Solution

- Camera stabilization, countering the shift in pitch and roll, is the key to solving our problem.
- Gimbal maintains camera in its initial starting orientation, using motors to oppose shift in pitch and roll.
- 3D printed parts and servo motors allow for a more cost-effective solution.

High Level Requirements

1. Camera is stabilized on +x axis (Pitch)
   - minimum slew rate on the x axis of 100 degrees/second.
2. Camera is stabilized on +y axis (Roll)
   - minimum slew rate on the y axis of 100 degrees/second.
3. User interface (button) works
   - First button press turns on system 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 normal gimbal mode.
   - Fourth button press turns the system off.
Design
Enclosure

- Camera Mount
- Motor Mount
- Enclosure Front
- Enclosure Back

Design

- Motor Mount
- Camera
- Button
- Enclosure Mount
## Problems

<table>
<thead>
<tr>
<th>The Issue</th>
<th>The Cause</th>
<th>The Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage Regulators were outputting voltages far too high (7 – 8 Volts)</td>
<td>Wiring issues in PCB with certain components and some miniscule resistors that were tough to solder</td>
<td>Switch to fixed voltage regulators and order resistors with larger package sizes</td>
</tr>
<tr>
<td>New parts orders did not arrive in time for the final demo</td>
<td>Parts order was not reviewed</td>
<td>Arduino Nano Dev board</td>
</tr>
<tr>
<td>Very jittery movement in orientation holding mode</td>
<td>Logic calculating altered positions were inefficient</td>
<td>Use switch case statements instead of the constrain or if/else statements</td>
</tr>
</tbody>
</table>
Power Subsystem

Requirements:

• Step down the 9V battery through 2 voltage regulators to 6V for the servo motors and 3.3V for the microcontroller and gyroscope.

Verification:

• Battery outputs 9V.
• Resistors step down voltage to 6V and 3.3V.
Control Subsystem

Requirements:

• Data Path: Gyro -> Microcontroller (modifies data) -> PWM signals to motor

• Motors must support a load of up to 180 grams (Camera/Phone)

Verification:

• Motor speed of at least 400 degrees/second

• Motors can support 189 g
Sensor Subsystem

Requirements:

- Gyroscope readings must occur at a minimum rate of 1000 Hz

Verification:

- Obtains readings and computes PWM signals (extra) at around 1050 Hz
Results
Overall Results
Results

Control Subsystem

- Camera mount weight
Control Subsystem

• Pitch motor speed test

Motor speed test code

```c
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(90); // set servo to 90 degrees
  while (servo.read() < 90) {} // 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 = 180.0 / 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
```

Motor speed test readout
Control Subsystem

• Roll motor speed test

```c
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(150); // set servo to 150 degrees
  while (myServo.read() != 90) {} // 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 = 180.0 / duration_seconds; // calculate rotation speed in degrees per second
  Serial.print("Rotation speed: ");
  Serial.println(rotation_speed);
  Serial.println("deg/s");
  delay(1000); // wait for servo to stabilize
}
```

Motor speed test code
Sensor Subsystem

- Roll motor speed test

```c
// Get Yaw, Pitch and Roll values

void OUTPUT_READING_YPR(ROLL)

float last_roll = micros();
mpu9250_get_quaternion_gyro_accroll(mpu_9250_quaternion, mpu_9250_gyro, mpu_9250_acc, mpu_9250_roll); // Yaw, Pitch, Roll values - Radians to degrees

roll = (int)(mpu_9250_roll * 180 / M_PI);

// Map the values of the MPU9250 sensor from -90 to 90 to 0 to 180 suitable for the servo control from 0 to 180
int servoValue = map(mpu_9250_roll, -90, 90, 0, 180);

// Control the servo according to the MPU9250 orientation
servo.write(servoValue);
Serial.print("Roll: ");
Serial.println(servoValue);
Serial.print("Pitch: ");
Serial.println(roll);
Serial.print("Yaw: ");
Serial.println(yaw);
Serial.print("Update rate: ");
Serial.println(updates_rate);

// last_roll = current_time;
```

Sensor update rate test code
Conclusion
What we learned

• Test circuitry on breadboards before designing PCB to have a stronger initial draft
• Thorough research prior to purchasing and testing components
• Don't be afraid the escalate issues and ask for help
Next Steps

- Secure new parts to create a fully functional PCB with linear voltage regulators
- Reprint 3-D enclosures without defects for a cleaner final product
- Swap motors to increase torque and speed to improve overall performance of Gimbal
- Improve orientation lock mode to have no jittery movement