ECE 445
Senior Design Laboratory
Final Report

Camera Gimbal

Team 75
Girish Manivel (ggc2)
Harrison Liao (hzliao2)

TA: Ugur Akcal
Professor: Arne Fliflet

May 3, 2023

Abstract

This final report will provide detailed information for the problem, solution, and implementation of our senior design project. It will include descriptions of the project's design, features, cost, and ethical analysis.

Contents

1. Introduction

1.1 Problem

A major problem in video processing is footage that is shaky. If you take the forward direction as +y, right direction as +x, and up direction as +z; Shaky video footage is a result of the camera rotating around the +y and +x axes at minute steps. For example, if you take out your hand with your palm facing forward and pretend that it is a camera. Wave your hand as if you are waving hello. Moving your hand left and right is the camera rotating around the y axis also known as roll. If you move your hand up and down, bending at the wrist, it is the camera rotating about the x axis also known as pitch. Many video stabilization solutions are expensive and can cost upwards of $300 for a phone camera gimbal. The camera gimbal we will create will be a cheaper alternative that will still achieve industry standards in camera stabilization.

1.2 Solution

Camera stabilization, countering the shift in pitch and roll, is the key to solving this issue. To do this, we want to make a camera gimbal. This will allow for stability in camera footage given an initial starting orientation of the camera. Once a button is pressed, the camera gimbal will take in an initial orientation from a gyroscope sensor. This reading will go to an encoder to the microcontroller. Two servo motors, controlled by the microcontroller, will be used to maintain the initial orientation by opposing the shift in pitch and roll, keeping the camera stable.

1.3 Visual Aid

Figure 1: Visual Aid

1.4 High Level Requirements

For our project to be considered successful, we will need to meet the following requirements:

i. Camera is stabilized on +x axis (Pitch)
   This can be tested by isolating one of the servo motors and seeing how they oscillate as the gyroscope/gimbal is moving around. After researching industry standards [7] for gimbal slew rates (speed at which the gimbal can rotate in a given direction), we will try to achieve a minimum slew rate on the x axis of 100 degrees/second.

ii. Camera is stabilized on +y axis (Roll)
   This can be tested by isolating one of the servo motors and seeing how they oscillate as the gyroscope/gimbal is moving around. We will try to achieve a minimum slew rate on the y axis of 100 degrees/second.

iii. User interface (button) works
   - First button press turns the system on and starts reading gyroscope orientation.
   - Second button press locks the camera orientation by saving gyroscope reading and turns the hold orientation mode on.
   - Third button press switches to ground zero gimbal mode.
   - Fourth button press turns the system off.

2 Design

2.1 Block Diagram

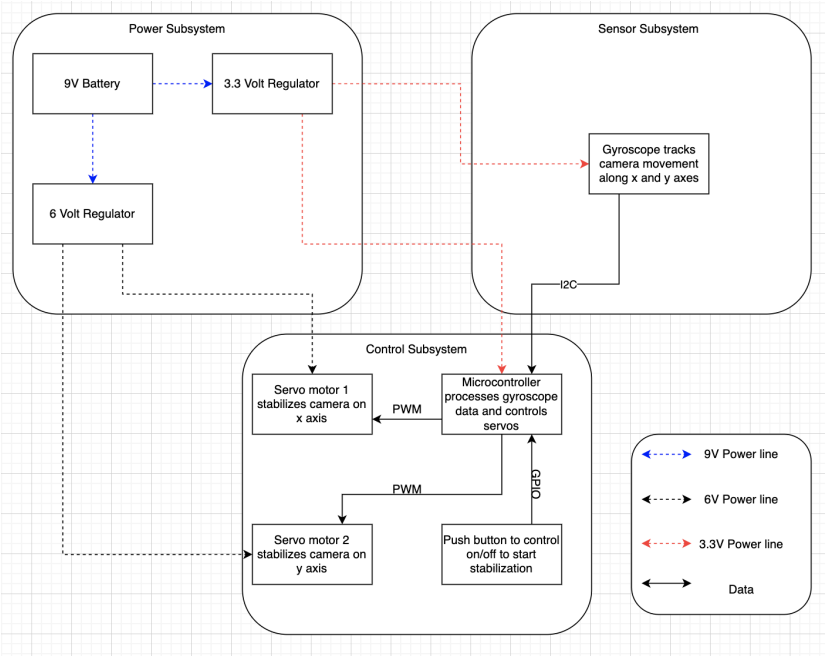

Figure 2: Block Diagram

2.2 Physical Design

In order to obtain safe and sound readings, we decided to place our gyroscope (which will be soldered onto the board) inside the enclosure so that there will be no variance of the orientation of the gyroscope compared to the handle. We will also be mounting our camera (GoPro Hero Session) to our gimbal. The GoPro is capable of streaming a live feed of the video footage to your phone through the GoPro app which we will be using to view the footage.

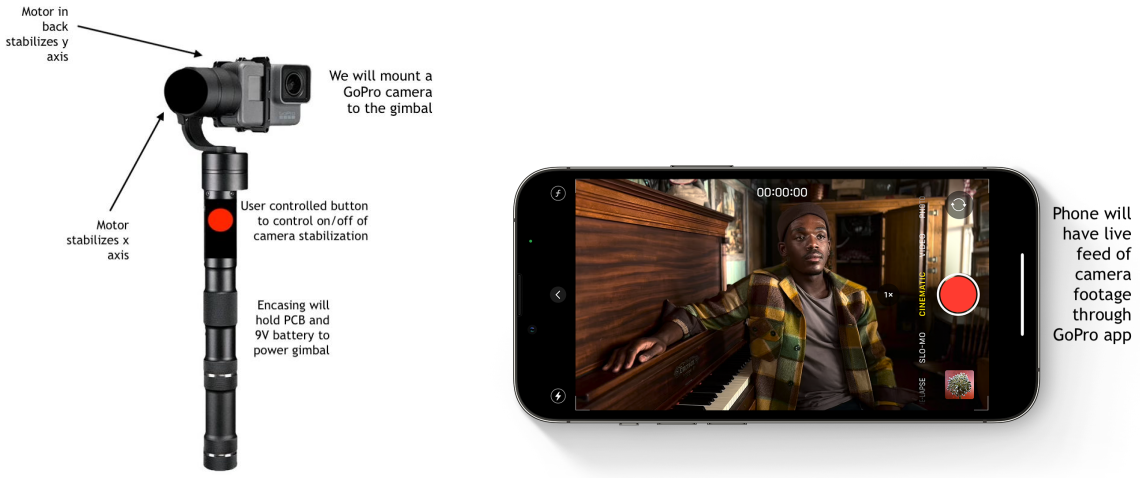

Figure 3: Physical Design

2.3 Power Subsystem

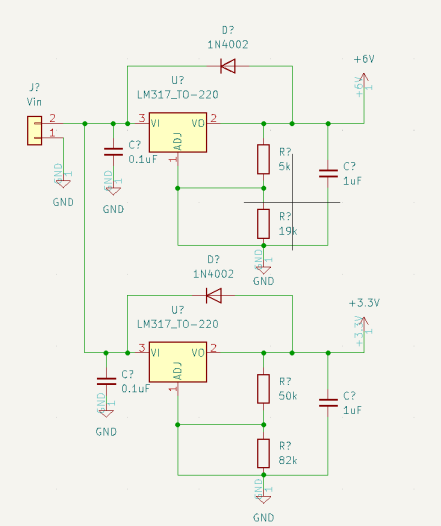

Figure 4: Original Power Subsystem Schematic

2.3.1 Overview

The purpose of this subsystem is to pull power from our 9-volt battery and route the required amounts to our other components: servo motors, gyroscope, and the microcontroller.

2.3.2 Requirements

Our power subsystem failed to satisfy the requirements we laid out in Table 1. We were unable to step down our voltage to 6V and 3.3V for the servo motors and microcontroller/gyroscope respectively. When we tested our PCB, we saw the regulators were outputting ~8V and ~7V for each regulator which is too much voltage for all our components.

2.3.3 Design Decisions

We decided to use the LM317T-ADJ for our voltage regulators as they are adjustable and were free at the ECE Electronics Service Shop. We used the equation from the datasheet [5] to pick our resistor values to step down the voltage: $V_{out} = 1.25(1 + \frac{R2}{R1}) + (I_{adj} * R2)$

At first, we ignored the Iadj*R2 part of the equation as it says Iadj is typically 50μA and negibile for most applications [5]. Because of this we picked resistor values of 5k and 19k ohms for the 6V regulator and 50k and 82k ohms for the 3.3V regulator. We did not realize this at first, but since our resistor values were so high, we are adding to the voltage output with the Iadj*R2 part as R2 is such a big value for us. Once we found this out through testing, we switched to a breadboard to test the voltage regulators with new resistor values. The values we decided to go with are 56 and 221 ohms for the 6V regulator as well as 500 and 825 ohms for the 3.3V regulator. When we used these resistor values on the breadboard with the LM317T, we were getting voltage outputs very close to 6V and 3.3V when testing with the multimeter. However, once we soldered the new resistors to our PCB, our regulators were still outputting 8V and 7V. Because of this, we decided that either the differing package sizes of the resistors or that the wiring of our original PCB was wrong. Due to this, we decided to redo our PCB with a new power subsystem with fixed voltage regulators instead of adjustable ones (See figure 5 below). We placed the order for new parts, however, the parts never came in so we were forced to switch to a development board (Arduino Uno) to implement our system.

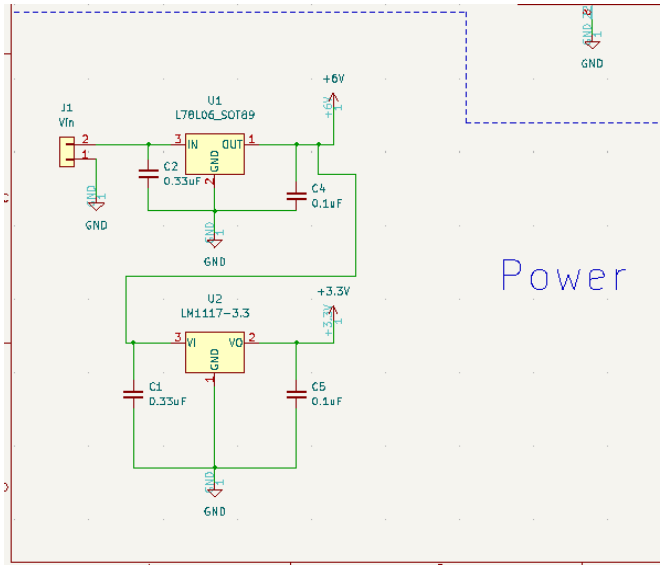

Figure 5: Revised Power Subsystem

2.3.4 Requirements and Verification Table

Requirements	Verification
1. Route the 9-volt ~500 mAh battery to two regulators where the 6 volt regulator can support up to 1A output (need a minimum of 800 mA to support servo stall). The 3.3 volt regulator can support up to 1.5 A of current, well surpassing the amount needed to support both the microcontroller and the gyroscope	• Ensure that the resistors used to control Vout have at least a ½ W power rating • Continuously check battery after testing in order to make sure it is outputting at least 6 V

Table 1: Power Subsystem Requirements and Verification

2.4 Control Subsystem

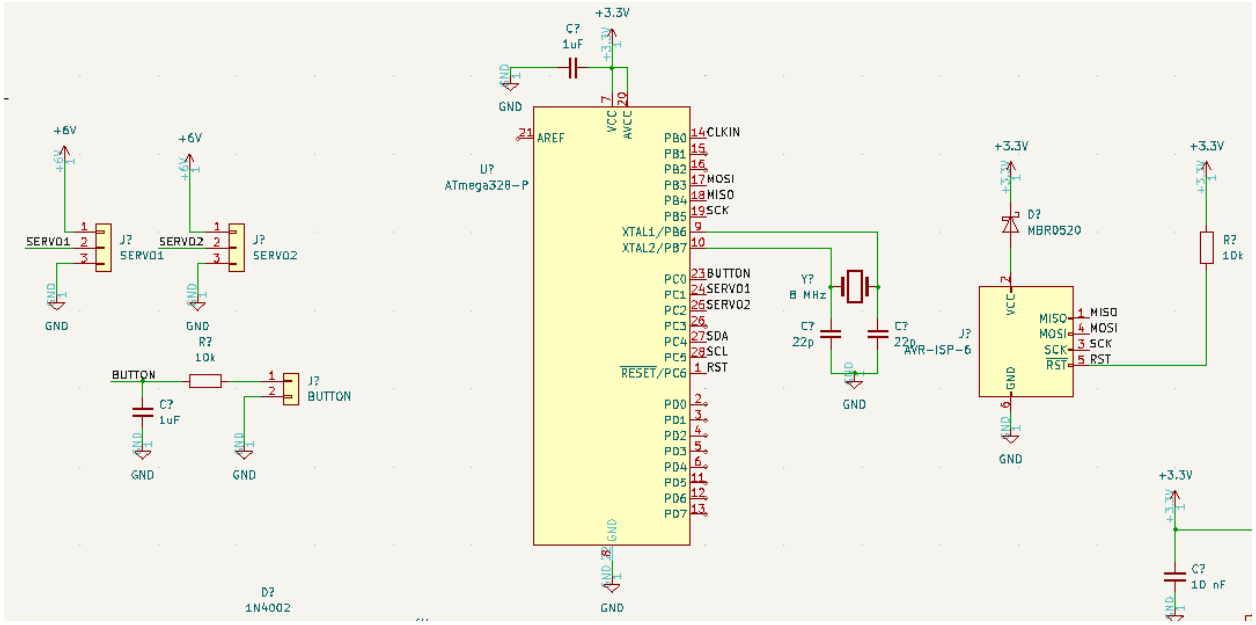

Figure 6: Control Subsystem Schematic

2.4.1 Overview

The purpose of this system is to actuate our motors in order to mimic a gyroscopic gimbal. We will use a microcontroller which interprets data from a gyroscopic sensor to set control inputs to motors.

2.4.2 Requirements

This system satisfies the following requirements:

1. The microcontroller will read data from the gyroscope, calculate necessary signals for both motors individually and update the motors input signal all in real time.
2. Each motor must be able to hold and alter the position to within 5° all while bearing the load of up to 250 grams.
    a. The first motor must be able to support the second motor as well as the camera mount and camera(< 250 g)
    b. The second motor must be able to support the camera mount and the camera

For the first requirement, we verified that the motors could move at or above a speed of 60 degrees every 0.15 seconds with the test code shown in figure 7. This is done by setting the servo motor to 20 degrees then starting a timer. We then tell the servo to write to 120 degrees and while the servo is reaching 120 degrees, we wait and run the timer. Once the servo reads 120 degrees, we will end the timer and divide the degrees moved (100) with the time in seconds to get the motor speed. For our first (pitch) motor, we were able to achieve a speed of around 250,000 degrees/second and our second (roll) motor was able to achieve a speed of around 500,000 degree/second, both of which are well above our requirement of 60 degrees/0.15 seconds which equals 400 degrees per second per the R&V table (Table 2).

```
Servo myservo;  // create servo object

void setup() {
  Serial.begin(9600);
  myservo.attach(6);  // attach servo to pin 6
}

void loop() {
  myservo.write(20);  // set servo to 20 degrees
  delay(1000);  // wait for servo to stabilize
  unsigned long start_time = micros();  // record start time
  myservo.write(120);  // set servo to 120 degrees
  while (myservo.read() != 120) {}  // wait for servo to reach position
  unsigned long end_time = micros();  // record end time
  float duration_seconds = (end_time - start_time) / 1000000.0;  // calculate duration in seconds
  float rotation_speed = 100.0 / duration_seconds;  // calculate rotation speed in degrees per second
  Serial.print("Rotation speed: ");
  Serial.print(rotation_speed);
  Serial.println(" deg/s");
  delay(1000);  // wait for servo to stabilize
}
```

Figure 7: Motor speed test code

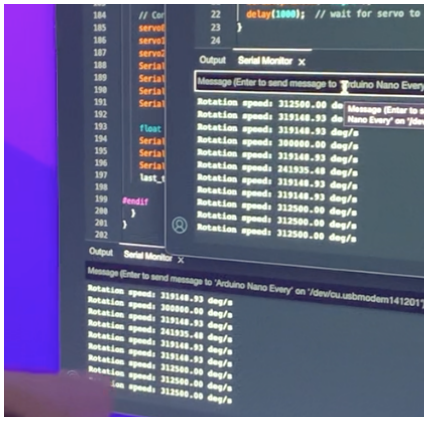

Figure 8: Pitch motor speed output

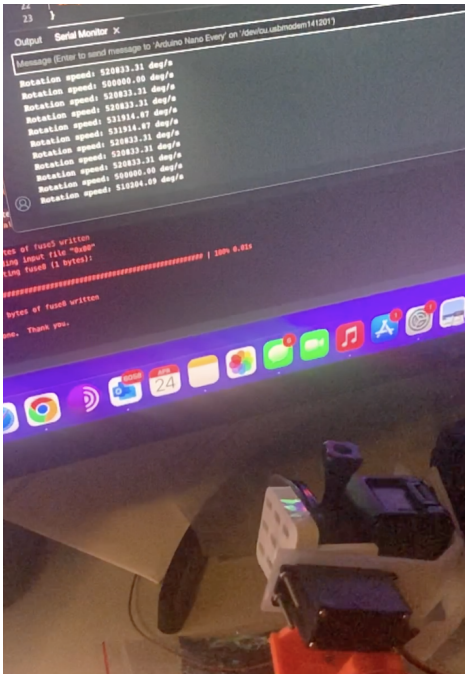

Figure 9: Roll motor speed output

For our second requirement, we were able to verify the weight of the GoPro camera and the camera mount to be 189 grams (see figure 10), which is above the requirement of 180 grams for the load of the second (roll) motor per table 2.

Figure 10: Camera mount weight

2.4.3 Design Decisions

We chose to use the ATMega328p [1] for our Gimbal since it had 6 PWM channels, and supported I2C communication for the gyroscope readings. This proved to be a great decision as explained below. Our next step was doing research on servo motors that could support the weight of 3D printed parts as well as the camera. We estimated that each motor mount would weigh

around 10 grams with the HS-311 [3] servo weighing ~43 grams. Since a motor needs to hold up to 250 grams, this allows us to have a camera device at ~185 grams. With our initial power subsystem, the microcontroller was powered with 3.3 V (Table 1), which means the microcontroller can only perform at a max speed of 10 MHz. A side-effect of our substitution of the power subsystem and the Arduino Uno dev board was that the SPX1117M3-L-5 regulates the voltage on the board to 5 V [10]. Therefore since the time of the demo, our microcontroller can run at a max speed of 16 MHz [1]. At 5 V, the HS-311 servo motors ran at 83% power, and with an original torque of 5.5 kg.cm , our updated torque is 4.58 kg/cm^2. This means that a motor will be able to hold 180 grams with enough torque up to 5 cm away from the motor. In order to accommodate this, the length of the 3D printed motor mount was 4 cm for the most efficient motor performance.

2.4.4 Requirements and Verification Table

Requirements	Verification
2. The microcontroller will read data from the gyroscope, calculate necessary signals for both motors individually and update the motors input signal all in real time	● The ATMega328p can support up to 10 MHz, which allows for a considerable update rate. ● Motors have a speed of 0.15sec/60° so a full rotation could be completed in 0.50 secs ● Motors will be able to keep up with consistent movement since the user physically won't move faster
● Each motor must be able to hold and alter the position to within 5° all while bearing the load of up to 250 grams. ○ The first motor must be able to support the second motor as well as the camera mount and camera(< 250 g) ○ The second motor must be able to support the camera mount and the camera	● With a torque of 5.5 kg/cm^2, the Gimbal can support up to 180 grams at the camera mount ● As long as the first motor, placed at the base, can support the weight at the "end-effector" the second motor will be perfectly fine ● In order to ensure that all weight could be easily supported for this prototype, we will be using a GoPro which weighs about 180 grams.

Table 2: Control Subsystem Requirements and Verification

2.5 Sensor Subsystem

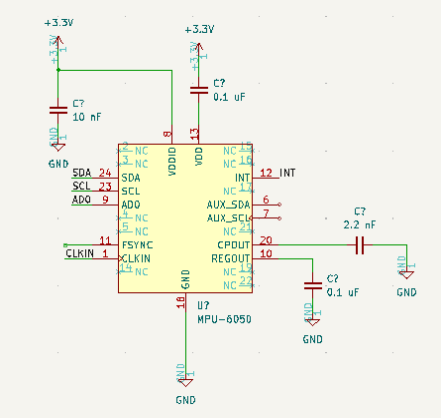

Figure 11: Sensor Subsystem Schematic

2.5.1 Overview

To understand the purpose of this subsystem, we need to first understand the mechanics of the device. We will have a user controlled handle where the control end of the handle will house our Pitch movement, and directly above that another motor will control the Roll movement. The gyroscope will be attached at the base of these motors so that the far end of the handle will be the modular platform (refer to the visual aid). Our gyroscopic sensor will read positional data to tell us how the gimbal is oriented. We will continuously read these output values in order to simultaneously control servo positions.

2.5.2 Requirements

The system satisfies this requirement:

1. The gyroscope must be able to obtain accurate readings throughout gimbal activation at a rate of at least 1000 Hz.

To test our computation speed, we used the inbuilt micros() function which provides the microseconds since the Arduino started running the program. We first set the variable start_time to the moment right before we get gyroscope readings. Then we retrieve the data from the mpu, and convert the readings to PWM signals for the motors. Finally, we set end_time to the moment after the computations are done. To convert to MHz, we simply compute $\frac{1\ (cycle)}{start\_time - end\_time} * 1000000\ (microsec\ per\ second)$. Please refer to Figure 12 for the code explained above, and Figure 13 for results.

We also support continuous reading by including the retrieval and storage of the MPU6050 readings in each case/mode of our loop() function.

```
    // Get Yaw, Pitch and Roll values
#ifdef OUTPUT_READABLE_YAWPITCHROLL
    start_time = micros();
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    // Yaw, Pitch, Roll values - Radians to degrees
    ypr[1] = ypr[1] * 180 / M_PI;
    ypr[2] = ypr[2] * 180 / M_PI;

    // Map the values of the MPU6050 sensor from -90 to 90 to values suatable for the servo control from 0 to 180
    int servo1Value = map(ypr[1], -90, 90, 0, 180);
    int servo2Value = map(ypr[2], -90, 90, 180, 0);
    int angle1 = abs(oldServo1Value - servo1Value);
    int angle2 = abs(oldServo2Value - servo2Value);
    // Control the servos according to the MPU6050 orientation
    servo1.write(servo1Value);
    servo2.write(servo2Value);
    end_time = micros();
    Serial.print("\nPitch: ");
    Serial.println(ypr[1]);
    Serial.print(" Roll: ");
    Serial.println(ypr[2]);
    float duration_seconds = (end_time - start_time) / 1000000.0;  // calculate duration in seconds
    float rotation_speedP = angle1 / duration_seconds;  // calculate rotation speed in degrees per second
    float rotation_speedR = angle2 / duration_seconds;  // calculate rotation speed in degrees per second
    Serial.print("Pitch speed: ");
    Serial.println(rotation_speedP);
    Serial.print("Roll speed: ");
    Serial.println(rotation_speedR);
    oldServo1Value = servo1Value;
    oldServo2Value = servo2Value;

#endif
  }
}
```

Figure 12: Update rate test code

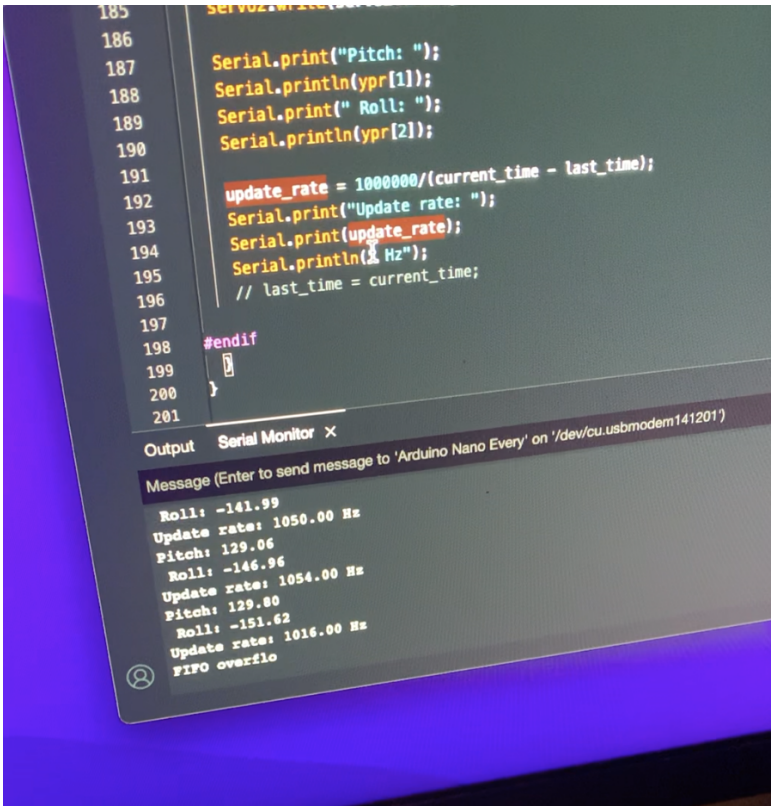

Figure 13: Update rate test output

2.5.3 Design Decisions

Due to the operating voltage of the MPU6050 ranging from 2.375 to 3.46 Volts [2], we had to step down our input voltage to at last 3.46 V, hence why we attempted using a LM317 [5] to step down to 3.3 V. When deciding on which arduino board we needed to use, we needed to make sure there was an option to power other components with 3.3 V. The Arduino Uno satisfied this requirement as it uses another LDO to step down to the same 3.3 V. In our original design, we had implemented a pcb with the gyroscope on the same board, with the intent of being powered by the correct voltage. But due to space concerns in fitting both the Arduino board as well as our pcb, it was more beneficial to use an MPU6050 Gyro Module [11]. We still needed to secure the module to obtain accurate readings, so we used a hot glue gun to hold the module in place throughout the Gimbal's functionality.

2.5.4 Requirements and Verification Table

Requirements	Verification
3. The gyroscope must be able to obtain accurate readings throughout gimbal activation at a rate of at least 1000 Hz.	• Our Gyroscope will be powered and taking readings at all times there is any power supplied to the Gimbal • The constant readings at 30 kHz allows us to pinpoint the motors positions at all times

Table 3: Sensor Subsystem Requirements and Verification

2.6 Tolerance Analysis

The most critical portion of our Gimbal is ensuring that the gimbal is able to react to changes in movements in a small enough time frame that there is very minimal to no visual shake in the camera feed. As there is no real metric to measure the shakiness of a camera feed in either mechanics or the digital output, our standard we want to achieve is at a minimum, an emulation of the industry standard camera gimbal [7]. This gimbal setup achieves a slew rate of 100°/sec which is the minimum for our project, but since this is for a much larger camera load than our intentions, we plan to perform at a rate of 200°/sec. In order to achieve this, we needed to first ensure that our data protocols could run at a sufficient speed in order for this to work.

In our testing and application, the GoPro HERO Session can record 1080p footage up to 60 fps. Since our HS-311 performs at a rate of 400°/sec with no load at 6.0 Volts.

$0.15\,sec/60° \Rightarrow (60°/0.15\,sec) * \frac{3}{20} \Rightarrow 400°/sec$

This maximized strength of the motor means that we are able to perform well above the required speed in order to maintain stable footage.

60 fps and a motor speed of 400°/sec allows us to complete a full servo rotation in between each frame in the video.

The data signals to control this movement originate from the gyroscope which has an output data rate of 8 KHz [2]. This data is then sent to the microcontroller with an internal clock of 1 MHz [2] and processing speed of about 12 MHz at 3.3 Volts [1].

The ATmega328P is configured to use Timer/Counter 1 for both PWM signals, and a prescaler value of 64 is used with a TOP value of 255 (8-bit resolution) [1], the total number of clock cycles required for one PWM cycle for each motor would be:

(1 / (12 MHz / 64)) * (256 cycles + 1 cycle for overflow) = 0.00136533 seconds = 1365 clock cycles.

Since we are only doing a simple read of the gyroscope, this will add another 200 clock cycles. This plus the amount of time to send PWM signals to both motors, we are at 2930 clock cycles total to get positional data from the gyroscope, process data in the microcontroller, then send PWM signals to the motors. With a 12 MHz clock, we are able to constantly update our motor positions through PWM signals more than 4000 times a second. This many updates within each frame means that we will have steady footage even after the physical movement time of the HS-311 motors.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

As Computer Engineering students, we will estimate the cost of labor based on the median starting salary for students that graduate with Computer Engineering degrees from the Grainger College of Engineering which is $105,352 [8]. Based on a 40 hour work week, it comes out to $50.65/hour.

Category of Work	Estimated Hours Girish	Estimated Hours Harrison
Circuit and PCB Design	20	20
Software Programming	20	20
Soldering	1	3
CAD for Encasing/Gimbal Grip	15	5
Prototyping and Debugging	40	40
Documentation and Logistics	40	45
Total Hours	136	133

Table 4: Estimated Labor Hours

Based on the estimated labor hours from table 4 and the computed hourly salary for each member the total cost of labor is:
$50.65 (Hourly Rate) * 269 (Total Estimated Hours) = $13,624.85

This comes out to $6812.43 per partner. When adding a 2.5x overhead multiplier, the total cost of labor comes out to $34,062.13 or $17,031.06 per partner.

3.1.2 Parts

Description	**Part #**	**Manufacturer**	**Quantity**	**Cost**
Adjustable Linear Voltage Regulator 1.2-37V	LM317T	STMicroelectronics	2	$0.00
Gyroscope	MPU-6050	TDK InvenSense	1	$8.00
9V Battery	EN-22	Procell	1	$1.67
9V Battery Clip	233	Procell	1	$0.60
3 Position MTA 100 Header	640456-3	TE Connectivity	2	$0.28
2 Position MTA 100 Header	6404562	TE Connectivity	1	$0.06
Servo Motor	HS-311	HiTec	2	$27.08

ATMega Microcontroller	ATMega328-P	Atmel	1	$7.20
Crystal 16.0Mhz SMD KX-7T 12pF,	12.88722	Geyer Electronic America, Inc.	1	$0.35
16mm Momentary Push Button Switch SPST	R13-507	weideer	1	$8.99
Thin Film Resistors - SMD RA73F 2A 10K 0.1% 1K RL	RA73F2A10KB TDF	TE Connectivity	2	$4.32
Thin Film Resistors - SMD RQ 603 82K5 0.1% 10PPM	RQ73C1J82K5B TDF	TE Connectivity	1	$1.09
Thin Film Resistors - SMD 3503G 2B 19K1 1% 1K RL	3503G2B19K1F TDF	TE Connectivity	1	$1.64
RES SMD 50K OHM 0.1% 1/4W 1206	RT1206BRD0750KL	YAGEO	1	$0.65
RES SMD 5K OHM 0.1% 1/4W 1206	RT1206BRD075KL	YAGEO	1	$0.65
Multilayer Ceramic Capacitors MLCC - SMD/SMT T101 COMMERCIAL	04023C104KAT2A-62	KYOCERA AVX	4	$1.48
Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 1uF X8L 1206 10% AEC-Q200	C1206C105K3NACAUTO	KEMET	4	$2.12
Multilayer Ceramic Capacitors MLCC - SMD/SMT 25Vol 22pF X7R 0603 10% AEC-Q200	C0603C220K3RACAUTO	KEMET	2	$0.30
Multilayer Ceramic Capacitors MLCC - SMD/SMT CGA 0603 50V 0.01uF X7R 10% AEC-Q200	CGA3E2X7R1H103K080AA	TDK	1	$0.10

Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 2200pF X7R 0402 5% AEC-Q200	04023C222J4T2A	KYOCERA AVX	1	$0.11
Rectifiers Diode, DO-41, 100V, 1A	1N4002	Diotec Semiconductors	2	$0.80
Schottky Diodes & Rectifiers 20Vr 14Vrms 20V 5.5A 0.45V 30pF	MBR0520-TP	Micro Commercial Components	1	$0.34
CONN HEADER VERT 6POS 2.54MM	BHR-06-VUA	Adam Tech	1	$0.25

Table 5: Parts and cost

The total cost of parts before shipping as shown above in Table 5 is $67.53. With around 5% in shipping and a sales tax of 6.25%, the total cost of parts will be around $75.13.

3.1.3 External Materials and Resources

We 3D printed the PCB enclosure/grip as well as the motor mounts. With a rate of $0.10 per gram [9] from the Illinois makerlab to 3D print, and a weight of about 180 grams for our external parts, this cost us around $18.

3.1.4 Total Sum of Cost

Category	**Cost**
Labor	$13,624.85
Parts	$75.13
External Materials	$18
Total Cost	**$13,717.98**

Table 6: Total Cost

4 Conclusion

4.1 Accomplishments

Holistically, our project was successful in that we fabricated a device that works as an efficient camera gimbal.

- We were able to build a clean and safe enclosure that is able to hold all components of the device. All components are out of reach of the user except the motors and their wires which have to be outside the enclosure.
- Our project is able to obtain gyroscope signals, modify them into PWM signals, and send those signals to the motors. At this base state, we have a functioning gimbal but tuning the gyroscope and motors was very beneficial to the performance.
- Successfully implemented multiple different modes for operation that can be decided via a user controlled button. Our project was initially meant to have 3 modes but this was changed so that the user flow made more sense. The user can turn on the device, which can then be used as a generic 'selfie stick' then with another press, the orientation at the moment the button is pressed is maintained. If this is not favorable, another press holds the camera at a ground zero position for a neutral perspective. Finally, pressing the button another time turns the system off awaiting the next press to go back to the first mode.
- Subsystem requirements were met and tested for all subsystems but power.

4.2 Uncertainties

The only flaw in functionality we noticed through our testing was jittery motor control in the orientation lock mode. By debugging this mode, we found that the cycle speed for computing the intended motor position based on the stored orientation lock was much slower than the other modes. Our initial code used the inbuilt Arduino function constrain() where, as its name suggests, it constrains a variable to a specified range. This function essentially just uses if/else statements to check against the range specified earlier. We realized that the error lies in the amount of time it takes to make the computations and since the only difference between this mode and the ground zero mode is that we are using the constrain() function. The cycle speed of the constrain() function was ~0.5 MHz but when we use switch case statements instead, we saw a slight increase to ~0.63 MHz. This increase in speed was able to slightly decrease the jitteriness in the orientation lock mode but this tells us that the processing speed is the root cause for the shakiness in this mode.

The most integral component of our system that failed was the power subsystem. We overlooked the Iadj term in calculating the output voltage of the LM317 adjustable regulator circuit. We initially expected this term to be negligible but after testing the pcb we saw that the output voltage was too high (Section 2.3.3). As a result, we redesigned the pcb using fixed voltage regulators to avoid buggy circuitry.

4.3 Future Work / Alternatives

Our next steps are mostly quality oriented excluding the issue described above. Assuming we had more time to work on this project, our first objective would be to obtain the new parts for our new pcb and build out the pcb. This would greatly improve the aesthetics inside the enclosure all while completely customizing the gimbal to our intended specifications. A change we think would greatly help the performance of our device is to upgrade the motors to one that has a higher torque output.

The gimbal would be able to support heavier cameras and the slack produced by the weight of the camera on each motor would be far less noticeable. In regards to functionality, our main goal is to edit the orientation mode to provide stable motor actuation by hastening the runtime of control signal calculations.

4.4 Ethics and Safety

Our group will follow the IEEE Code of Ethics and will hold all team members accountable to the highest ethical standards which will include, but is not limited to:

1. To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities. [4]

   We will make sure that our design will not put any users into harm's way and will disclose anything that may be deemed unsafe or can put people or the environment at risk. We will seek and accept honest criticism of our work by going to the professor or our TA often for feedback on our designs. We will also make sure that we are technically competent in what we do which includes the completion of Lab Safety Certification, CAD assignment, and soldering assignment.

2. To treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others. [4]

   Our group will ensure that everyone in our team as well as the rest of our class is treated with respect regardless of age, gender, race, etc. Harassment and bullying will not be tolerated and will be reported. Everyone in the group will have access to all files and work done by members of the group. All group members are also roommates so there will be ample communication throughout the group. When working in the lab, the group members will avoid injuring others physically and emotionally.

3. To strive to ensure this code is upheld by colleagues and co-workers. [4]

   We will support each other in upholding the highest ethical standards and to speak up if someone is not following these standards.

In regards to the safety and regulations for our project:

1. We will enclose our motors and electronics in plastic boxes to ensure that nothing can pop out and harm the user.
2. We will ensure that our project follows relevant licensing terms and follow the terms of service for the software components used in our camera gimbal system.
3. We will be using lithium ion batteries in our project which can be dangerous, so we will be following the guidelines laid out in the ECE 445 General Battery Safety document [6] in order to mitigate the risk of fire and damage to life or property. This will be done by reviewing datasheets and restrictions for the battery we will use, ensure that all components used and circuit designs are reviewed and validated by all team members as well as a TA, simulating the power circuitry before connecting it to hardware, and ensure that the battery and the components it will be connected to are stored in a safe place.

5 References

[1] "Atmega88a," *Microchip ATMega88A*. [Online]. Available: https://www.microchip.com/en-us/product/ATmega88a. [Accessed: 09-Feb-2023].

[2] "Haoyu Electronics : MPU-6000 and MPU-6050 Product Specification Revision 3.3." [Online]. Available: https://www.haoyuelectronics.com/Attachment/GY-521/mpu6050.pdf. [Accessed: 09-Feb-2023].

[3] "HS-311 Standard Economy Servo," *HiTec*. [Online]. Available: https://hitecrcd.com/products/servos/analog/sport-2/hs-311/product. [Accessed: 08-Feb-2023].

[4] "IEEE code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2023].

[5] "LM317 3-terminal adjustable regulator - texas instruments," *TI*, Apr-2020. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm317.pdf?ts=1641097833769. [Accessed: 02-May-2023].

[6] "Safe Practice for Lead Acid and Lithium Batteries", General Battery, ECE445 Senior Design Spring 2016 Course Staff, Jan. 2016. [Online]. Available: https://courses.grainger.illinois.edu/ece445/documents/GeneralBatterySafety.pdf.

[7] "Shotover G1 a lightweight, weather resistant, gryo-stabilized gimbal platform," *Shotover*. [Online]. Available: https://shotover.com/products/g1. [Accessed: 02-Mar-2023].

[8] T. G. C. of Engineering. ""The Grainger College of Engineering - Computer Engineering"." (2022), [Online]. Available: https://grainger.illinois.edu/academics/undergraduate/majors- and- minors/computer- engineering#:~:text=Post%5C%2DGraduation%5C%20Success&text=The%5C%20average%5C%20salary%5C%20between%5C%202020,median%5C%20signing%5C%20bonus%5C%20of%5C%20%5C%2415%5C%2C000 [Accessed: 23-Feb-2023].

[9] "What we offer," *Illinois MakerLab*. [Online]. Available: https://makerlab.illinois.edu/pricingservices/#printing. [Accessed: 23-Feb-2023].

[10] "Arduino® Uno R3." [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. [Accessed: 18-Apr-2023].

[11] A. D. SAVAGE, "MPU6050 3 Axis Accelerometer Gyroscope Module," Amazon, https://www.amazon.com/dp/B00LP25V1A?ref=ppx_pop_mob_ap_share. [Accessed 18-Apr-2023].