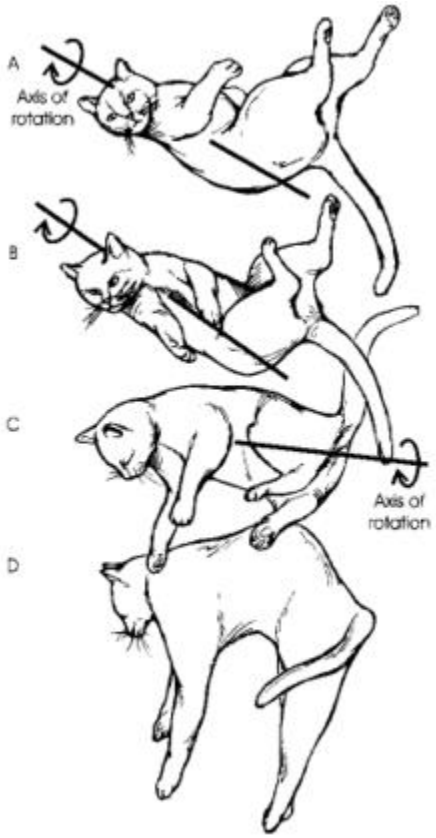
Advanced Mechatronics Final Project Raspberry Pi

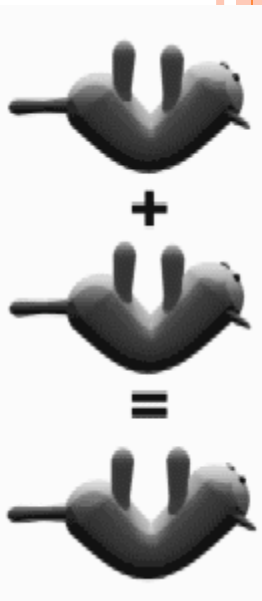
Members: Manoj Bandri, Wenjie Chen

Presentation Date: 5/13/16

Motivation

- Last time emphasizing motivation
- When a cat falls in the air, it knows how to reorient itself to land upright on its feet
- Robotic systems can also take advantage of such maneuver to properly orient itself and land properly





Conservation of Angular Momentum

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

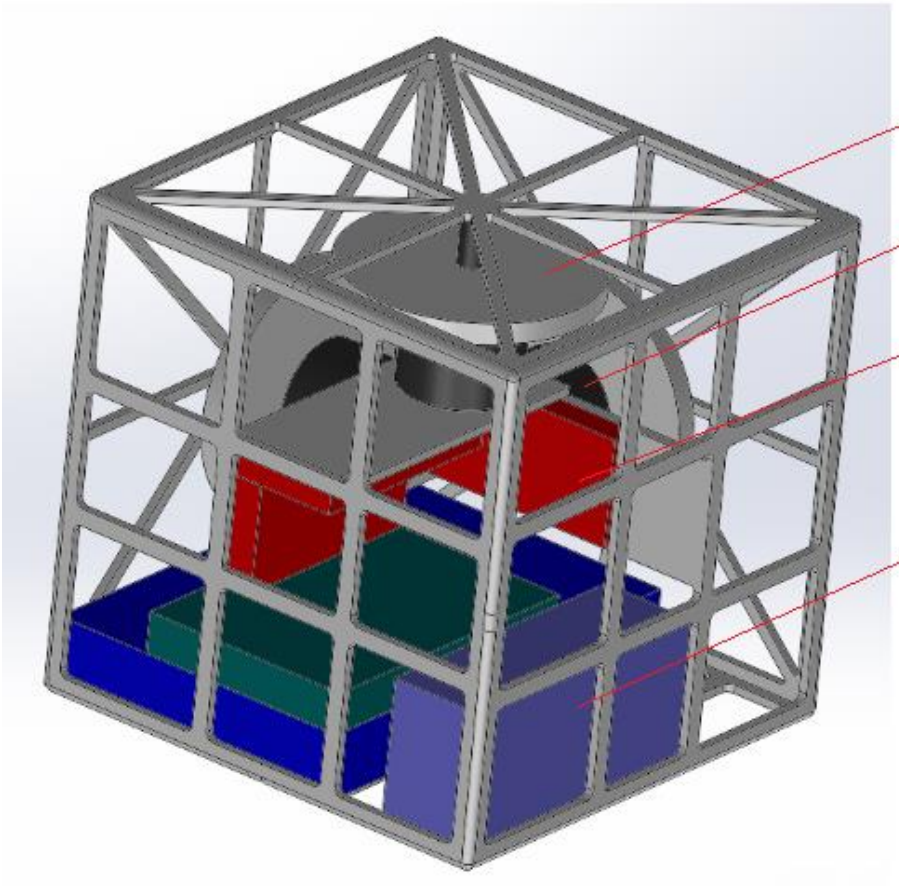
$$\Sigma M_o = \dot{H}_o$$

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

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

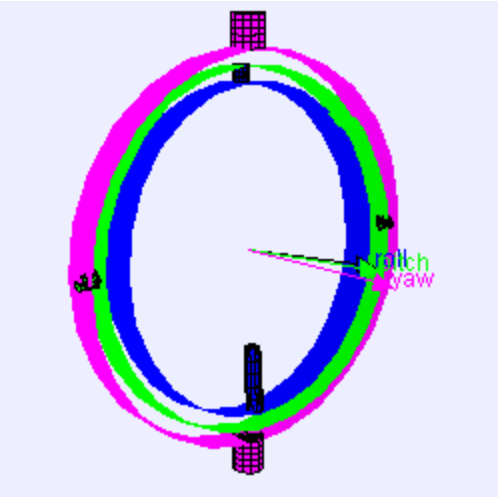
- Reorienting in mid air is possible due to internal moments

Original Design

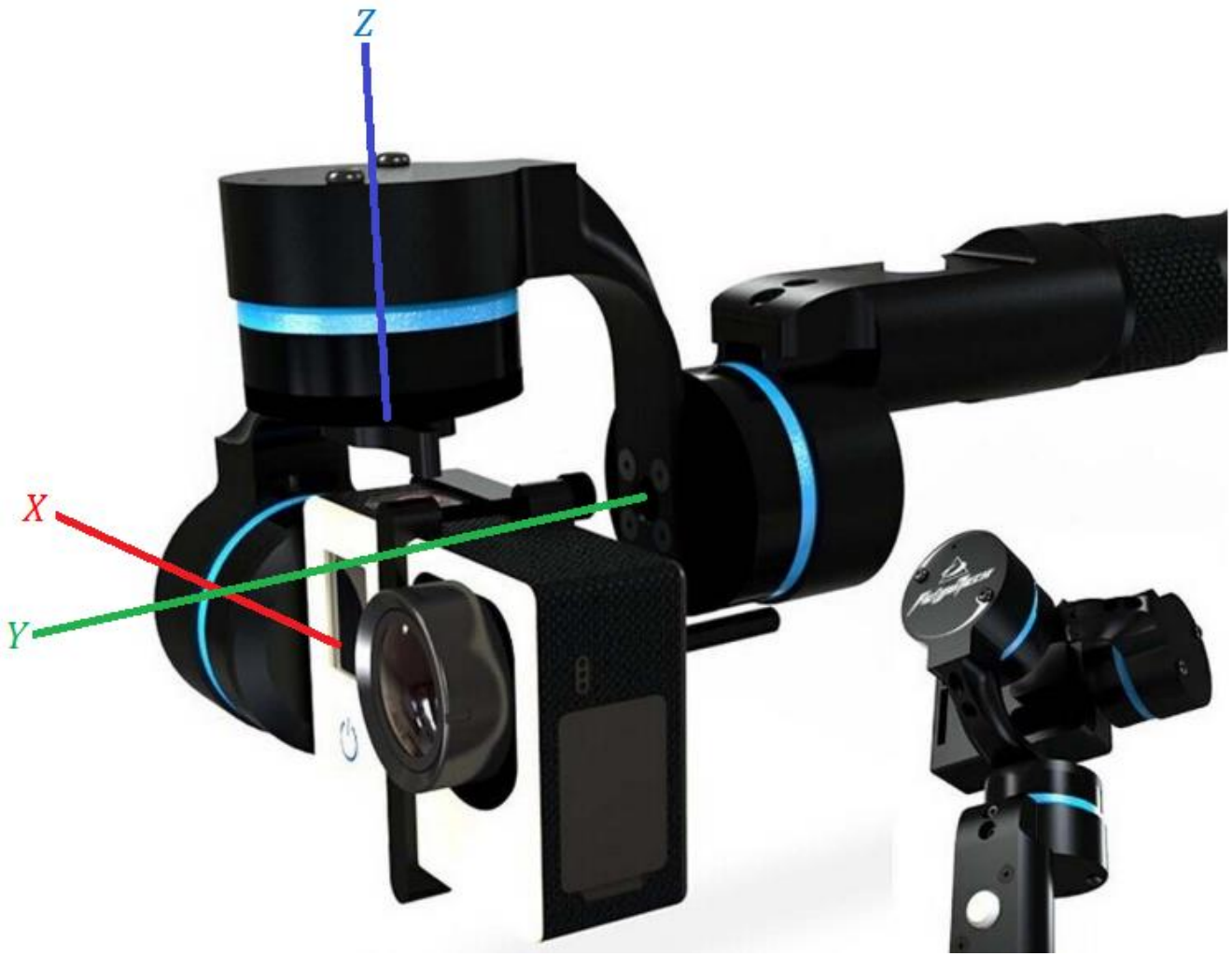


Follow-Up Motivation

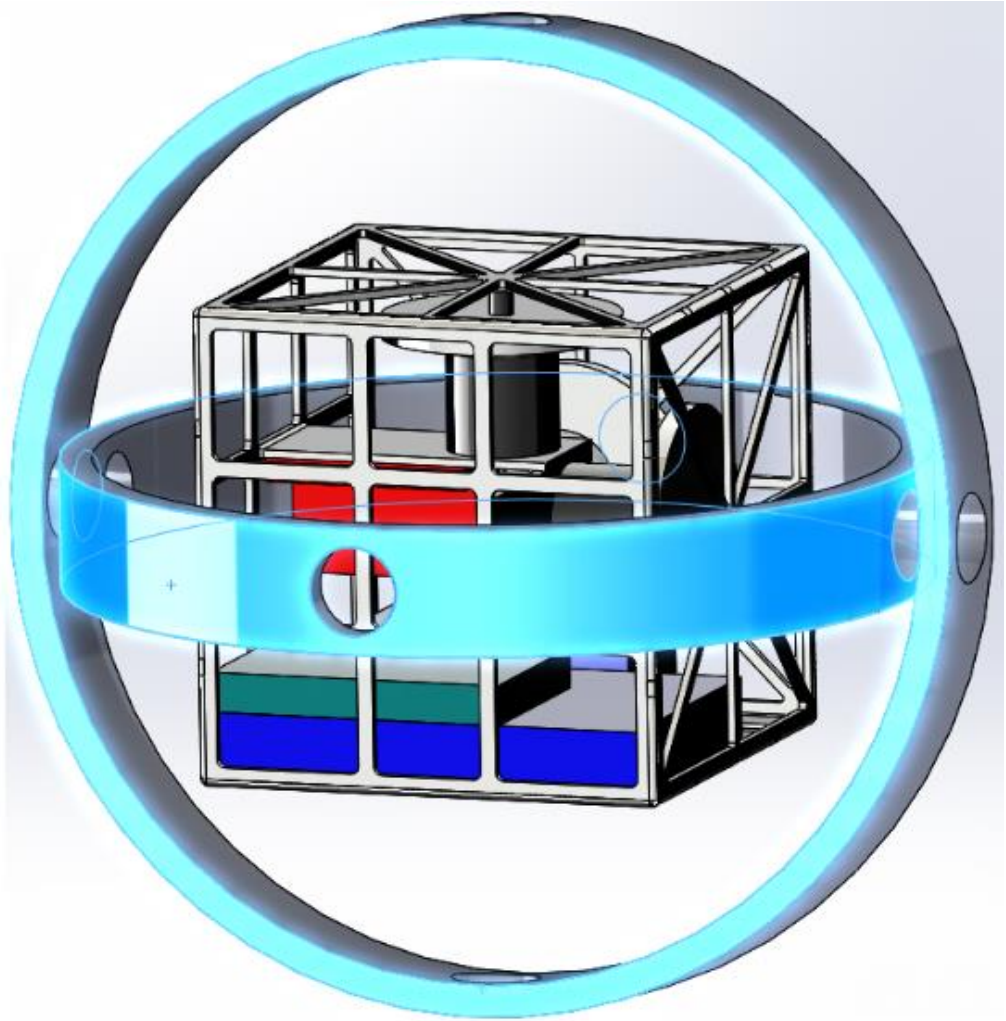
- Proper orientation is significant for cameras
- Gimbals implemented to provide 3 DoF
- Camera can rotate to track an object of interest as it moves without the need to follow it ourselves
- Limited by the need to be attached to a base for relative rotation



Camera on 3-Axis Gimbal



New Design with Gimbal

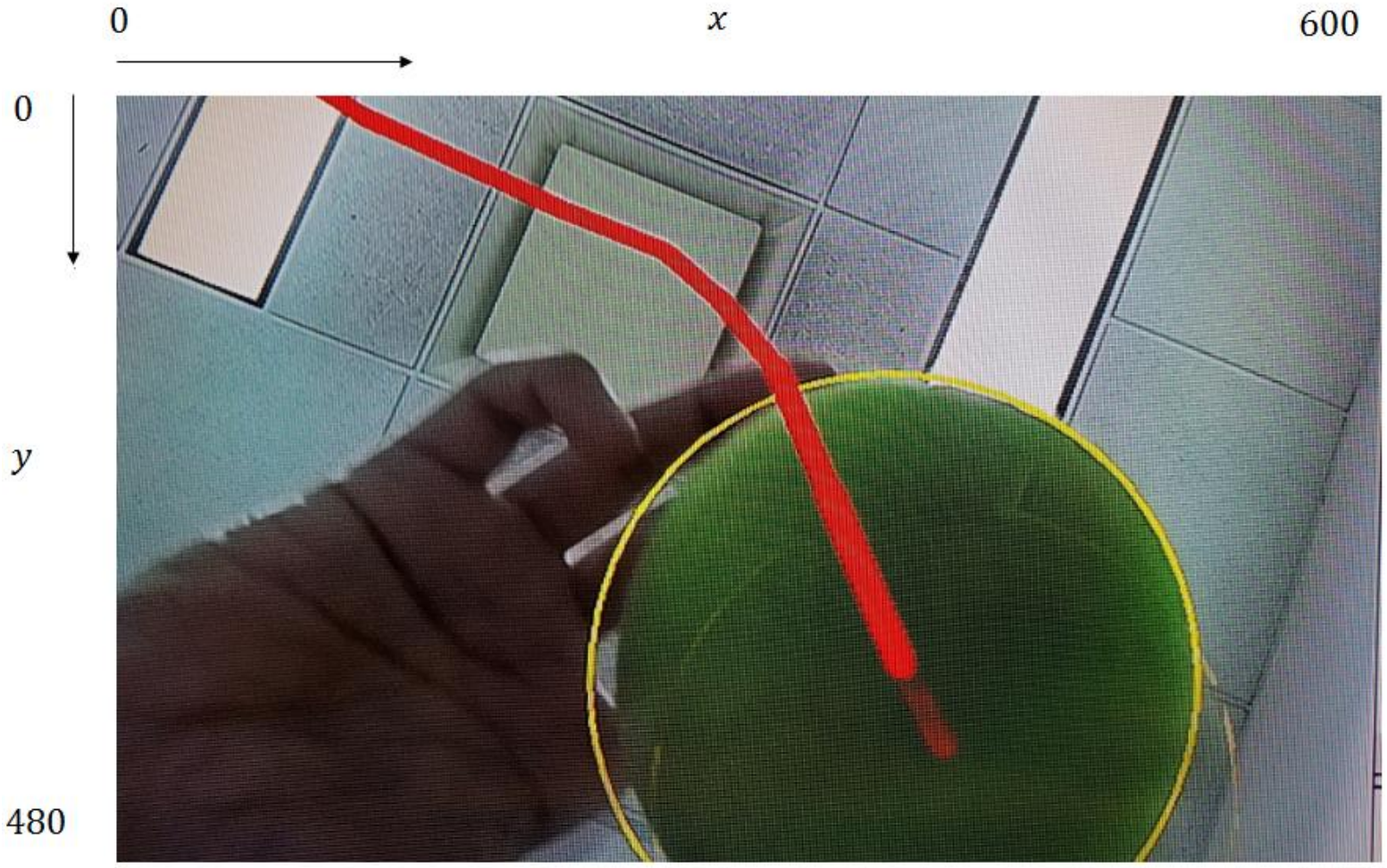


Gimbal Design

Project Plan

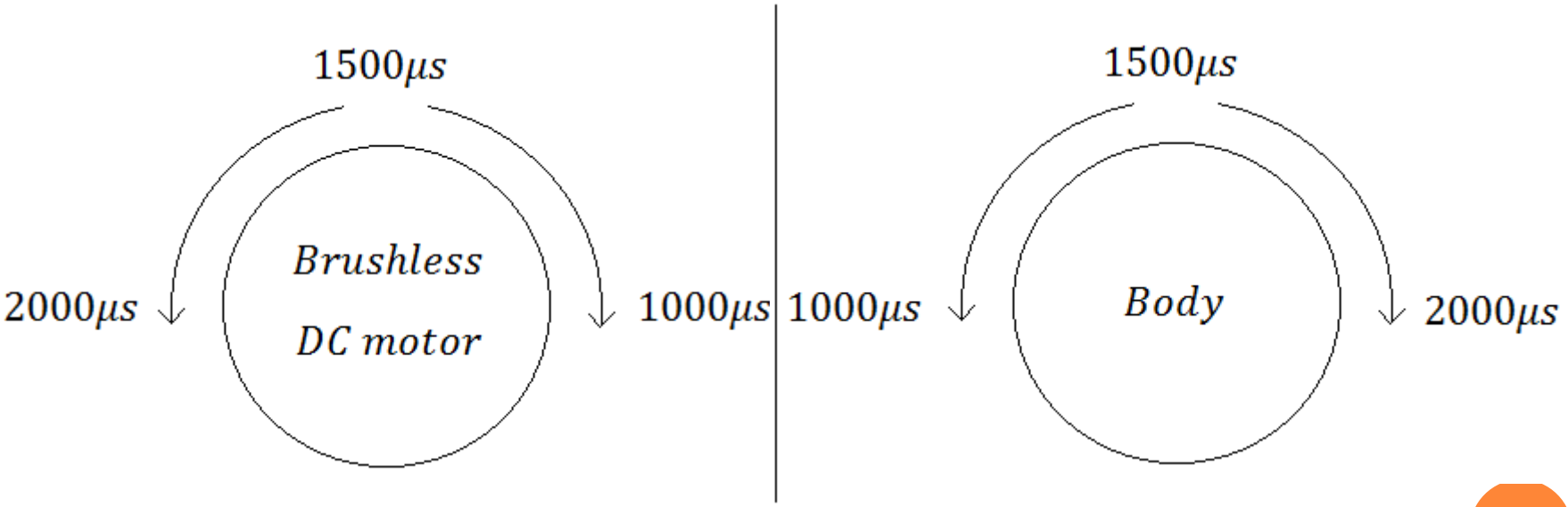
- Design 3 DoF camera without the need to be mounted to a base (Gimbal for demonstration purpose)
- Arduino Uno and Raspberry Pi B
- Track Green ball with Pi Camera
- Sends data to Arduino to actuate motors
- Control Algorithm based on coordinate of the center of the ball relative to the captured image's center

Pi Camera with Ball Tracking

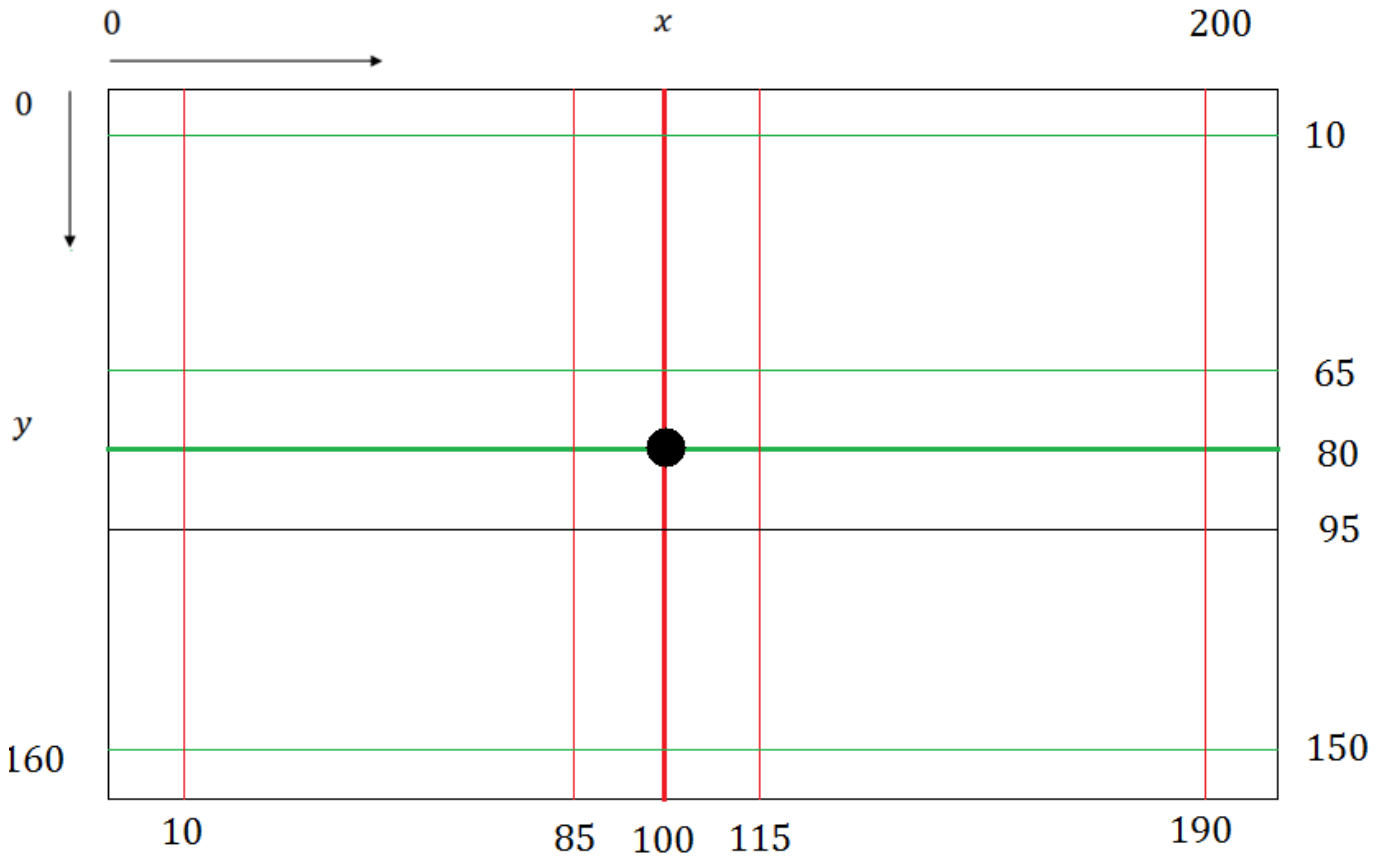


Control Algorithm

- Motors controlled with values from 1000µs - 2000µs
- Angular Velocity Control of motors and body

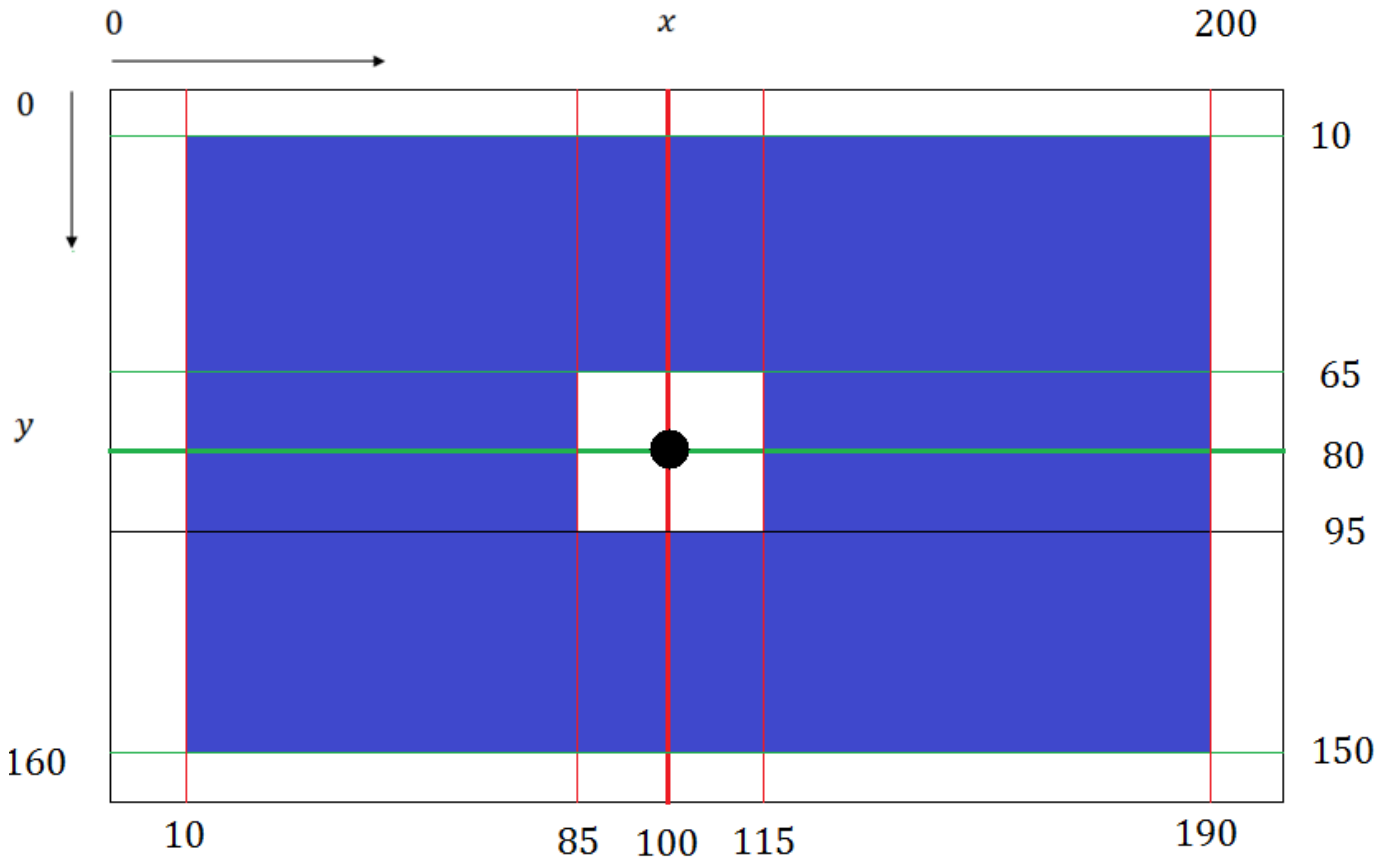


Control Algorithm



- Red/Green lines are thresholds used for control

Control Algorithm



- Blue: Regions to actuate motors

Arduino Code

```
#include "math.h"
#include <Servo.h>

Servo mZ;
Servo mY;
int Yval = 1500;
int PwmY;
int Zval = 1500;
int PwmZ;

void setup() {
  mZ.attach(10);
  mY.attach(11);
  delay(5000);
}
```

```
void loop() {
  if(Zval<85)
  {
    PwmZ=1400-(85-Zval)*4.0/3;
    mZ.write(PwmZ);
    delay(5);
  }
  else if (Zval>115)
  {
    PwmZ=1600-(115-Zval)*4.0/3;
    mZ.write(PwmZ);
    delay(5);
  }
  else
  {
    mZ.write(1500);
  }
```

```
  if(Yval<65)
  {
    PwmY=1400-(65-Yval)*100.0/55;
    mY.write(PwmY);
    delay(5);
  }
  else if (Yval>115)
  {
    PwmY=1600-(95-Yval)*100.0/55;
    mY.write(PwmY);
    delay(5);
  }
  else
  {
    mY.write(1500);
  }
}
```

RasPi Code

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy as np
import argparse
import imutils
from collections import deque
import serial
import struct

ser=serial.Serial('/dev/ttyACM1',9600)
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v",
    help="path to the (optional) video file")
ap.add_argument("-b", "--buffer", type=int, default=64,
    help="max buffer size")
args = vars(ap.parse_args())

# define the lower and upper boundaries of the "green"
# ball in the HSV color space, then initialize the
# list of tracked points
greenLower = (29, 86, 6)
greenUpper = (64, 255, 255)
pts = deque(maxlen=args["buffer"])
```

RasPi Code

```
camera=PiCamera()
camera.resolution=(640,480)
camera.framrate=32
rawCapture=PiRGBArray(camera, size=(640,480))

time.sleep(0.1)

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    frame=frame.array
    #cv2.imshow("Frame",image)
# Start Image processing code

        #frame = imutils.resize(frame, width=600)
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # construct a mask for the color "green", then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, greenLower, greenUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
```

RasPi Code

```
# find contours in the mask and initialize the current
# (x, y) center of the ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None

# only proceed if at least one contour was found
if len(cnts) > 0:
    # find the largest contour in the mask, then use
    # it to compute the minimum enclosing circle and
    # centroid
    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
    new_center= (center[0]/3, center[1]/3)

    new_center= ('%0*d' % (3,new_center[0]), '%0*d' % (3,new_center[1]))
    print new_center
    message1=str(10)
    message2=str(10)
    print message1
    ser.write(message1)
            ser.write(message2)
```

RasPi Code

```
        # only proceed if the radius meets a minimum size
        if radius > 10:
            # draw the circle and centroid on the frame,
            # then update the list of tracked points
            cv2.circle(frame, (int(x), int(y)), int(radius),
                (0, 255, 255), 2)
            cv2.circle(frame, center, 5, (0, 0, 255), -1)

    # update the points queue
    pts.appendleft(center)

    # loop over the set of tracked points
    for i in xrange(1, len(pts)):
        # if either of the tracked points are None, ignore
        # them
        if pts[i - 1] is None or pts[i] is None:
            continue

        # otherwise, compute the thickness of the line and
        # draw the connecting lines
        thickness = int(np.sqrt(args["buffer"] / float(i + 1)) * 2.5)
        cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

    # show the frame to our screen
    cv2.imshow("Frame", frame)
    rawCapture.truncate(0)
    key = cv2.waitKey(1) & 0xFF

# End of code
```

Complications

- Ball Bearings used to mount Gimbal rings still has “some” friction.
  - There is external moment applied to the cube, so angular momentum is not conserved.
- Very difficult to mount the cube’s center of mass exactly at the center of the assembly.
  - There is external moment of gravitational forces applied to the cube. It will tend to rotate until it reaches equilibrium