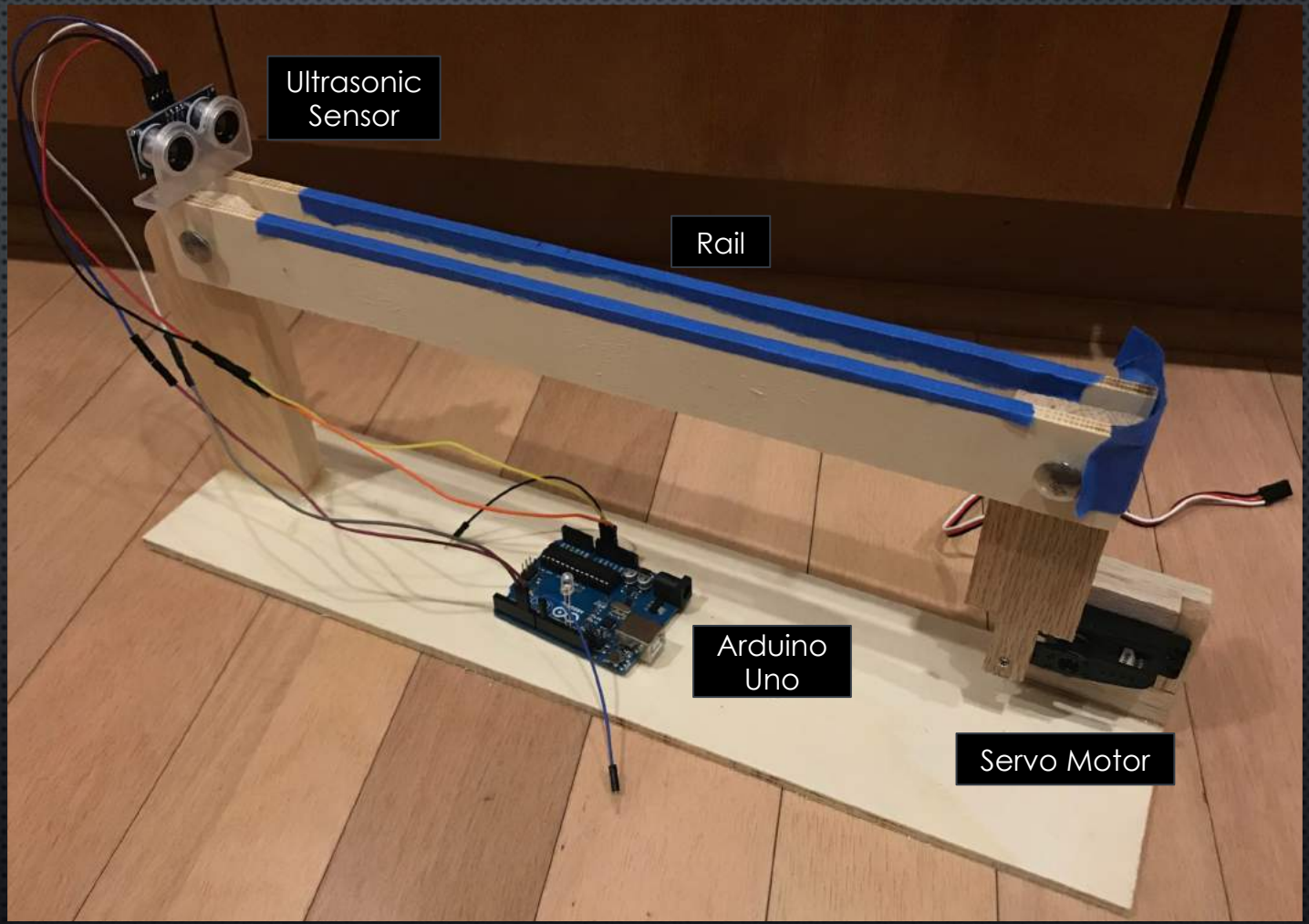
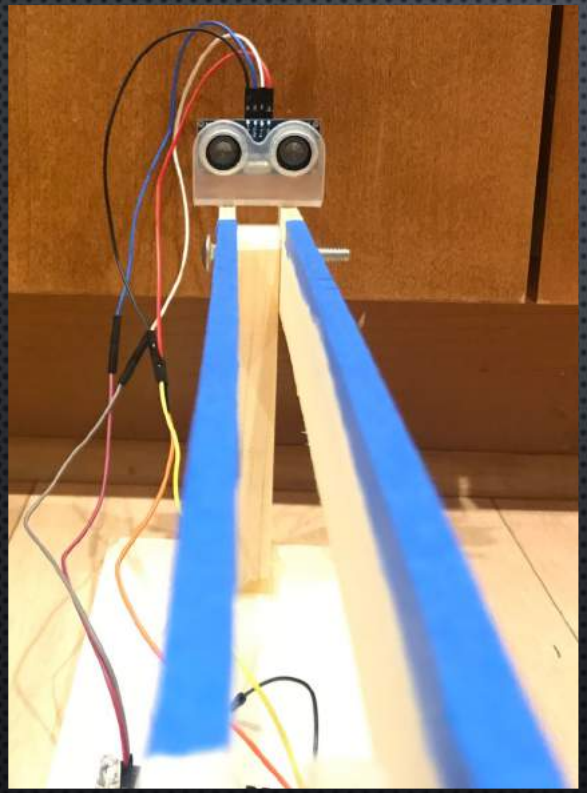
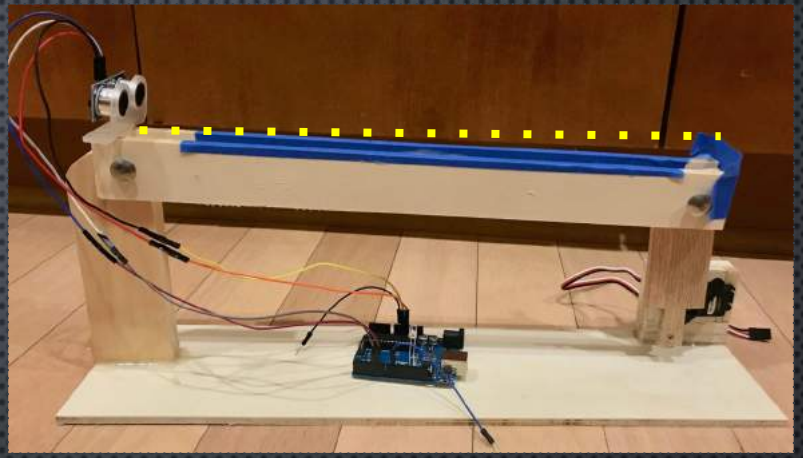
Components

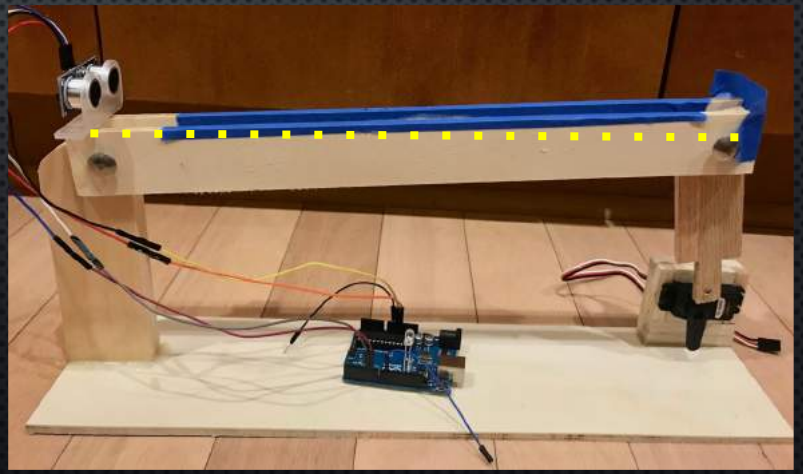
- Arduino Uno
- Servo Motor
- Ultrasonic Sensor
- 7.4V Li-Po Battery

THE SETUP







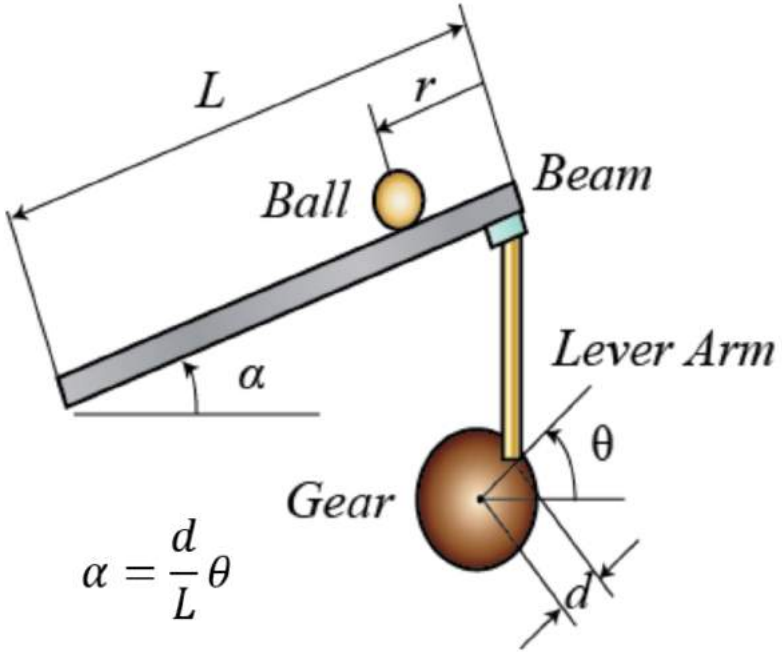


External factors

- Rail is not perfectly even (friction)
- The signals of ultrasonic sensor deflects unevenly @ longer distances
  - Fluctuations in reading
- Servo jerks causing SS-error
- Assume wheel as ball

MODEL

$$0 = \left(\frac{J}{R^2} + m\right)\ddot{r} + mg\sin\alpha - m\dot{r}\alpha^2$$



$$\alpha = \frac{d}{L}\theta$$

(m)	mass of the ball	kg
(R)	radius of the ball	m
(d)	lever arm offset	m
(g)	gravitational acceleration	9.8 m/s^2
(L)	length of the beam	m
(J)	ball's moment of inertia	kg.m^2
(r)	ball position coordinate	
(alpha)	beam angle coordinate	
(theta)	servo gear angle	

TRANSFER FUNCTION

$$0 = \left(\frac{J}{R^2} + m\right)\ddot{r} + mg\sin\alpha - m\dot{r}\alpha^2$$

Linearize equation about $\alpha = 0$, *gives us the approximation*:

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\alpha$$

The equation relates the beam angle to the angle of gear and can be approximated as:

$$\alpha = \frac{d}{L}\theta$$

Substituting this equation:

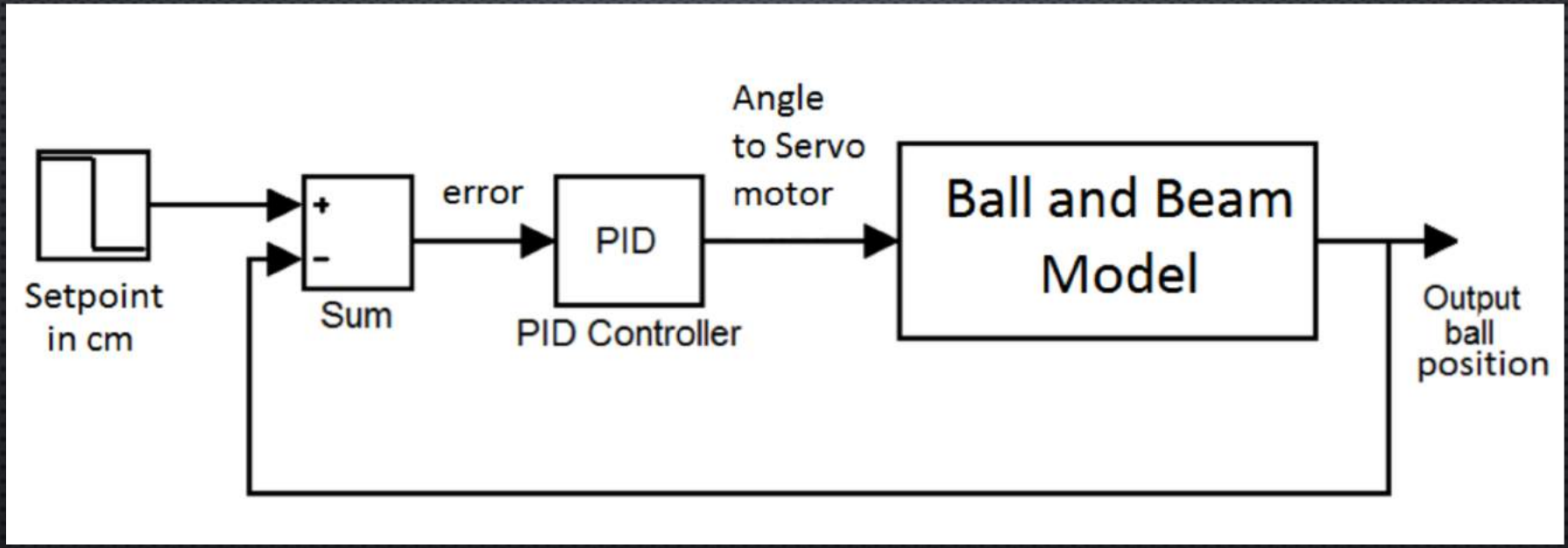
$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\frac{d}{L}\theta$$

Apply Laplace to get Transfer Function:

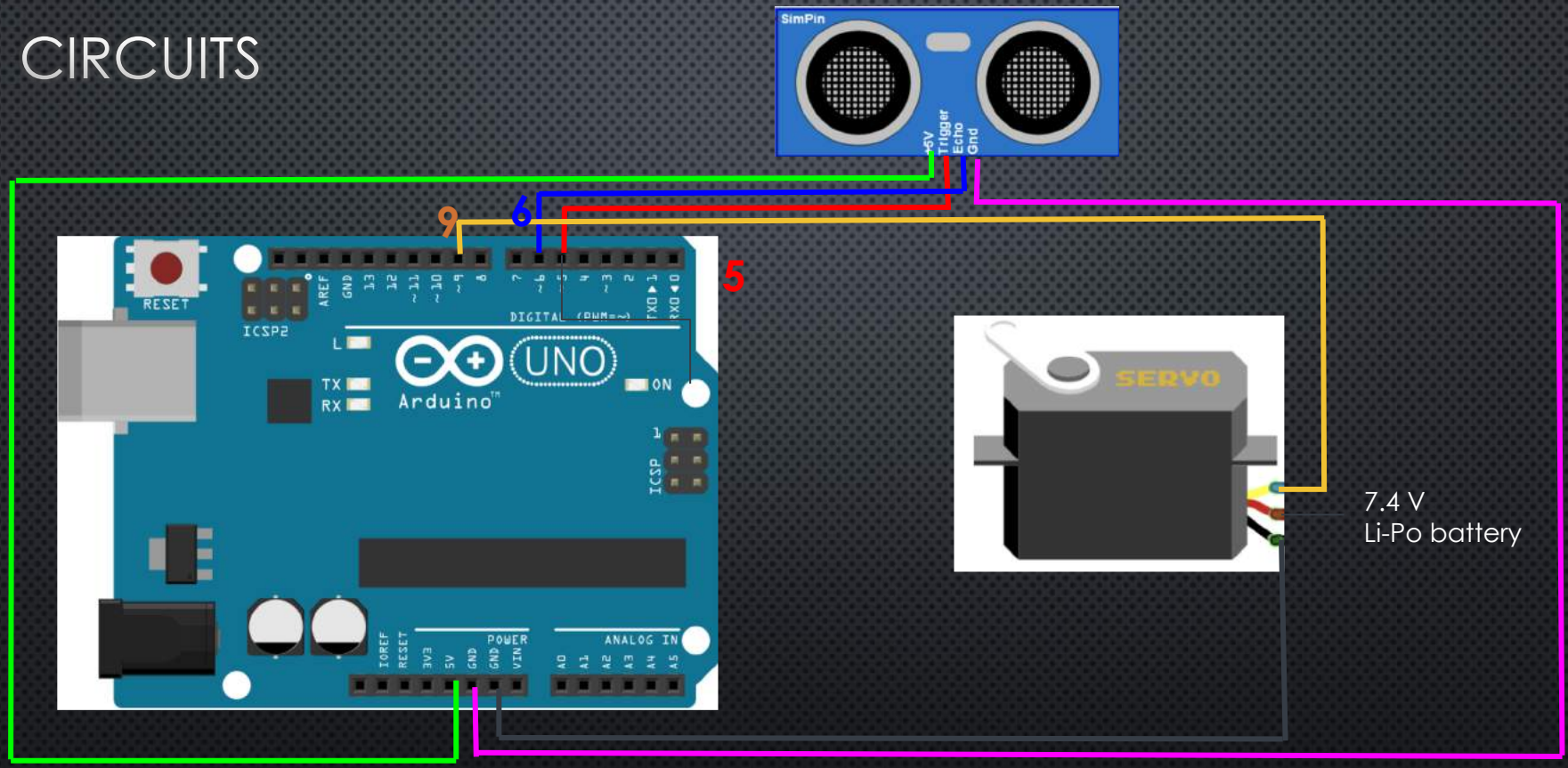
$$\frac{R(s)}{\Theta(s)} = \frac{mgd}{Ls^2\left(\frac{J}{R^2} + m\right)}$$

$$G(s) = \frac{0.4671}{s^2}$$

BLOCK DIAGRAM



CIRCUITS


SimPin
+5V
Trigger
Echo
Gnd
9
6
5
RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN
SERVO
7.4 V
Li-Po battery

CODE: SETUP

```
#include<Servo.h>
#include<PID_v1.h>


const int servoPin = 9;                                          //Servo Pin

float Kp_low = 2.5;                                              //Low Proportional Gain
float Ki_low = 0.2;                                              //Low Integral Gain
float Kd_low = 0.5;                                              //Low Derivative Gain

float Kp_high = 5;                                               //Aggressive Proportional Gain
float Ki_high = 0;                                               //Aggressive Integral Gain
float Kd_high = 2.5;                                             //Aggressive Derivative Gain

double Setpoint, Input, Output, ServoOutput;
PID myPID(&Input, &Output, &Setpoint, Kp_low, Ki_low, Kd_low, DIRECT);  //Initialize PID object, which is in the class PID.

Servo myServo;                                                   //Initialize Servo.


void setup() {

  Serial.begin(9600);                                            //Begin Serial
  myServo.attach(servoPin);                                      //Attach Servo

  Input = readPosition();                                        //Calls function readPosition() and sets the balls
                                                                 //  position as the input to the PID algorithm


  myPID.SetMode(AUTOMATIC);                                      //Set PID object myPID to AUTOMATIC
  myPID.SetOutputLimits(-40,40);                                 //Set Output limits to -40 and 40 degrees.
}
```

Adaptive PID gains

CODE: LOOP

```
void loop()
{

  Setpoint = 20;
  Input = readPosition();

  double gap = abs(Setpoint-Input);

  if (gap < 2) {
    myPID.SetTunings(Kp_low, Ki_low, Kd_low);
  } else {
    myPID.SetTunings(Kp_high, Ki_high, Kd_high);
  }

  myPID.Compute();                          //computes Output in range of -40 to 40 degrees

  ServoOutput = 75 + Output;                // 75 degrees is my horizontal
  myServo.write(ServoOutput);               //Writes value of Output to servo
}
```

Desired Position → **Actual position**

CODE: FUNCTION

Ultrasonic sensor distance calculation

```
float readPosition() {
  const int trigPin = 5;
  const int echoPin = 6;
  long duration, distance;

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Clear trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Set trigPin on HIGH for 10 microsecs
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads echoPin and returns sound wave travel in microsecs
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance
  distance = duration*0.034/2;

  if(distance > 30)      // 30 cm is the maximum position for the ball
  {distance = 30;}

  Serial.println(distance);

  return distance;                                    //Returns distance value in cm.
}
```

Observed Response

- Steady-State error of (0 - 1 cm)
- 2% Settling time ~ 2s