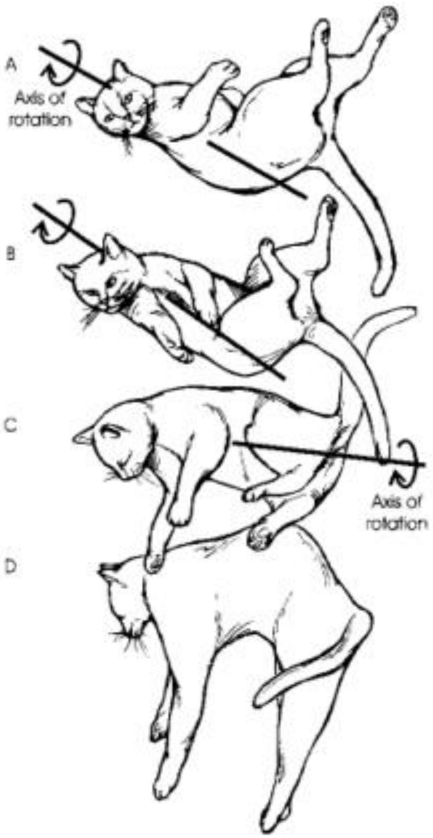
Advanced Mechatronics Project 1: Arduino

Members: Manoj Bandri, Wenjie Chen

Presentation Date: 3/21/16

MOTIVATION

- When a cat falls in the air with its back facing the ground, it knows how to maneuver itself to land upright on its feets
- Robotic systems can also take advantage of such maneuver to properly orient itself in the case of falling from heights



Conservation of Angular Momentum

- Moment is equal to the derivative of angular momentum with respect to time

$$\Sigma M_o = \dot{H}_o$$

- The total angular momentum of a system is conserved (constant) when no external moments are applied to the system.

$$\dot{H}_o = 0 \qquad H_o = \text{constant}$$

- Cats are capable of orienting itself in mid air due to internal moments applied

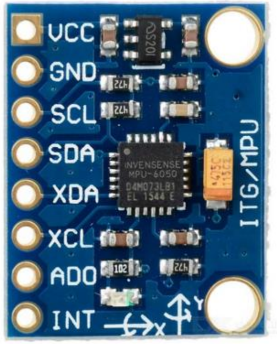
Project Idea

- Motors can be implemented into a falling object to change its orientation in mid air
- Force of Gravity and Drag Force applied
- Moment applied due to drag force can be neglected
- To simplify problem, only rotation with respect to the z-axis will be controlled by a motor

Materials

- Arduino Uno

- MPU 6050 (6-Dof Accelerometer and Gyroscope)



- Plastic Enclosure

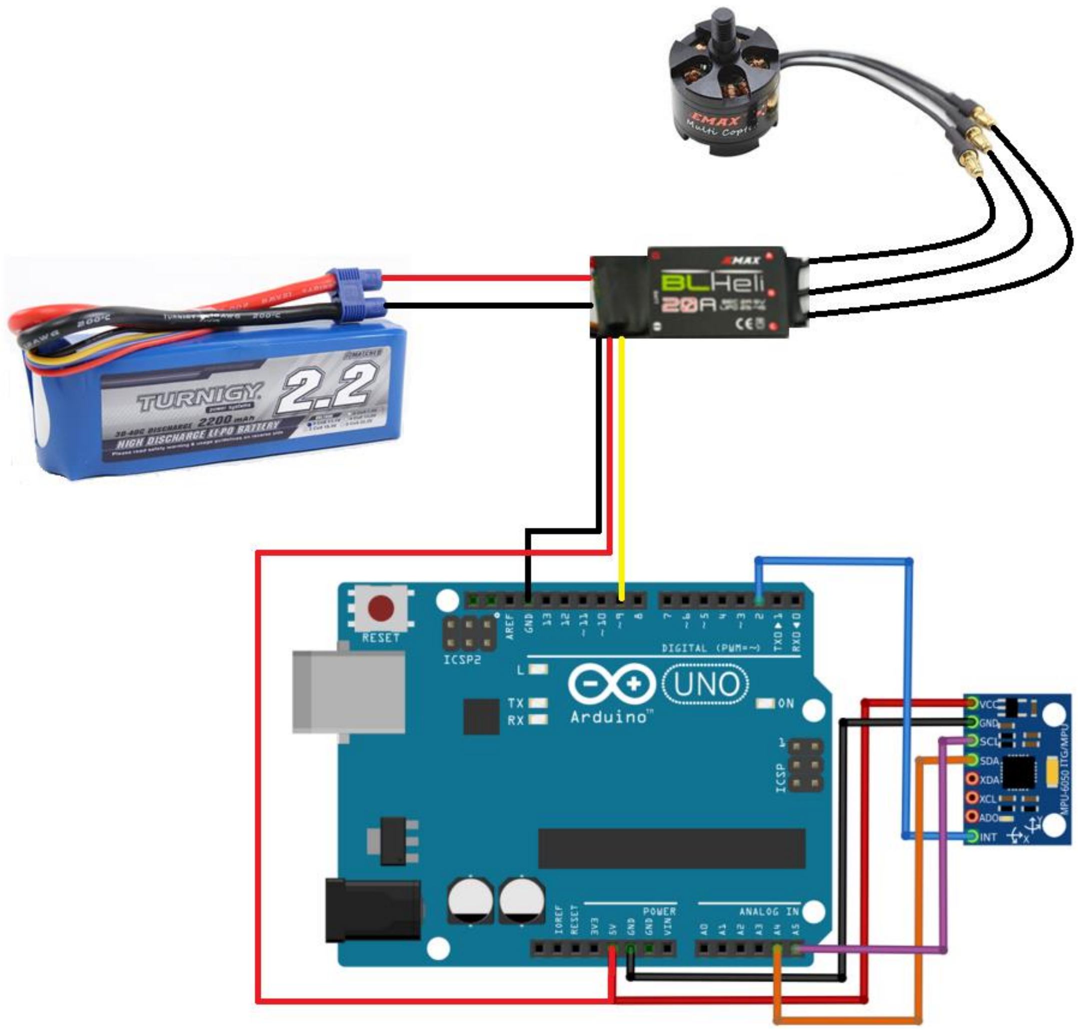
MATERIALS

- Brushless DC motor

- ESC (Electric Speed Controller)

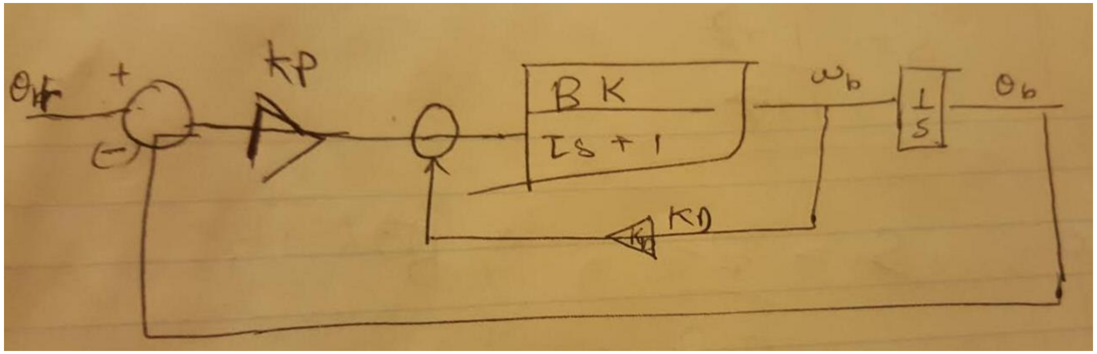
- LiPo Battery (suitable for esc and motor)

Circuit Diagram



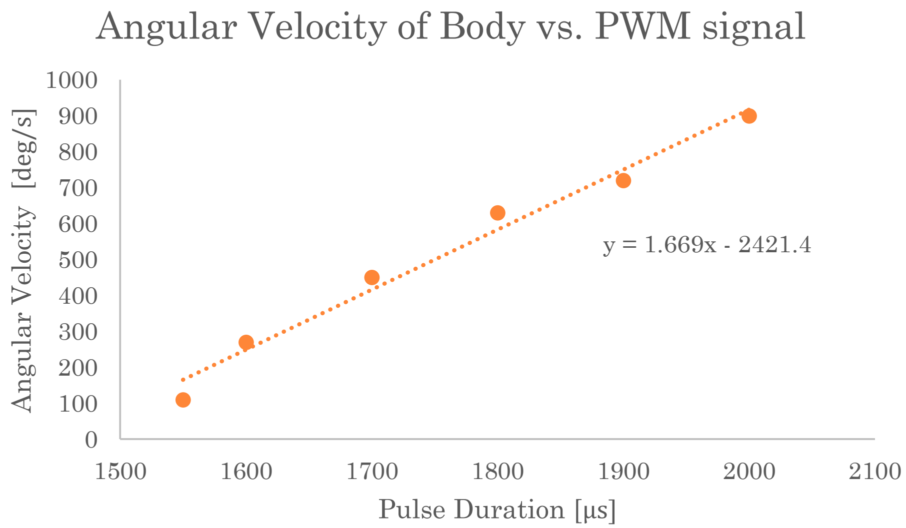
APPROACH

- Ideal Solution:
  - DC Identification of Brushless DC Motor
  - Determine K and τ
  - Determine MoI of the entire body and motor
  - Implement PD controller to output angular position of the entire body from input of angular velocity of motor
- Problems:
  - Arduino's processing speed cannot keep up with the high speed of motor → Cannot find complete relation between PWM and motor speed to control input
  - For motor speeds that are measureable, data gathered does not behave as governed by DC motor transfer function → Cannot determine K and τ
  - Cannot determine MoI with available equipments



APPROACH

- Alternative Solution:
  - Assume direct control of entire body based on PWM input
  - Approximate angular velocities for different inputs of PWM signals for 1 sec, neglecting acceleration from rest and deceleration to rest
  - Assume linear correlation between PWM signal and the entire body speed based on experimental data



Control Algorithm

- The body must orient itself back to its reference angular position as it drops before landing

- Apply appropriate PWM signal to rotate the body within the time of drop based on:
  - Angular position offset ("degree")
  - Height from which it is dropped ("pos_z_curr")

- Use MPU-6050 to gather information on angular position and height

CONTROL ALGORITHM

- Calculate position from uniform acceleration

$$a = \frac{dv}{dt} \rightarrow v - v_0 = a(t - t_0)$$

  - $v = v_0 + a\Delta t$

$$v = \frac{dx}{dt} \rightarrow \frac{dx}{dt} = v_0 + a\Delta t$$

  - $x = x_0 + v_0\Delta t + \frac{1}{2}a(\Delta t)^2$

- Calculate angular position from angular velocity

$$\omega = \frac{d\theta}{dt} \rightarrow \theta - \theta_0 = \omega(t - t_0)$$

  - $\theta = \theta_0 + \omega\Delta t$

Control Algorithm

- Calculate time it takes to fall from rest

$$0 = h - \frac{1}{2}g(\Delta t)^2$$

$$\Delta t = \sqrt{\frac{2h}{g}}$$

- Approximate uniform angular velocity

$$\omega = \frac{\Delta\theta}{\Delta t}$$

- Set signal bases on required angular velocity
  - Positive angular offset

$$s = \frac{\omega + 2421.4}{1.669}$$

  - Negative angular offset

$$s = \frac{\omega + 2583.4}{1.669}$$

Program

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include <Servo.h>
Servo bldc;
float rotate;

MPU6050 accelgyro;

int16_t ax, ay, az, gx, gy, gz;

#define LED_PIN 13
bool raised = false;
bool blinkState = false;
bool disorient = false;
bool fall = false;
int addr = 0;
int accel_reading;
int accel_corrected;
int accel_offset = -256;
float accel_z;
int vel_z_prev = 0;
int vel_z_curr;
float pos_z_prev = 0;
float pos_z_curr;
int spin_time;
int i = 0;
```

```
int gyro_offset = 151;
int gyro_corrected;
int gyro_reading;
float gyro_rate;
float gyro_scale = 0.02;
float gyro_angle;
float degree;
float omega;
float loop_time = 0.05;

int last_update;
int cycle_time;
long last_cycle = 0;
```

Program

```
void setup() {

  Wire.begin();
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
  bldc.attach(9);
  delay(3000);
  accelgyro.initialize();

  while (i<100){
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    accel_reading = az;
    accel_corrected = accel_reading - accel_offset;
    accel_z = (float)(accel_corrected)*9.81/16600;
    vel_z_curr = vel_z_prev + 100*(accel_z-9.775)*loop_time;
    if (vel_z_curr>0){
      pos_z_curr = pos_z_prev + vel_z_prev*loop_time + 1/2*(accel_z-9.775)*100*sq(loop_time);
    }
    vel_z_prev = vel_z_curr;
    pos_z_prev = pos_z_curr;
    i++;
    time_stamp();
  }
  raised = true;
}
```

Program

```
void loop() {

  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  accel_reading = az;
  accel_corrected = accel_reading - accel_offset;
  accel_z = (float)(accel_corrected)*9.81/16600;

  if (vel_z_curr==0){
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
  }

  gyro_reading = gz;
  omega = -((float)((gyro_reading)/131 - gyro_offset) *
          gyro_scale - 0.151/(-loop_time))*50.0;
  gyro_angle = gyro_angle + omega * -loop_time;
  degree = gyro_angle;

  if(degree>180){
    degree = degree - 360;
  }
  else if(degree<-180){
    degree = degree + 360;
  }
```

```
if (accel_z<8){
  fall = true;
}
else {
  fall = false;
}

if (abs(degree)>10){
  disorient = true;
}
else{
  disorient = false;
  fall = false;
  bldc.write(1499);
}
```

Program

```
  if( ((fall & disorient) & raised) ){
    float t = sqrt(2.0*pos_z_curr/9.81/100);
    float w = degree/t;
    int counter=0;
    int msec = t*1000;
    int PWM = 1499;
    //  >1500 rotates box clockwise
    //  <1498 rotates box counter clockwise
    if (degree>10){
      PWM = (w/2 + 2421.4)/1.669;
    }
    else if (degree<-10){
      PWM = (w/2 + 2583.4)/1.669;
    }
    counter = millis()+msec;
    while(millis()<counter){
      bldc.write(PWM);
    }

    if (degree>10){
      bldc.write(1510);
    }
    else if (degree<-10){
      bldc.write(1490);
    }
    raised = false;
  }
  time_stamp();
}
```

```
void time_stamp(){
  while ((millis() - last_cycle) < 50){
  delay(1);
  }
  last_cycle = millis();
}
```

Demonstration