Rotating Video Mount for Ford Motor Company: Final Report

Mark Cuson (mcuson@stanford.edu)
Yuto Watanabe (yutow@stanford.edu)
Will Dannemann (wjdann@stanford.edu)
Daniel Espinel (desipnel@stanford.edu)
ME 113 Spring 2014

Abstract

We built an auto-rotating video mount, the SwivelShot, for the Ford Motor Company, with the primary objective being to create a hardware platform to dynamically capture the driving experience. This hardware integrates into Ford's existing open source project, OpenXC. The mount can rotate 360^{o} automatically using preset tracking patterns, or custom tracking patterns based on car data. There is also potential for a software or hardware based controller via bluetooth. Additionally, the mount accommodates multiple attachment mechanisms, including suction cups and clamps, as well as multiple recording devices, like smartphones and GoPros. Our final design demonstrates functional mechanical, electrical, and control systems, whose critical elements are an Arduino, a magnetic encoder, and a DC motor with a gearbox and one-stage transmission. The project will now be handed off to Ford for further development as they see fit.

Table of Contents

Introduction

Ford Motor Company challenged us to create a dynamic video mount for capturing the driving experience. Companies like GoPro currently have very good products for capturing static video in extreme sport situations, including driving, and the Galileo video mount is a great example of a dynamic video mount platform. But there has not been anything specifically developed for use with motor vehicles. Furthermore, a solution is needed to bridge the large gap between amateur and professional car videographers and allow the average hobbyist to showcase their driving experiences in an unique and dynamic manner.

This project has culminated in an open source mount designed for the interior of cars that is capable of rotating 360^{o} and is compatible with the OpenXC software platform that Ford has created. By open sourcing the project, other inventors will be able to take the work that is done on this project and extend it, using information that the OpenXC platform outputs to create more interesting things with the software. Both hardware and software designs and instructions for replication will be released for everything that is done.

Hypothetical Use Cases

User Scenario 1: The user is a muscle car enthusiast and amateur adventure driver. He has a new Ford Mustang, and is taking it out to a racetrack for an afternoon. He has no recording equipment besides a GoPro camera and the Ford Auto-Rotating Video Mount, but wants to get some awesome footage to bring back home to show his friends. Beforehand, he uploads code to the mount so that it turns with the steering wheel, anticipating the curves so as to make for more engaging and interesting footage. He clamps the mount to the passenger seat headrest, to give the perspective as if you were in the car with him.

Figure 1: Vehicle at the track

Figure 2: Video mount on headrest

User Scenario 2: The user is taking a weeklong road trip with a couple of friends along the Pacific Coast Highway, and wants to document the adventure in a simple but interesting way. From beautiful sandy beaches to redwood forests to the streets of San Francisco, she wants to dynamically record the passing scenery, while also capturing the interior of the car, where the real memories will be made. She attaches her iPhone to the Ford Auto-Rotating Video Mount using a case with a tripod attachment, and suctions the entire apparatus to the windshield. Her friend in the backseat manually controls the mount during particularly exciting portions of the drive, or sets it to a default sweep between the interior and exterior of the vehicle.

Figure 3: In car view of mount

Figure 4: Road trip!

Design Features

- A remotely controllable camera mount for a GoPro for use in the interior of cars with a 360^{o} range of movement at a max speed of 180^{o}/second
- Preset modes for the mount to allow for changing views without direct input while driving
- Ability to utilize different recording devices (GoPro, iPhone, traditional camera, etc) through universal tripod mount
- Ability to affix the camera mount to multiple locations on the interior or exterior of the vehicle by using different mounting hardware that is readily available on the market
- Wireless connection to the output from Ford's OpenXC platform for collecting information from the car
- Full documentation so that the project can be replicated by hobbyists or continued by Ford
- Bluetooth connectivity from controller to mount to eliminate the need for a hardwire connection
- Communication from mount to an Android smartphone for more precise control

Factors Considered in Design

Given the open-ended nature of our assignment, it was important to refine our design objectives to consider a more limited amount of factors. These factors can be divided into three categories: human factors, integration with Ford specific products, and extensibility.

Human Factors: As a customer facing product, it was extremely important to make a device that is easy to use and appealing to put in a car. By definition, the demographic that is being targeted with a video camera mount such as ours has an interest in portraying their vehicle and driving experiences in the best possible light. This requires a mount that is designed with aesthetic appeal in mind. Part of that aesthetic is that it would be required to look natural in the car environment with an emphasis being on the camera used and not on the mount itself. Effectively, the design would have to be both subtle and attractive. It is also important that the user interface for the device is intuitive and responsive to ensure ease of use.

Integration with Ford: Since this is a product being built specifically for Ford, it was very important to design around Ford products. This manifested itself in two ways: the physical form factor and the integration with the OpenXC software platform. The primary physical constraint was the distance between the dashboard and the windshield of the 2011 Ford Mustang that we used for testing purposes.

For OpenXC integration, it was important to consider that people who do not have the same technical familiarity with the mount as our team may still want to hack it. This requires a relatively simple interface for getting car data. This interface was built upon Ford's existing research in the area in order to ensure that the communication protocol is stable.

Figure 5: Dashboard of 2011 Ford Mustang

Extensibility: In order to make both the hardware and software portions of the project Open Source, it was important that we make the product as easy to customize and augment as possible. To achieve this end, we provided ample documentation of the building process, and made the design as simple and easy to understand as possible. This allows hobbyists to easily replicate the design, and eventually expand upon it.

Exploration of Mechanical Design

Frame Evolution: The initial design was a rough prototype (V1) done in wood. The V1 served to demonstrate the feasibility of the overall design and verify that the selected motor had the required torque to rotate the camera at a reasonable speed.

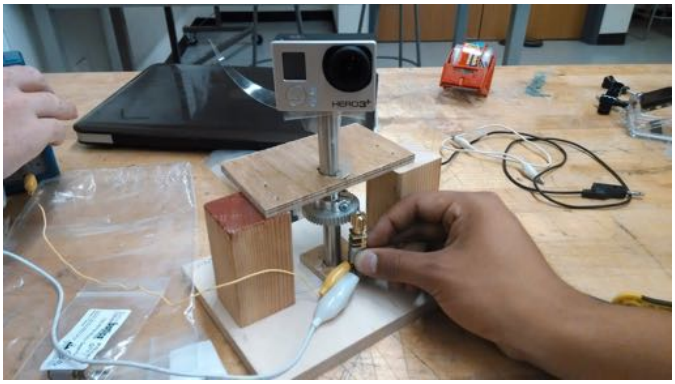


Figure 6: V1 prototype

The next design (V2) resulted in a large increase in resolution of the overall skeleton of the mount. The V2 frame consists of four laser cut pieces of duron held together with bolts and spacers, with appropriate geometries to align the mechanical components while allowing easy access to electrical testing and verification.

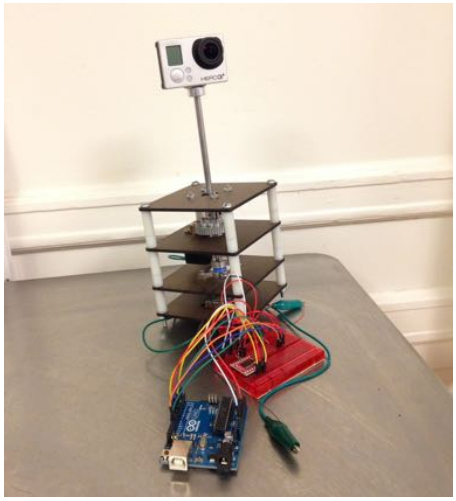


Figure 7: V2 prototype

After V2 demonstrated mechanical and electrical functionality, the primary goal for the next frame design was to compact the mechanical components into a usable size. The result was V3, which consists of three 3-D printed parts assembled into a box. The fundamental concept of layers is the same as in V2, but the layout is much more space efficient. Spacers are built into each layer, and the top layer is a full enclosure, protecting internal components from the environment. However, this iteration does not include space for the electrical components, as seen in Figure 8. An additional mounting hole was added on the side, so as to reduce the effective vertical height of the mount when connected to the suction cup. This mounting feature was also kept in our final product, as will be seen later. Before moving on to our next design, we verified the functionality of the V3 mount by testing the prototype as shown in Figure 9.

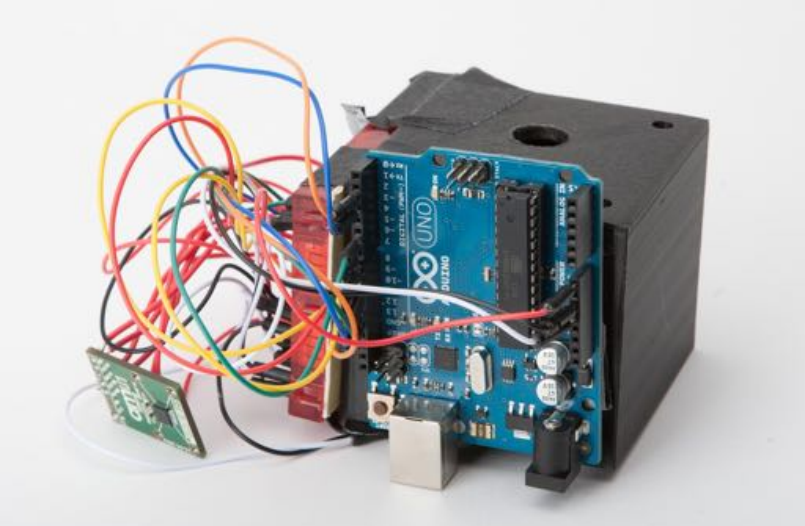


Figure 8: V3 prototype

Figure 9: Testing V3 prototype

The clear next step for V4 was to incorporate the electrical parts into the housing. As will be discussed in further detail in the electrical design section, the final boards were quite small and unobtrusive. The driving factor in the design of V4 was in fact the battery, which is oriented vertically so that it intersects the middle layer as seen in Figure 10b. The overall box had to be widened slightly for two reasons: so that the battery would not interfere with the gears contained in the upper compartment of the housing and so that the shaft remained symmetrical in at least one dimension. Otherwise, the design remained essentially unchanged from V3.

Figure 10a: V4 prototype

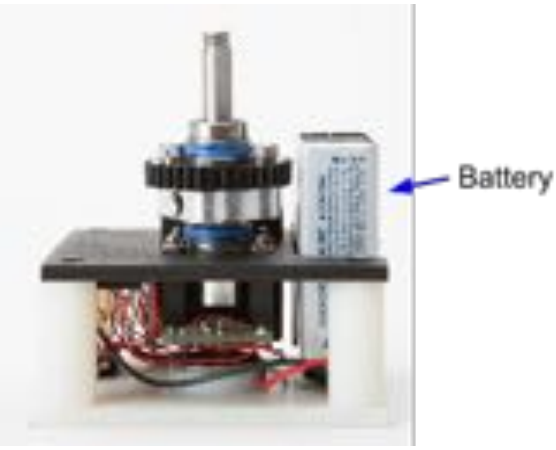


Figure 10b: Battery positioning

For our final version, we decided to source a high quality printed housing from a specialized company recommended by Ford. Aside from adding a slot for a power switch and integrating some internal support structures, the design of V5 is identical to V4. Engineering drawings for the V5 can be found in the Appendix A.

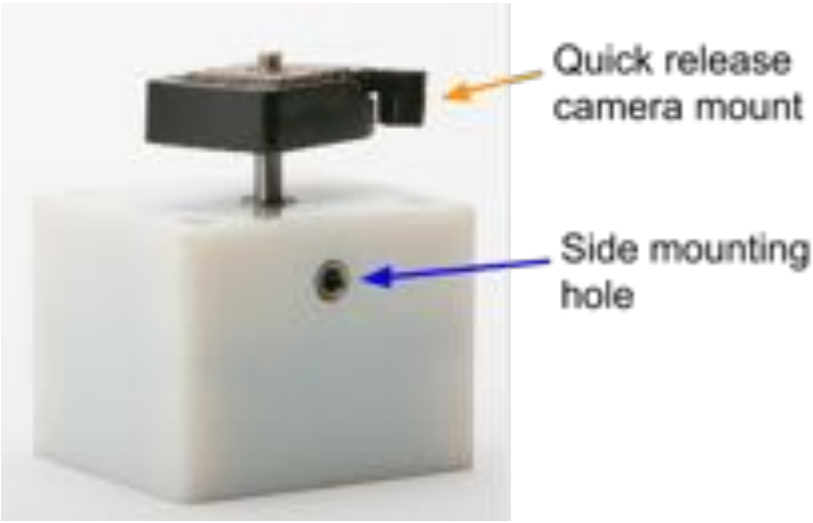


Figure 11: Final product

Figure 12: Final product with GoPro and suction cup attached

Motor Exploration: There are many motor options available on the market. Before deciding upon our solution, several of these options were considered. Our motor has two major requirements: the ability to move smoothly between locations, and the ability to determine the angle that the camera shaft is currently positioned. Both functions would not necessarily have to come from the same device.

Motors	Pros	Cons
Stepper	Ability to move quantized distances accurately	No ability to sense current location Limited resolution (generally around 2°) Needs to be zeroed before use Can have somewhat jerky movement
Servo	Can specify absolute position Smooth rotation	Unable to rotate over 360° while maintaining location awareness Can be somewhat loud
DC (brushed)	Very smooth rotation	No location awareness, relative or absolute

Ultimately, a two device solution was settled upon. To close the control loop of the motor, some kind of feedback would be needed. A servo has such a control but can only rotate in a limited range. As a result, a magnetic encoder was settled upon to do shaft angle measurement. By precisely mounting the encoder chip as shown in Figure 13a, the angular position can be measured due to the Hall effect[1]. Once the current angle is known, the Arduino inputs a desired angle, and based on the difference between the two angles, a speed and direction are set. Once the measured position is within 1° of the set angle, rotation stops, and the motor will hold that position even if disturbed. By combining the DC motor with an external encoder, a closed loop system is formed that uses absolute positioning to eliminate the need for calibration, allows for

[1] The Hall effect is the measured voltage change across a conductor due to the presence of a magnetic field perpendicular to a conductor. The voltage changes with the angle of the magnetic field relative to the conductor

unlimited rotation, and gives an incredibly simple programing interface for setting camera positions.

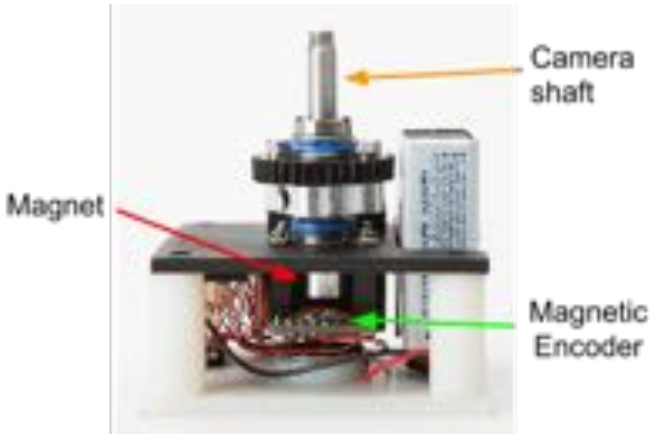


Figure 13a: Encoder mechanism

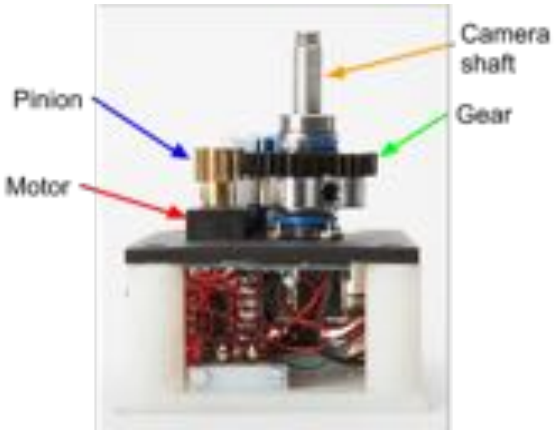


Figure 13b: Transmission mechanism

Using stepper motors was attempted, as seen in Figure 14. However, the results were less than satisfactory. Stepper motors were promising due to their ability to position themselves with an input signal. However, the motor's lack of torque and complete inability to operate or hold a position when the axis of rotation was tilted, removed the motor from consideration. The motor's inability to reliably hold an angle would have necessitated constant re-zeroing of the system or some other form of positioning management which would have come with all of the disadvantages of the DC motor and magnetic encoder system while not providing any real benefits.

Figure 14: Stepper motor testing

Exploration of Electrical Design

The prototype used an Arduino Uno as a microcontroller. The Uno controls two integrated circuits, one to control the motor and one to obtain information from the magnetic encoder over the SPI (Serial Peripheral Interface) bus. The motor driver used is the Toshiba TB6612FNG on a breakout board from Sparkfun Electronics to facilitate prototyping. The magnetic encoder is the ams AS5048A. The B variant uses the I2C (Inter-Integrated Circuit) bus and would also work but since only one peripheral is being used the SPI bus is simpler to set up in the opinion of the team. The AS5048A is also on a breakout board provided by the manufacturer that comes with mounting holes to allow for precise mounting to get the magnet into precise position above the chip.

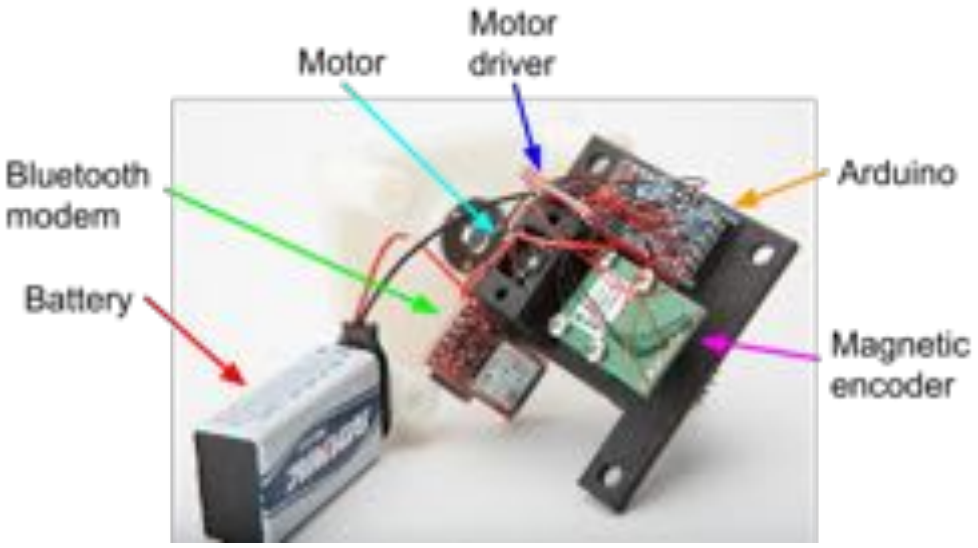


Figure 15: Layout of electronics

The fundamental electrical design did not change substantially between the V2 prototype and the final version of the electronics. The V3 prototype used the exact same wiring. The main differences resulted from the need to consolidate all of the electronics into a much smaller

space while remaining reliable. This was achieved by pursuing several different approaches. The first was the substituting in the 5V Arduino Pro Mini for the Arduino Uno. The Pro Mini has approximately 1/5th the footprint of the Uno and is substantially shorter as well. However, the Pro Mini is also much harder to use for prototyping since all wire connections need to be made with soldering. There is also no on board programmer, so an external programmer needs to be used for uploading code. Lastly, there is no direct plug for a power supply, so the barrel jack had to be cut open and have the raw leads power the entire rig.

Much smaller wires were also used to connect the components in the final version of the electronics, moving from 20 gauge to 30 gauge. This did not change the design but did make the attachment process much more difficult and less robust. To counteract this loss in robustness, the wires were prestressed into their final positions before they were soldered into place. This technique was discovered after having several wires break during assembly and helped maintain solid connections despite large vibrations and impacts.

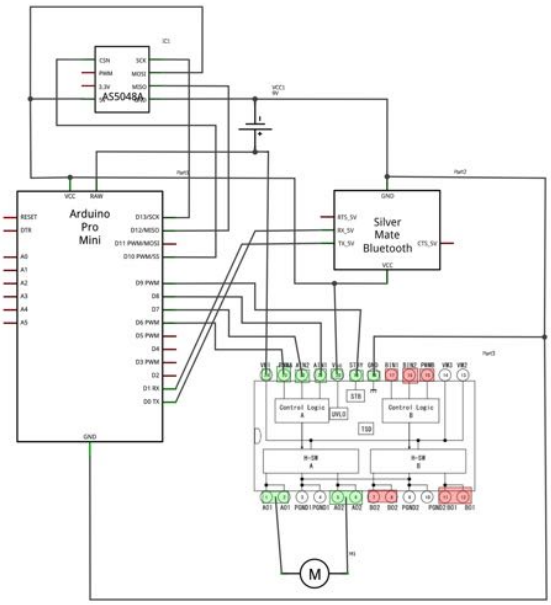


Figure 16: Electrical schematic

Lastly, in an effort to make the final design wireless, a bluetooth module was incorporated to connect the Arduino to an Android phone. This ended up taking a small amount of space in the final product but allowed for the complete elimination of wires coming out of the mount. The electrical schematic used in this design can be seen in Figure 16.

To facilitate the transition from V4 to V5, the electronics were designed as a unit. All of the electronics were designed to fit onto the middle platform of the final mount so that it could be swapped between units with matching enclosures easily. The V5 mount uses almost exactly the same electronics as the V4 mount with the addition of a power switch to eliminate the need to disassemble the mount to turn it on and off.

Software Development

The Arduino ran several different software versions depending on the desired functionality. The full code for the final control loop can be seen in Appendix C. All of the various options allow for input in different ways, but all of them use a simplistic proportional control law combined with the feedback from the magnetic encoder to position the DC motor appropriately.

The Arduino is also fully enabled to use Bluetooth communications. This decision was motivated by the need to connect the mount to OpenXC. Connecting the mount to the information obtained from OpenXC was only possible by using an Android as an intermediary relay and connecting via Bluetooth. This makes future module additions, such as a wireless controller or a custom Android app to expand functionality beyond just OpenXC integration, relatively simple without having to completely redesign the electronics. The motor control scheme is already set so adding additional functionality is simply a matter of changing the inputs to the control loop.

The software currently only supports steering wheel angle as an input. However, it is relatively simple to update the program to take other parameters from the car or even a combination of different inputs. Valid car parameters include accelerator position, car velocity, gear, fuel level, engine RPM, longitude and latitude, and engine torque.

Final Product

The final design, that we are tentatively calling the SwivelShot, is an elegant and effective solution to the problem that was presented, but still has massive potential for enhanced functionality. It can be made from scratch relatively easily with standard hobbyist skills and equipment, it is fully integrated into Ford's OpenXC platform, and is able to capture dynamic video of driving as intended. The mount as is an effective tool for capturing video but the real strength of the design is that it can be expanded as needed by tinkerers for specific applications. For a full bill of materials, see Appendix B.

The mount can be placed in numerous locations in the car, including but not limited to the front windshield, the headrest via a special headrest mount, and the exterior of the car. Thanks to the universal camera-mount inspired threading of both the shaft and the mount itself, adding additional mounting fixture locations is simply a matter of acquiring an appropriate mounting device. This allows for maximum flexibility by leveraging already existing solutions.

This same principle also applies to recording devices. By utilizing a standard fixturing method, numerous video recording devices can be used as seen in Figures 17. While the mount was designed with GoPro cameras in mind, with appropriate attachments any recording device could be used, including smart phones and DSLR cameras. This further increases the flexibility of the mount.

Figure 17a: Mounted with GoPro

Figure 17b: Mounted with Smartphone

The SwivelShot was also designed for ease of use, although it is still evolving along with OpenXC and SignalStream. To start using the mount, all that is required in to flip the power switch and pair the mount with an Android device running SignalStream. It will then begin moving along with the steering wheel angle.

Testing: We tested V4 in Ford's in-house Mustang, as seen in Figures 18. We verified several different mounting locations, including the exterior of the car.

Figure 18a: Interior mounting

Figure 18b: Exterior mounting

Figure 18c: Headrest mounting

We took video footage from the windshield position, using a GoPro and the suction cup mounted to the side hole on the SwivelShot. After driving around downtown Palo Alto for a few minutes, the mount proved to be fully functional except for a few minor control system glitches. These issues are most likely due to problems with the data transfer frequency between the Android tablet used and the mount. Additionally, the sound of the DC motor in the raw video footage was very prominent, and blocked out any other sound coming from the interior of the car. Although this issue did not strictly interfere with the performance of the SwivelShot, it detracts from the overall user experience.

Figure 19a: Front facing footage

Figure 19b: Capturing the car interior

Future Steps

Controller: The final mount could only take inputs either from a computer or from OpenXC, which made the mount somewhat difficult to use for general purpose applications. To address this issue, we implemented a version of the hardware that used an optical encoder knob (Figure 20) to control the position of the mount without having to rely on other software inputs. The next step would be to integrate this into a fully fledged remote control. Figure 21a illustrates a potential hardware based controller. Alternatively, we could write a smartphone application (Figure 21b) to achieve the same functionality.

Figure 20: Optical encoder controlled mount

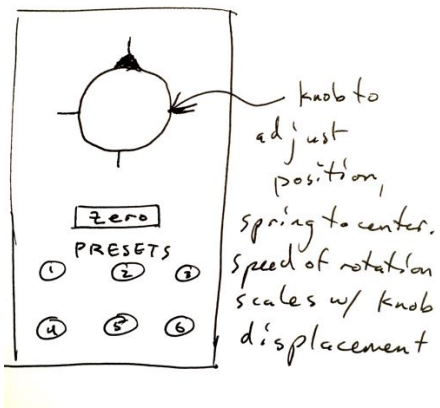


Figure 21a: Mechanical controller concept

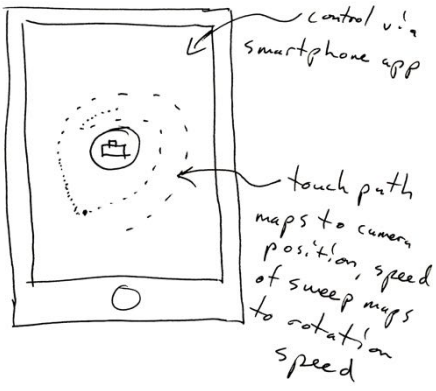


Figure 21b: Smartphone controller concept

Noise management: As noted in the V4 testing, the GoPro camera picked up a substantial amount of motor noise. There are several possible explanations for this, including an amplification of noise due to the camera's casing as well as the proximity of the camera to the mount's motor. It would be useful to do a full study in order to isolate the source of the noise and then make efforts to minimize the noise that the camera picks up.

Electronics refinement: The electronics are currently fully functional, although the wiring used to connect the various components could be improved. To achieve this, it would be desirable to get a custom printed circuit board made so that proper connections would be guaranteed while having the added benefit of substantially decreasing the size of the electronic system. The size of the electronics could be reduced to at least half the current size by creating a board specific to the application.

Casing manufacturing: As noted before, the current mount housing was 3D printed, but to increase quality and decrease cost, we would manufacture the next version of the housing using injection molding. This would require a substantial upfront investment for tooling but would reduce the marginal cost of producing the casing by at least 95%, thereby eliminating the largest line item in the bill of materials.

Reducing overall size: By further refining the design, we could greatly reduce the overall size of the mount. This has numerous benefits including lower cost of manufacturing, improved aesthetics, and increased mounting versatility. This process would be ongoing throughout the lifetime of the product since a primary goal is for it to be as compact as possible.

Implementing more sophisticated software: There are numerous potential applications for this device that could be realized through software. This could include giving a first person perspective in real time for a remote viewer by using a smartphone with a data link which would have use for engaging remote viewers with the driving experience. Race car drivers looking to

get real time feedback from coaches without having another person physically in the car could also utilize such a device. Such a program could control the angle and relay the video to a remote location. The mount could then be used as not only a recording device but as an interactive one as well.

Conclusion

Our team was quite successful at accomplishing the goals we were challenged to accomplish. We created a sophisticated video mount and now that our phase of the project is over we look forward to seeing where Ford will go with the work that we have put in.

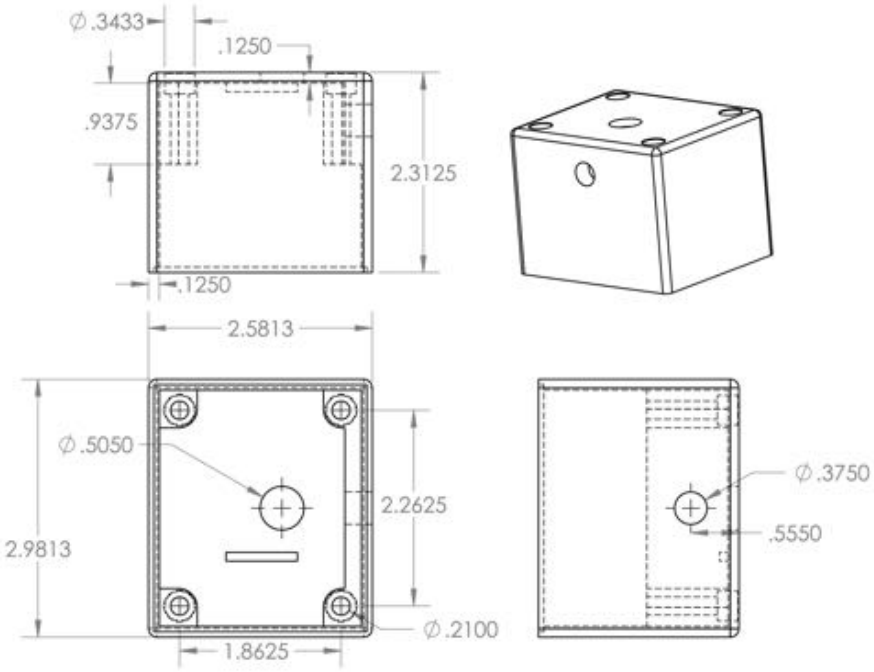
Acknowledgements

- The 113 Teaching staff, especially our coaches John Howard and Joaquin Carcache.
- Our corporate partners at the Ford Silicon Valley Lab: Sudipto Aich, Jamel Seagraves, and Chih-Wei Tang.

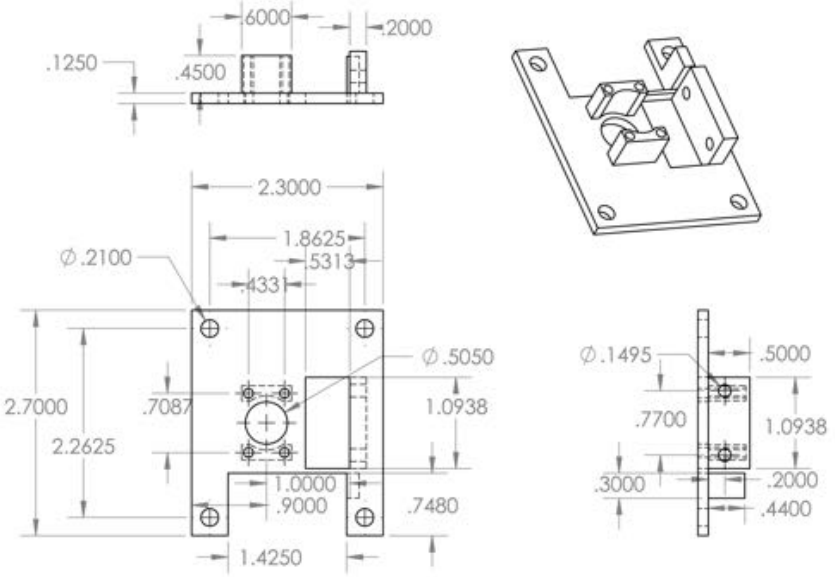
Appendix

Appendix A: Engineering Drawing of Housing

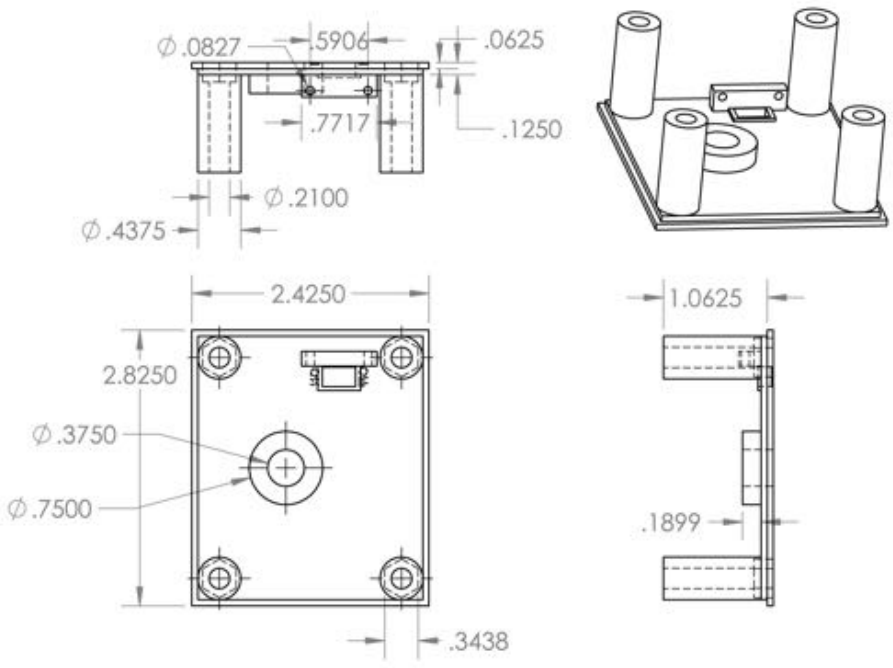
Enclosure



Middle Platform



Bottom Platform



Appendix B: Bill of Materials

Item	Supplier	Supplier Part Number	Quantity	Price ea.	Total Price
SPDT Panel Slide Switch SS12F55	Amazon	B007Q854MS	1	0.55	0.55
Slik DQ-10 Quick Release Adapter Small	B&H Photo	SLQRAS	1	18.95	18.95
AS5048 14-bit Magnetic Rotary Encoder IC	Digi-Key	AS5048A-AB-1.0-ND	1	19.39	19.39
3D printed VeroWhite enclosure	Fictiv		1	258	258
3D printed VeroWhite middle platform	Fictiv		1	47	47
3D printed VeroWhite bottom platform	Fictiv		1	85	85
Type 316 Stainless Steel Socket Head Cap Screw, 10-24 Thread, 2-1/2" Length	McMaster	92185A257	4	1.13	4.52
18-8 Stainless Steel Undersized Machine Screw Hex Nut, 10-24 Thread Size, 5/16" Width, 7/64" Height	McMaster	90730A011	4	0.06	0.22
Metric 18-8 Stainless Steel Thin Hex Nut, M2.5 Size, .45MM Pitch, 5MM Width, 1.6MM Height	McMaster	90710A025	4	0.05	0.18
Metric 316 Stainless Steel Pan Head Phillips Machine Screw, M2.5 Size, 20MM Length, .45MM Pitch, Packs of 100	McMaster	90116A124	4	0.12	0.46
18-8 Stainless Steel Metric Pan Head Phillips Machine Screw, M2 Size, 14mm Length, .4mm Pitch	McMaster	92000A020	2	0.05	0.09
Metric Type 316 Stainless Steel Thin Hex Nut, M2 Size, .4mm Pitch, 4mm Width, 1.2 mm Height	McMaster	93935A305	2	0.12	0.24
Type 316 Stainless Steel Socket Head Cap Screw, 6-32 Thread, 3/8" Length	McMaster	92185A147	4	0.01	0.39
270 RPM Micro Gearmotor Blocks	ServoCity	638122	1	9.99	9.99
Micro Gearmotor Enclosure	ServoCity	555120	1	0.99	0.99
32 Pitch (3mm Bore) Gearmotor Pinion Gear	ServoCity	615318	1	9.99	9.99
32 Pitch Acetal Hub Gear, 36 Tooth	ServoCity	RHA32-36-36	1	4.49	4.49
0.770" Set Screw Hub, 1/4" Bore	ServoCity	545548	1	4.99	4.99
Flanged Ball Bearings, 1/4" Bore	ServoCity	535044	2	1.69	3.38
Thrust Bearings	ServoCity	6655K13	2	4.4	8.8
1/4" Precision Shafting	ServoCity	634160	1	0.89	0.89
Aluminum Set Screw Collars, 1/4" Bore	ServoCity	9946K11	1	4.5	4.5
FTDI Basic Breakout - 5V	SparkFun	DEV-09716	1	14.95	14.95
Break Away Headers - Straight	SparkFun	PRT-00116	2	1.5	3
Motor Driver 1A Dual TB6612FNG	SparkFun	ROB-09457	1	14.95	14.95
9V to Barrel Jack Adapter	SparkFun	PRT-09518	1	2.95	2.95
9V Alkaline Battery	SparkFun	PRT-10218	1	1.95	1.95
Bluetooth Modem - BlueSMiRF Silver	SparkFun	WRL-12577	1	39.95	39.95
Arduino Pro Mini 328 - 5V/16MHz	SparkFun	DEV-11113	1	9.95	9.95
Wire Wrap Wire (30 AWG)	SparkFun	PRT-08029	1	8.95	8.95
Total					**579.68**

Appendix C: Prototype Arduino Code

```
/*
//
// closed_loop_basic.ino
//
// written by Will Dannemann for Ford SVL
// 6/7/2014
//
// This code is free to use and modify as desired.
//
// closed_loop_basic is the control program to be uploaded to an Arduino for controlling the video mount developed in ME 113, Spring 2014, for Ford.
// The system utilizes:
//                      - an Arduino Uno, or Pro Mini 5V, or equivalent
//                      - a DC motor
//                      - a TB6612FNG motor driver
//                      - the ams AS5048A magnetic encoder
//                      - a BlueSMiRF Silver for wireless communication
//
// The system uses the AS5048A to determine the positioning of the shaft thanks to a magnet mounted to the end.
// That position is then compared to a "goal" position that is input either by the user over Serial or automatically via software.
// The speed and direction of the motor is then set to get to that position. The speed decreases when the shaft is within a small distance
// of the goal point.
//
// It is worth noting that this is not a differential movement system but rather an absolute positioning control.
// It could be rewritten as a differential system, but doing so would require slight modifications to this source code.
//
// The software is designed to take input from SVL's SignalStream Android application to control the motor positioning.
*/

#include "SPI.h"
#include <SoftwareSerial.h>

int bluetoothTx = 2;
int bluetoothRx = 3;

bool inNumber;

//software serial is used to allow for bluetooth communication while not blocking the TX, RX lines required for
//programming
SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

int position = 0;
int MSB = 0;
int LSB = 0;

int goal = 0;

int time = 0;

int maxEncoderOutput = 16383; // the enconder can output 0-16383. Scale appropriately for degrees

//used for reading from Serial input
String readString;

//motor driver pins, only one of the two potential outputs are used
int STBY = 9;
int PWMA = 6;
int AIN1 = 8;
int AIN2 = 7;

void setup ()
{
```

```
  Serial.begin(9600);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  SPI.setDataMode(SPI_MODE1);

  pinMode(STBY, OUTPUT);
  pinMode(PWMA, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);

  //configure the bluetooth module
  bluetooth.begin(115200); //default 115200bps
  bluetooth.print("$");
  bluetooth.print("$");
  bluetooth.print("$");  // Enter command mode
  delay(100);  //wait for CMD to return
  bluetooth.println("U,9600,N");
  bluetooth.begin(9600);
}

void loop() {

  /*
  // This block will read angles from 1 to 360 and position the shaft at that angle

   while (Serial.available() > 0) {
      char c = Serial.read();
      readString += c;
      delay(2);
    }
  if (readString != "") {
    goal = readString.toInt() % 360;
    goal = goal * 44;
    readString = "";
  }
  */


  /*
  // This block will move the motor to random angles as long as the Arduino has power

  unsigned long milli = millis();
  Serial.print(milli);
  Serial.print(",");
  Serial.println(time);
  if (milli / 1000 > time) {
    time = milli / 1000;
    goal = random(0,maxEncoderOutput);
  }
  */

  // This block will get Steering Wheel Angle from Signal Stream and control the motor accordingly

  int wheelAngle = getWheelAngle();

  // turn Wheel Angle into an appropriate Goal value (0-16383)
  // this should be eventually modified to account for the initial angle of the wheel being 0 degrees
  // regardless of the actual angle

  if (wheelAngle != 0) {

    wheelAngle *= 20;

    if (wheelAngle < 0) {
      goal = (maxEncoderOutput + wheelAngle % (-1 * maxEncoderOutput));
    } else {
```

```
      goal = wheelAngle % maxEncoderOutput;
    }
  }

  // get data from encoder via SPI
  digitalWrite(SS,LOW);
  MSB = SPI.transfer(0b00000000);
  LSB = SPI.transfer(0b00000000);
  MSB &= 0b00111111;
  MSB = MSB << 8;

  position = MSB | LSB;
  digitalWrite(SS,HIGH);

  /*
  // debug
  Serial.print("goal: ");
  Serial.println(goal);
  Serial.print("position: ");
  Serial.println(position,DEC);
  */

  // lower speed when within ~ 10 degrees of goal to ensure that the motor doesn't overshoot

  int speed = abs(goal-position) > 500 ? 255:100;

  if (abs(goal-position) < 44 || (abs(goal-position + maxEncoderOutput) < 45)) {
    stop();
  } else {
    if (goal > position && (goal-position) < maxEncoderOutput/2) {
      move(speed,0);
    } else if (goal > position && (goal-position) > maxEncoderOutput/2) {
      move(speed,1);
    } else if (goal < position && (position-goal) < maxEncoderOutput/2) {
      move(speed,1);
    } else {
      move(speed,0);
    }
  }
}

//within goal, stop the motor
void stop(){
  analogWrite(PWMA,0);
}

void move(int speed, int direction){
//speed: 0 is off, and 255 is full speed
//direction: 0 clockwise, 1 counter-clockwise

  digitalWrite(STBY, HIGH); //disable standby

  boolean inPin1 = LOW;
  boolean inPin2 = HIGH;

  if(direction == 1){
    inPin1 = HIGH;
    inPin2 = LOW;
  }

  digitalWrite(AIN1, inPin1);
  digitalWrite(AIN2, inPin2);
  analogWrite(PWMA, speed);

}
```

```
// getWheelAngle will take the signals coming from SignalStream and parse them appropriately
// it has only been tested on steering wheel angle, though
// using the Arduino signalstream library could help make this process easier, but signalstream
// requires use of the Arduino TX and RX lines, making reprogramming the Arduino impossible without
// desoldering the bluetooth module each time
int getWheelAngle()
{
  inNumber = false;
  String steeringWheelAngle;
  while (bluetooth.available()) {
    char c = ((char)bluetooth.read());
    if (c == ':') {
      inNumber = true;
    }
    if (inNumber) {
      steeringWheelAngle += c;
    }
    if (c=='}'){
      inNumber = false;
      break;
    }
  }
  if (steeringWheelAngle.length() > 0) {
    steeringWheelAngle = steeringWheelAngle.substring(2,steeringWheelAngle.length()-2);
    //Serial.println(steeringWheelAngle);
    double wheelAngle = atof(steeringWheelAngle.c_str());
    //Serial.println(wheelAngle);
    return (int)wheelAngle;
  }
  return 0;
}
```