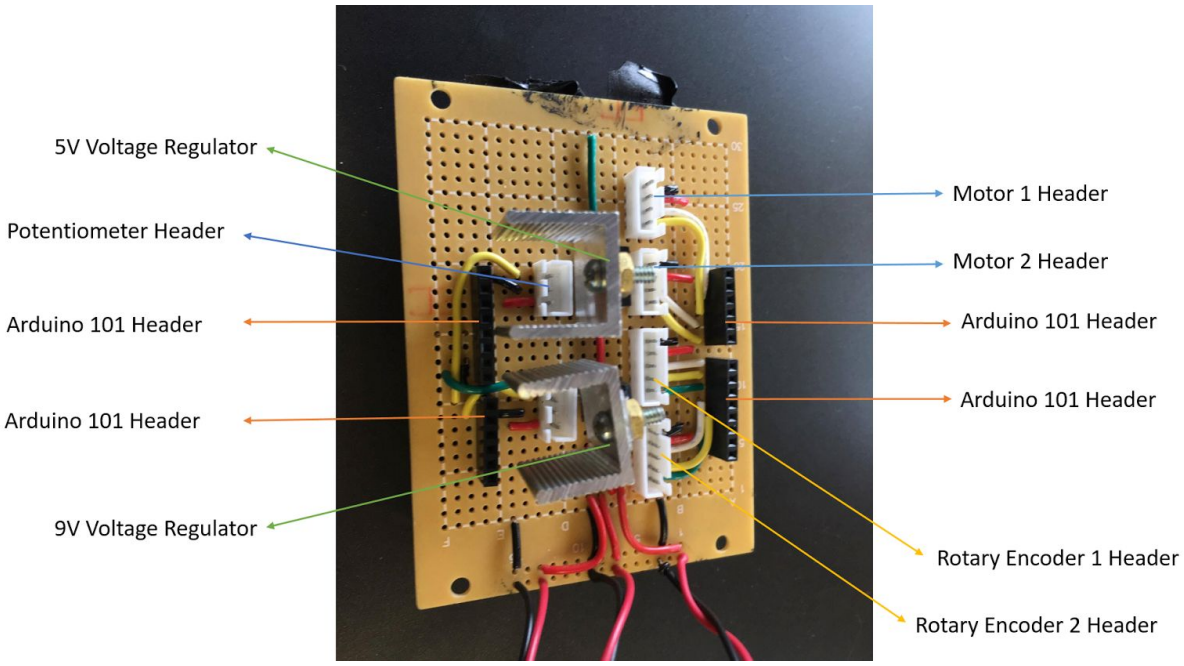

Figure 6.4- Protoboard Circuit Diagram

5V Voltage Regulator converts 19V to 5V which is used by all sensors and drivers
9V Voltage Regulator converts 19V to 9V to power the Arduino 101.
Motor drivers have 4 pins headers, 2 for motor control connecting to Arduino 101 Headers and 2 for 5V and GND.
Rotary Encoder has 5 pins header, 3 for digital feedback connecting to Arduino 101 Headers and 2 for 5V and GND.
Potentiometer has 3 pins header, 1 for analog feedback connection to Arduino 101 Headers and 2 for 5V and GND.

Appendix H Relative Comparison of Structural Metals

	Aluminum	Steel	Titanium
Cost	Low	Low	High
Availability	High	High	Medium
Machinability	High	Medium	Low
Strength	Low	Medium	High
Weight	Light	Heavy	Medium

Relative Comparison of Structural Metals

Importance

- Low cost is very important because of Constraint 2 in the Requirements Document
- High availability is prefered, although not necessary, to more easily satisfy Constraint 3 and Goal 4
- High machinability means it is easy to fabricate parts out of this material, which relates to the speed at which parts can be fabricated as well as the the cost to fabricate the parts and is very important because of Non-Functional Requirements 7, Constraint 2, and 3
- High strength is somewhat important because of Non-Functional Requirement 2, all three structural metals can be designed to carry the projector payload
- Light weight is somewhat important because of Constraint 1, 400 kg is more than enough to design using any of the three structural metals

Appendix I Relative Comparison of Structural Metals

	Spur Gear	Belt and Pulley	Helical/Double Helical Gear	Worm Gear
Cost to Implement	Low	Medium	High	High
Practical Gear Ratio	Low	Medium	Low	High
Availability	High	High	Low	Low
Back-Drivable	Yes	Yes	Yes	No
Transmission Efficiency	Medium	High	High	Low
Precision	Low	Medium	Medium	High
Accuracy	Medium	High	High	Medium
Noise	High	None	Medium	Low
Maintenance	Medium	Low	Medium	High

Relative Comparison of Structural Metals

Importance

- Low cost is very important because of Constraint 2 in the Requirements Document
- High gear ratio results in higher torque and is prefered, although not necessary, to achieve Non-Functional 2, and 3 in the Requirements Document to more easily satisfy the requirements, Goal 1 is also more easily satisfied the higher the gear ratio
- High availability is prefered, although not necessary, to more easily satisfy Constraint 3 and Goal 4
- Not back-drivable is always prefered for any positioning or motion application because it means that the transmission mechanism is only one way, where the motor can move the mount, but it is not possible to move the mount without using the motor (i.e. high winds, bumping into it, or unbalanced moment)
- Transmission efficiency is not important and not required in this application and for this project, however having high efficiency is always prefered over low efficiency
- High precision relates to how fine an adjustment can be made and is very important because of Non-Functional 3

- High accuracy relates to how smooth an adjustment can be made and is very important because of Non-Functional 3 and Goal 1
- Noise is not important and not required in this application and for this project, however having low or no noise is always prefered over having a loud mechanism
- Low maintenance is very important because of Functional 7 in the Requirements Document

References

[1] Stepper motor Driver -https://www.pololu.com/product/1182
[2] Potentiometer-http://www.mouser.com/ds/2/54/590-776468.pdf
[3] Rotary Encoder- http://www.mouser.com/ds/2/307/e6b2-c_ds_csm491-25665.pdf
[4] Bluetooth Versions- https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions
[5] Stepper motor-
https://jkongmotor.en.alibaba.com/product/60232212584-800892390/42mm_stepper_motor_JK42HS40_1204D_NEMA17_stepper_motor.html
[6] GT2 Tooth Profile - http://www.sdp-si.com/PDFS/Technical-Section-Timing.pdf
[7] 5 Voltage Regulator LM7805 - https://www.sparkfun.com/datasheets/Components/LM7805.pdf
[8] 9 Voltage Regulator LM7809 -
https://static1.squarespace.com/static/5416a926e4b09de8832655bc/t/54427037e4b03de3b67b895a/1413640247188/lm7809.pdf
[9] Arduino 101 - https://www.arduino.cc/en/Main/ArduinoBoard101
[10] Curie BLE https://www.arduino.cc/en/Reference/CurieBLE