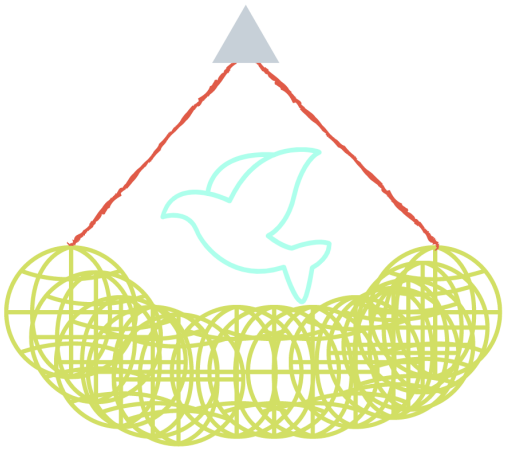


PENDULUM+

Team Riddler: Technical Manual

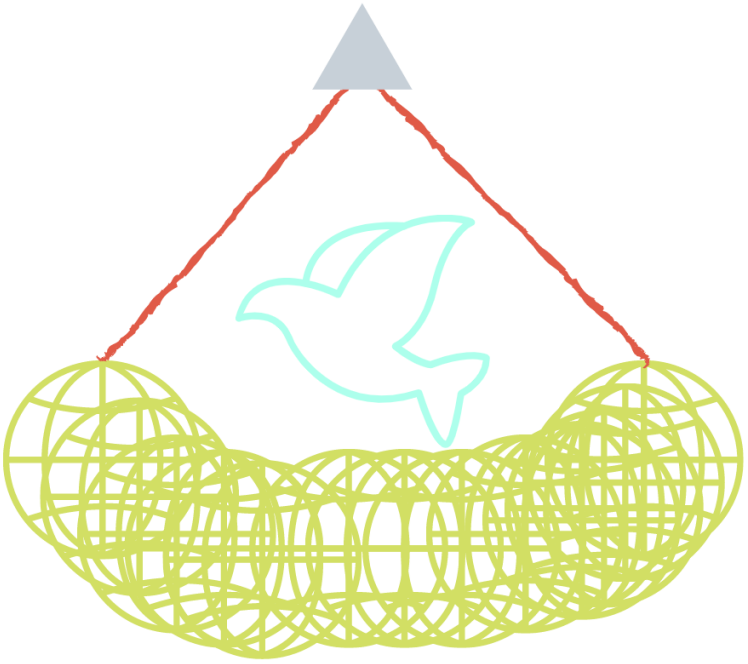
Pendulum +

Electronics Project

December 2019

Do you suspect deep down that your students are struggling to understand natural frequencies?

Is engineering in general really dodging balls at them?



Introducing PENDULUM+

...the first-ever teaching aid device that adjusts the string length to change natural frequency and achieve resonance!

Buy now, and let this pendulum motivate failing students to learn by doing!

1 (800) 736 - 3858

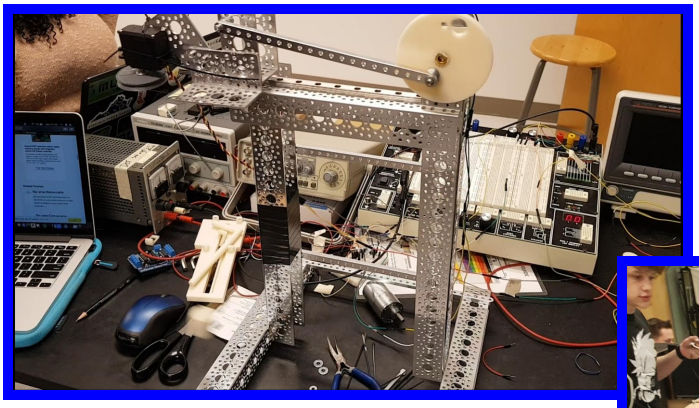
Our Story

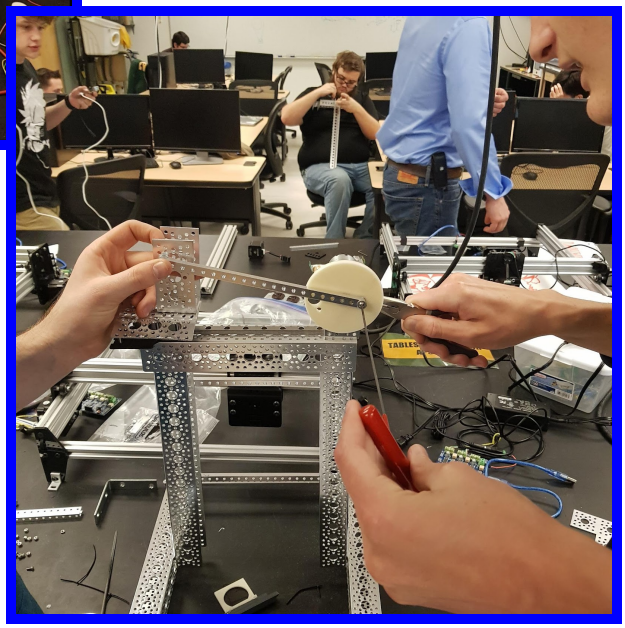
At Pendulum +, we were not satisfied with your average ball and rope for teaching students about natural frequencies. We desired to mechanically drive a pendulum with adjustable lengths to vary the natural frequency. This goal oscillates a mass at different user selected frequencies to the highest accuracy. We strove to better illustrate how resonance, and natural frequencies in pendulums are dictated to students. The pendulum was built to run simply so students could easily understand how it functions. The DC motor driving the oscillations is based on simple solid body rotations. Changes in height are found with the rotation of a wheel at a specified radian amount. Our teaching aid helps better explain this key physics concept allowing for better learning and understandability. We even included a sensor to calculate the deviation from the set frequency and then adjust to get as close as possible to the selected frequency.

We engineered the frame from sturdy and light aluminum beams with predrilled holes for easy assembly, corrosion resistant, and aerospace quality durability. The sliding mechanism features precision milled and fitted ball bearings for smooth, low friction operation. The pendulum hangs on sturdy 3D printed parts connected to a servo for a customized frequency operational range. Fishing line comprises the string holding the mass for easy repairability and near massless performance.

For easy operation a Raspberry Pi 4 acts as the brains. A python code was developed to efficiently control the driving force, height, and adjust for error. As a ready to run, plug and play, unit no 00co00ding experience is needed to operate our pendulum.

We hope you enjoy the optional frequency changes, electronics, and fun factor of our pendulum, For students who struggle with physics we heavily recommend this necessary tool that can teach engineering, electronics, and how pendulums function.





USER'S MANUAL

TABLE OF CONTENTS

pg. 5: General Design Overview and Description

pg. 9: Design Specs

pg 11: Tutorial
pg 11: Intended Use
pg 11: Installation Instructions
pg 11: Operational Procedures
pg 11: Troubleshooting
pg 12: Safety Warnings
pg 12: Maintenance Information

pg 14: Appendix
pg 14: Team Members & Roles
pg 15: Features & Accessories
pg 16: Components
pg 19: CAD Drawings
pg 20: Circuit Diagrams
pg 29: Finding Duty Cycle to Frequency Conversion
pg 30: Software Programs

General Design Overview and Description

The logic behind Pendulum+ stems from the following equation for a pendulum's natural frequency where *g* is the gravity of 9.81, and *l* is the length of the pendulum string.

$$\omega_0 = \sqrt{\frac{g}{l}} \tag{1}$$

In describing to an average student how we arrive at the equation above, we start by considering the mass *m* suspended from a string of length *l*. A simple pendulum is a machine that lets the mass swing freely from side to side within a vertical plane. In Figure 1, is the angle of the string counterclockwise from the vertical. The second derivative of θ, $\frac{d^2\theta}{dt^2}$, is the variable that is subject to the angular motion of the pendulum where *I* is the moment of inertia of the mass and τ is the torque acting on the system. Mathematically,

$$I \cdot \frac{d^2\theta}{dt^2} = \tau \tag{2}$$

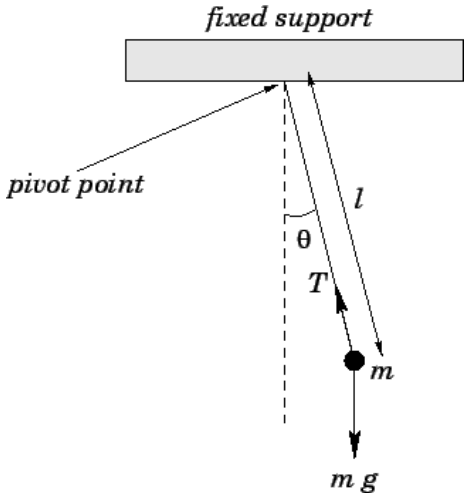


Figure 1. Setup diagram of the simple pendulum from
http://farside.ph.utexas.edu/teaching/301/lectures/node140.html

Since the mass *m* is situated at a distance *l* from the axis of rotation attached to the center of the fixed support (i.e., pivot point), the moment of inertia would be,

$$I = ml^2 \tag{3}$$

The main forces that are acting on the mass are the gravitational force *mg* and the tension *T*. The line of action of mg passes the distance $l \cdot sin\,\theta$ from the pivot point. Since the magnitude of *mg* is downward, it will be negative, and the magnitude of τ shall be,

$$\tau = -mgl \cdot sin\theta \tag{4}$$

Substituting *I* and τ into Equation (2) gives us,

$$ml^2 \cdot \frac{d^2\theta}{dt^2} = -mgl \cdot sin\theta$$

$$l \cdot \frac{d^2\theta}{dt^2} = -g \cdot sin\theta \tag{5}$$

Assume the relatively smallest deviation from the equilibrium state of . If $\theta < 6^\circ$, we can approximate that,

$$sin\theta \simeq \theta \tag{6}$$

Substituting $sin\,\theta$ into Equation (5) gives us the following equation for simple harmonic motion of the simple pendulum,

$$l \cdot \frac{d^2\theta}{dt^2} = -g \cdot \theta \tag{7}$$

Equation (7) is a reference to Newton's second law of motion, which via Hooke's law provides the restoring force f and force constant k of the spring. Mathematically,

$$f = -kx$$

$$m \cdot \frac{d^2x}{dt^2} = -kx \tag{8}$$

You may find it helpful to know the following steady-state solutions to Equation (8), where a, ω, and φ are constants when

$$x = a \cdot cos(\omega t - \varphi) \tag{9}$$

$$\frac{d^2x}{dt^2} = -\omega^2 a \cdot cos(\omega t - \varphi) \tag{10}$$

Substituting Equations (9) and (10) into Equation (8) will give us the angular equation for simple harmonic motion.

$$-m\omega^2 a \cdot cos(\omega t - \varphi) = -ka \cdot cos(\omega t - \varphi)$$

$$m\omega^2 = k$$

$$\omega^2 = \sqrt{\frac{k}{m}} \tag{11}$$

Referring back to Equation (7), we can suggest that its solutions shall be the following:

$$\theta = a \cdot cos(\omega t - \varphi) \tag{12}$$

$$\frac{d^2\theta}{dt^2} = -\omega_0{}^2 a \cdot cos(\omega t - \varphi) \tag{13}$$

Substituting Equations (12) and (13) into Equation (7) will give us the angular natural frequency of small amplitude oscillations of a single pendulum, which is essentially Equation (1). Note that this frequency depends on the length and gravity of the pendulum, and it is independent of the mass and swing amplitude.

$$-l\omega_0{}^2 a \cdot cos(\omega t - \varphi) = -g \cdot a \cdot cos(\omega t - \varphi)$$

$$l\omega_0^2 = g$$

$$\omega_0 = \sqrt{\frac{g}{l}}$$

In addition to the natural frequency, the servo motor wheel can change the length of the string. Determining the new length derives from a mathematical concept of arc length. In Figure 2, θ is the angle of rotation in radians and r is the radius of the wheel. To determine arc length s, we use the following,

$$s = \theta r \tag{14}$$

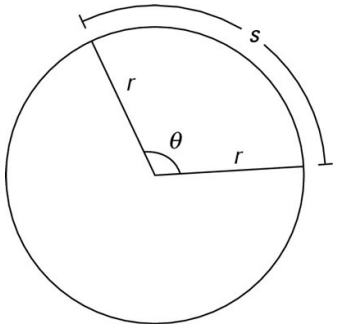


Figure 2. Arc circle diagram from
https://www.dummies.com/education/math/calculus/how-to-determine-the-length-of-an-arc/

It is helpful to note that unwinding the string decreases θ whereas winding it back increases θ. Assuming that s is the string wrapped around the servo wheel, we can add s to the previous length L_1 of the hanging string to determine a new string length L_2. That is,

$$L_2 = L_1 + \theta r \tag{15}$$

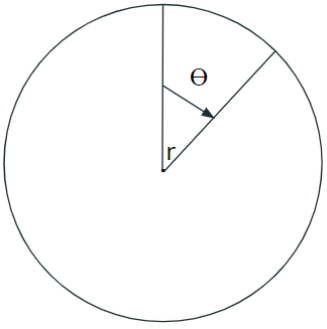


Figure 3. Improved diagram for servo wheel rotation

To learn more, visit the following websites:

1. The simple pendulum, The University of Texas at Austin:
   http://farside.ph.utexas.edu/teaching/301/lectures/node140.html

2. Simple harmonic motion, The University of Texas at Austin: http://farside.ph.utexas.edu/teaching/301/lectures/node138.html#eshm
3. Measurement of angles, Dave's Short Trig Course, Clark University: https://www2.clarku.edu/faculty/djoyce/trig/

Design Specs

Aesthetics	Sleek, Sturdy, and Functional	The stand and control box of the product are made from aluminum alloy. The design is compact and contains no unnecessary components, meaning it is easy to use and will stand the test of time.
Target Audience	Students: Middle school through college	Pendulum + is intended for students learning about simple harmonic motion.
Function	To drive a pendulum at a range of frequencies (0.91-1.06 Hz)	The Pendulum + product is an electrically driven pendulum that can operate at a range of predetermined frequencies.
Materials	Stepper Motor, Servo Motor, Raspberry Pi 4, and Aluminum Stand	These are the main components necessary to create the framework for a driven pendulum. These are supplemented by other electrical components such as voltage amplifiers and MOSFET circuits.
Size and Weight	Requires a surface area of 45 cubic centimeters for safe use. Weighs 5 kg.	The Pendulum + product with the stand and control box weighs no more than 5 kg and should occupy a space large enough that the pendulum and swing freely with no obstruction.
Installation	No Installation	The product comes pre assembled and only requires a source of electricity to run

Maintenance	Minimal	Pendulum + should not be left plugged in for longer than 2 hours at a time. Make sure the string is untangled before using the product.

Tutorial

Intended use

Frequency matching and selection of a swinging mass:

See online video for step by step operation of Pendulum+

Installation instructions

- The product ships ready to operate. The frame, motors, and electronics are pre-assembled for out of the box operation.
- No installation is required.

Operational Procedure

When first opening the product, make sure all components are accounted for and visibly unbroken. To begin, place the pendulum stand on a solid and flat surface. Put the control box containing the Raspberry Pi next to the pendulum stand and plug it into a power source, such as a wall outlet. Once the controller is on, the screen will prompt the user for frequency within a certain range. Once you have inputted the desired frequency the pendulum will start to swing. The pendulum will continue to operate until you cancel the motion by pressing the ‘Stop’ button on the control box or you input a new frequency.

Troubleshooting

- The control box can become overloaded under heavy operation and fail to react to inputs, user controls, or clearing the pins. In the event of wacky operation,

failure of the controller to adjust the servo motor, lack of adjustment to frequency inputs, or freezing turn the control box off and unplug it. Allow the control box to cool for an hour until it is no longer hot to the touch. At this point plug the control box back in and restart the program.

- In the event that the cool off does not work, please call customer support.

Safety warnings

- Use product on a flat surface with minimal vibrations to limit interference while in operation
- Both the stand and control-box contain electronic components, keep away from fluids and other sources of electricity.
- The pendulum contains a swinging mass that could impact nearby objects or people, keep away from fragile objects and young children.
- Do not touch electronic components while in operations to avoid short circuits or shock
- The pendulum is a choking hazard for children under 3 years of age.
- String may become tangled and can strangulate small children or animals.

Maintenance information

- Repair information
    - If your Pendulum + has stopped working please call our customer support before attempting to take apart the product.
- Information on disposal of the product and packaging

    - Please recycle all packaging components that are not included with the stand or control box. If the whole product no longer works please follow local laws to safely dispose of the electronic components, which include both motors and the control box.
- Index
- Glossary
- Warranty information
    - Pendulum + has a 2 year warranty when the product is purchased. If an accident occurs which causes the pendulum to stop working, or the product simply fails to operate, the Pendulum + will be replaced with no extra charge to the customer. If the product is taken apart or tampered with, the warranty is voided and cannot be replaced.
- Contact details
    - Customer service is available 24/7 at 1 (800) 736 - 3858

Appendix

- **Bennett Atwater:** Acted as team leader to ensure the proper distribution of work and its accomplishment. Designed Fusion360 models for 3D printing. Handled code for interfacing Raspberry Pi with sensors. Compiled Python code for running the apparatus into a singular file. Acted as key coding reference for Team Riddler. Wired sensor circuitry.
- **Giles Corzine:** Built the aluminum stand for the pendulum and helped attach components of the project to said frame. Wired the non-inverting amplifier for the servo motor. Contributed to the tutorial and appendix sections for the user manual. Filmed and created the video highlighting the construction process of the project.

- **Martha Gizaw:** Debugged the Python code for the duty cycles and frequencies to observe changes in the pendulum string length and swing motion. Tested and delivered the string and other small-scale materials for constructing the pendulum. Co-performed a material cost analysis for each electronic component installed in the product. Determined how to best arrive at the appropriate formulas and solutions for running the pendulum. Ensured that the product, manual, and promotional materials are of satisfactory quality and are meeting their standards before going into the market.

- **Daniel Slyepichev:** Optimized height of pendulum. Acquired data for the natural frequency of a pendulum given its length. Arranged method for matching the frequency of the pendulum to the DC motor's duty cycle rotation rate. Assisted with mathematical code design. Created frequency graphs and circuit diagrams.
- **Jacob Wacht:** Worked on designing, manufacturing, and testing the Pendulum. Modeled Pendulum prototypes in Fusion 360, 3-D printed, and selected designs. Designed and built the DC motor drive circuit and servo motor to work with the Pi. Worked with the group to code the Pi to run both the Servo and DC motor at the same time while adjusting to the user inputted frequency. Coded to find the actual frequency the Pendulum was operating at and output the value to the user.

Features/accessories

- Unique design components
    - Dual motor action combining a DC motor for oscillations and stepper motor with PWM controls for changes in height to achieve maximum available natural frequencies of a pendulum and driving motion
    - Sturdy frame
        - light, adjustable aluminum rails
    - Smooth action
    - Error feedback and pendulum tracking

        - acoustic sensor
    - Adjusting to the error for accuracy and precision

Components

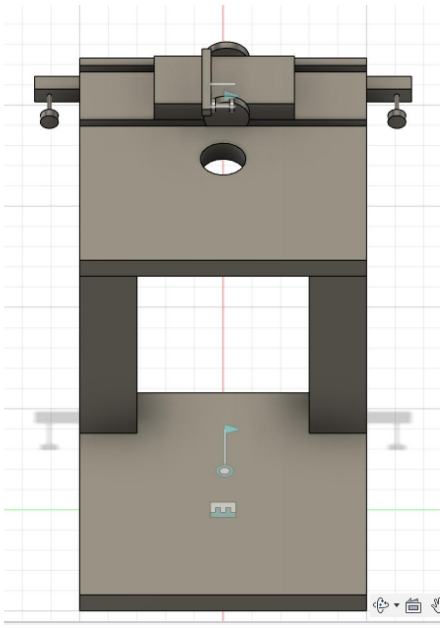
- **Assemblies**
    - Frame/stand
    - Rail system
        - accomplish smooth oscillations back and forth
- **Individual components**
    - Raspberry Pi 4
        - Controls all electronics, motors, and user interface
        - $45
    - High torque servo motor
        - Controls the height of the hanging mass
        - Hitec HS-645MG High-Torque 2BB Metal Gear Servo
        - $29.99
            - Vendor: Tower Hobbies (https://www.towerhobbies.com/cgi-bin/wti0001p?I=HRCM0927)
    - DC motor
        - Provides the driving force with a piston like motion
        - LMioEtool DC Gear Motor, High Torque Reversible Electric Geared Motor - with Eccentric Output Shaft Gearbox (12V/87RPM)

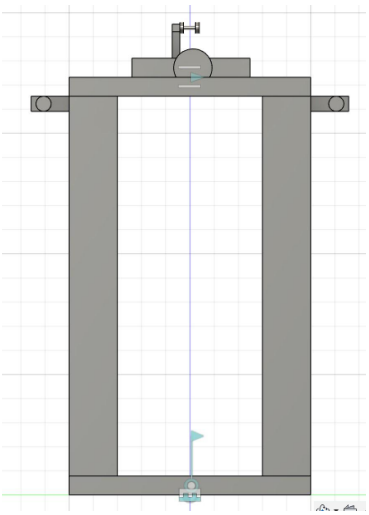
        - Source (amazon.com)
    - $13.99
- Thin Synthetic String
    - Holds the mass and act as a massless string for better results
    - ~$0.50
- Hanging Mass
    - Swing back and forth like a pendulum
    - 3D printed (cite design in appendix)
    - ~$1.50
- Acoustic sensor
    - Measure the actual frequency of the pendulum and adjust for error
    - HC-SR04 Ultrasonic Range Finder
    - $2.99
- frame components
    - 2x 12 inch actobotics aluminum rails ($9.99 each)
    - 2x 15 inch actobotics aluminum rails ($11.99 each)
    - 1x 9 inch actobotics aluminum rail ($7.99)
    - actobotics channel A bracket ($4.99)
    - Hub mount B 90 degrees ($4.49)
    - Dual channel flat bracket ($1.49)
    - Dual channel 90 degrees
    - Misc hardware ($6)

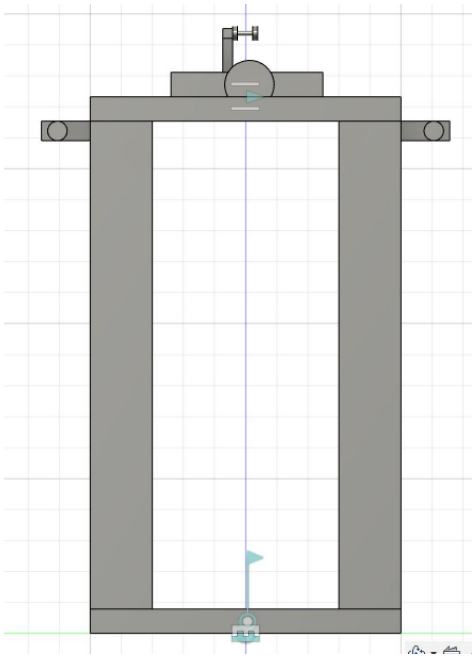
- Source for frame components: SparkFun (https://www.sparkfun.com/pages/Actobotics)

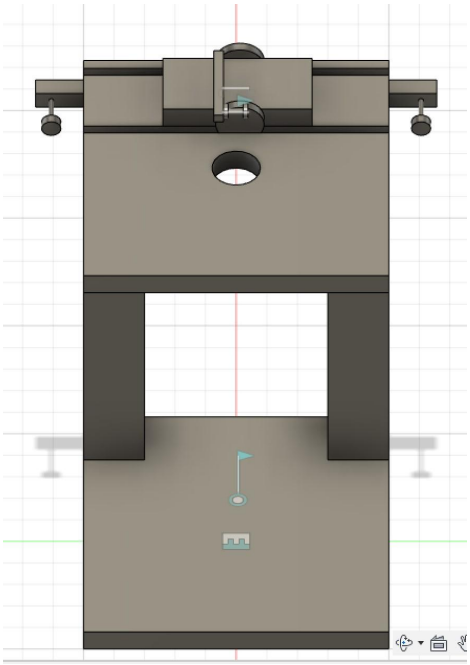
CAD drawings

Iteration 1:







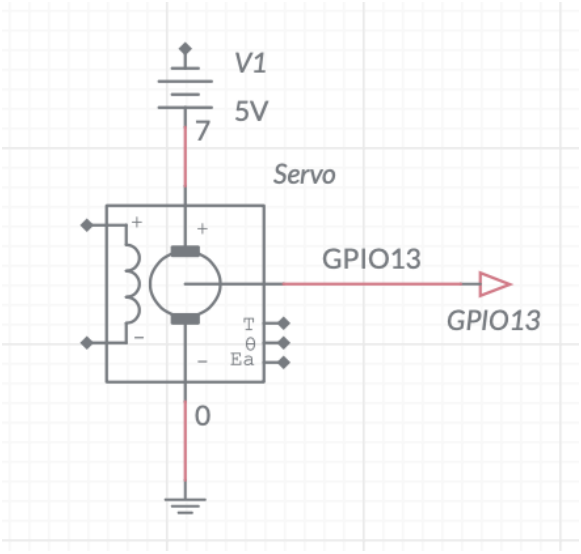


Iteration 1 included a cart oscillated on rails by two motors on the frame. The pendulum was suspended from the cart and would swing through the central hole.

Iteration 2: (include images of the final design)
The second iteration involved scrapping the most of the 3-D printed components for prefab aluminum frame pieces and a sliding rail assembly.

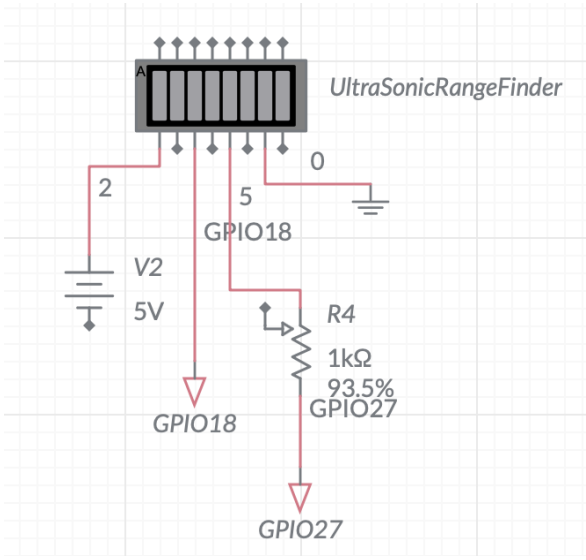
Circuit diagrams:

Servo Control Circuit:



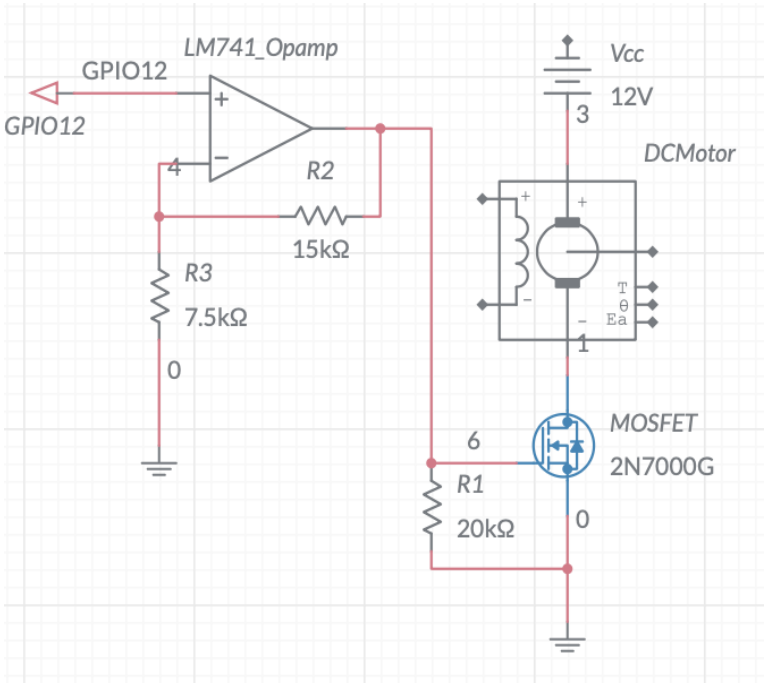
5v are input to the servo and it is grounded. Motion is controlled by the GPIO pin 13 from the Pi. A specific PWM signal is sent that tells the servo motor how far to rotate. The Servo is sensitive to 15 degree changes.

Pendulum Timing Control:



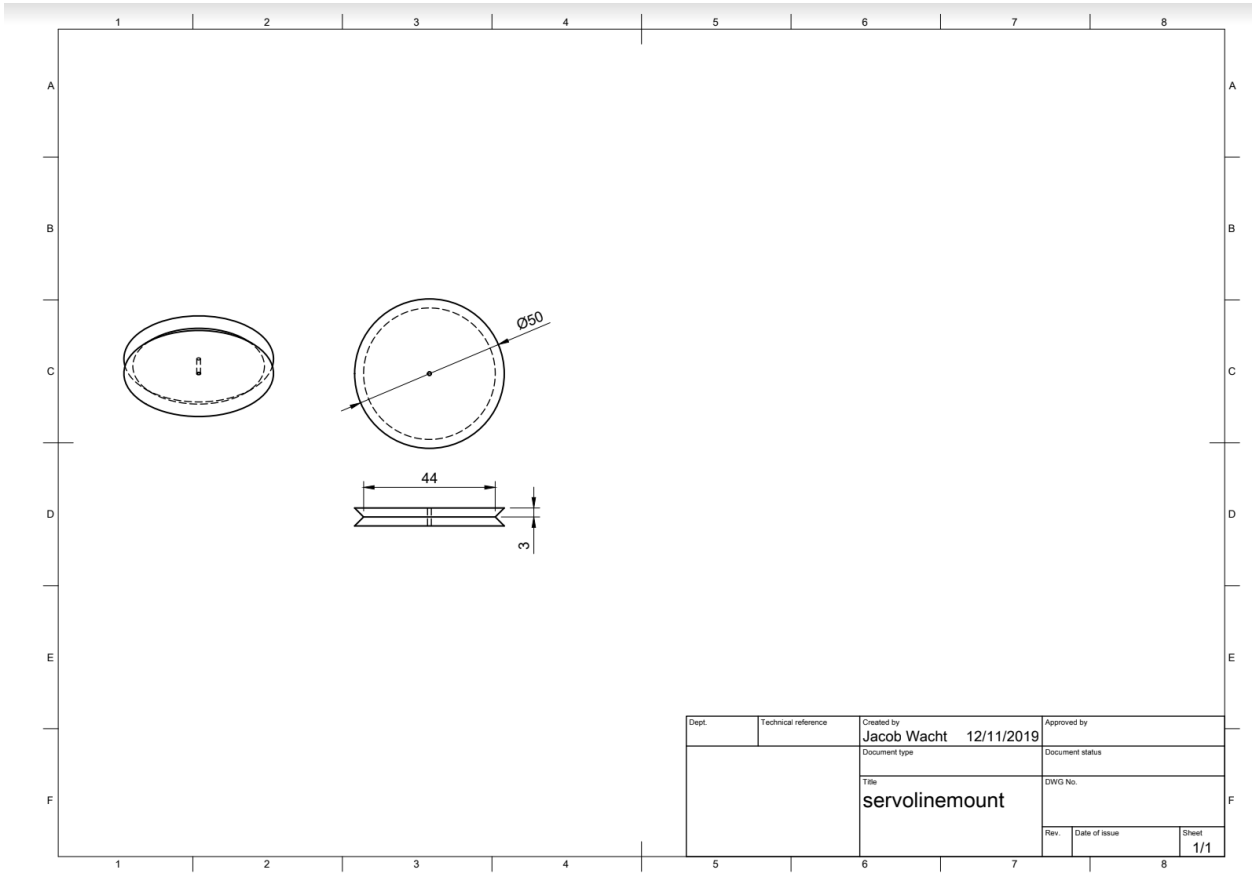
An acoustic sensor is wired to the Pi to record changes is distance at a set trigger rate. The changes in distance are filtered within the program to find the average experimental frequency.

DC motor control:

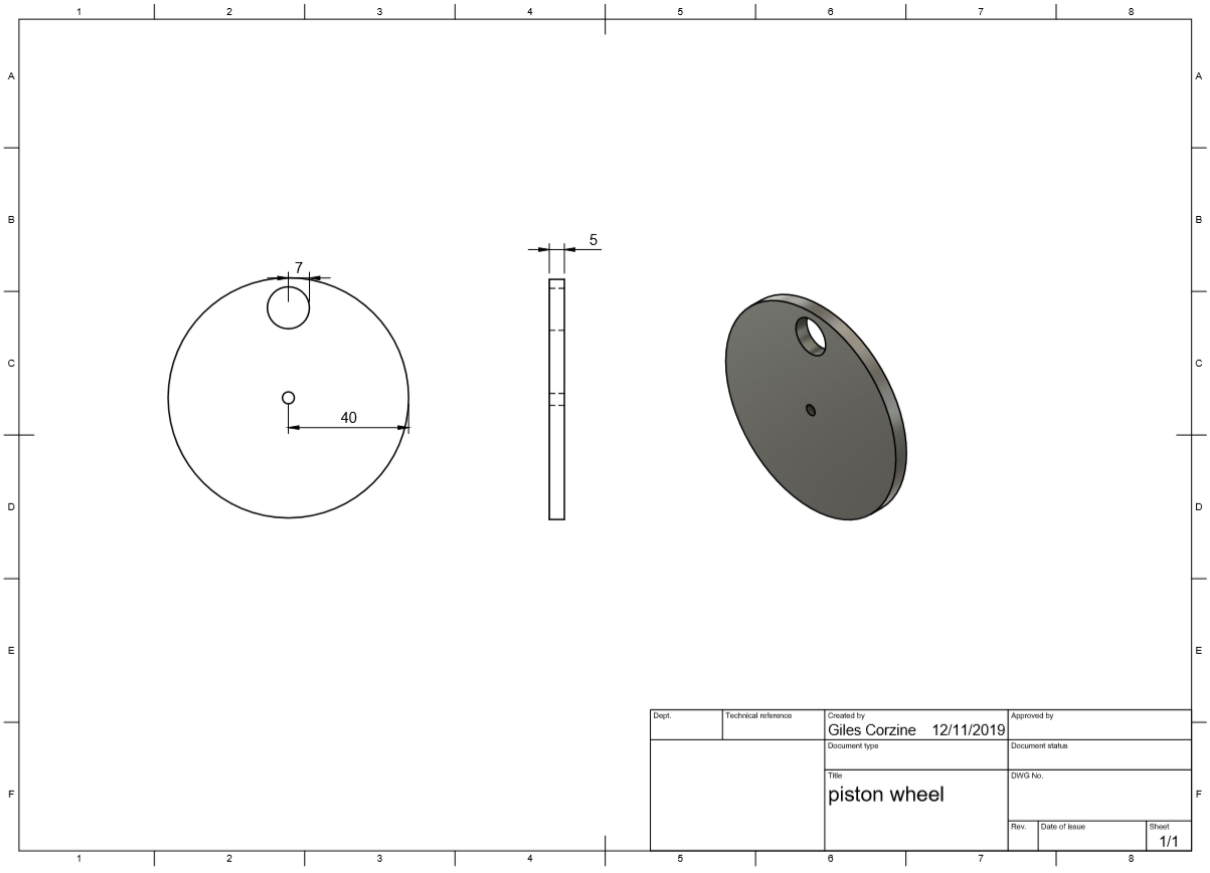


The Signal from the Pi is amplified by an op amp up to 5 volts for the MOSFET to allow current to be amplified and flow through the motor allowing for motor rotation. A PWM signal is sent at different pulse widths to allow the motor to run for different lengths resulting in a different rotational frequency hence driving frequency.

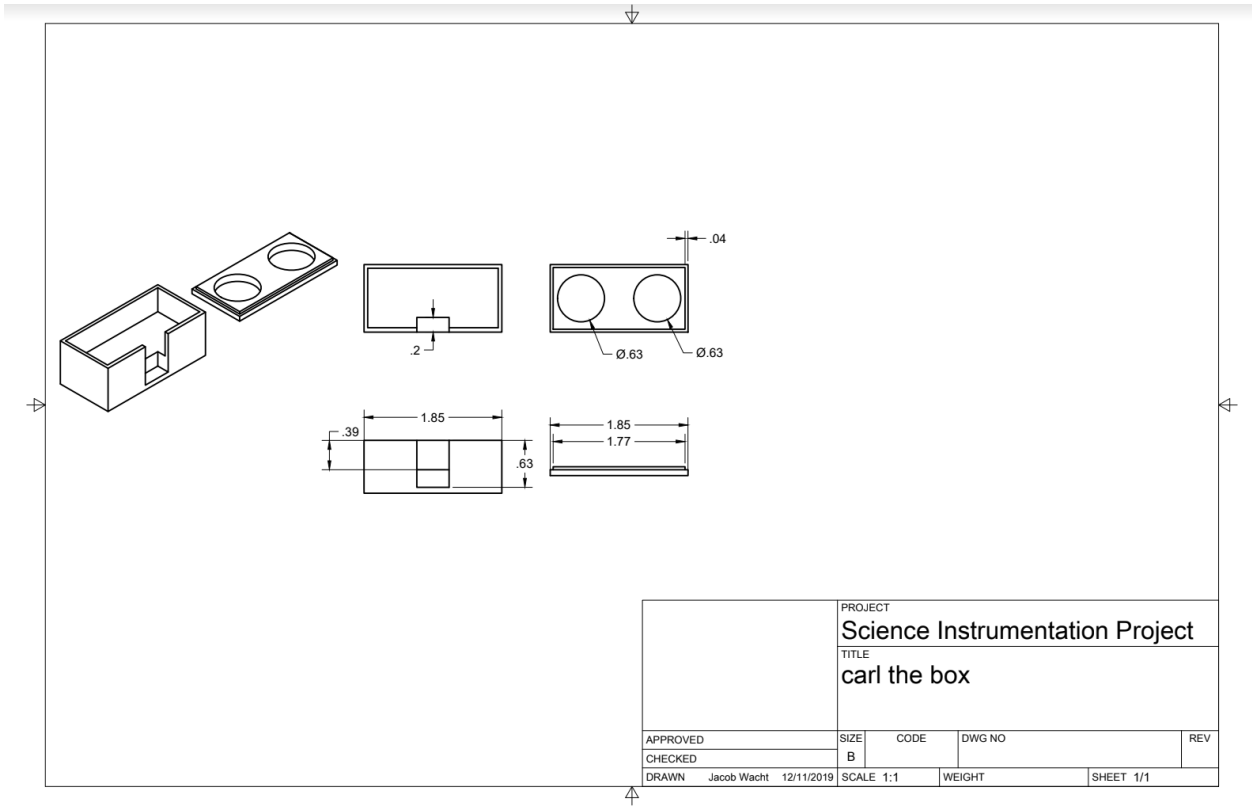
Pendulum Height changing Wheel:



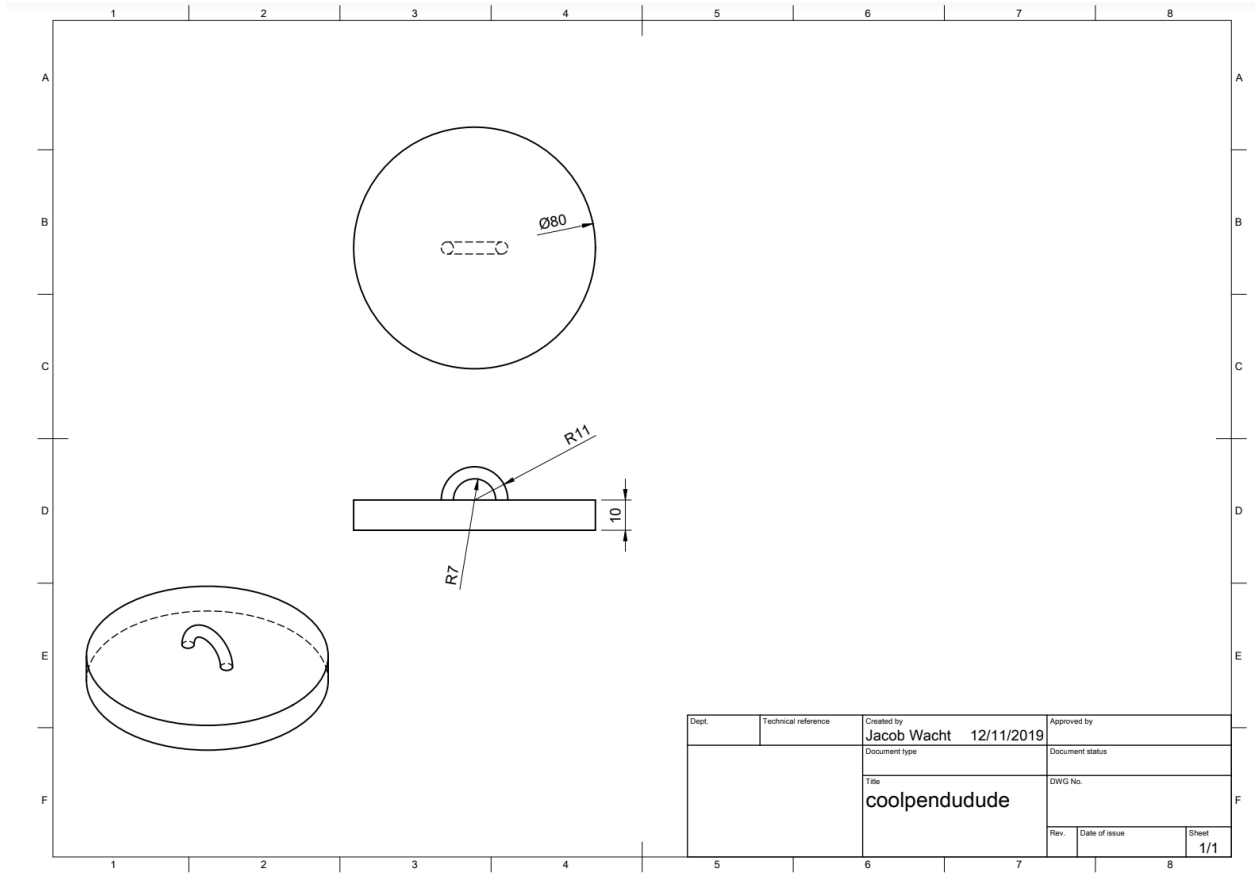
Rotating Wheel powered by DC Motor:

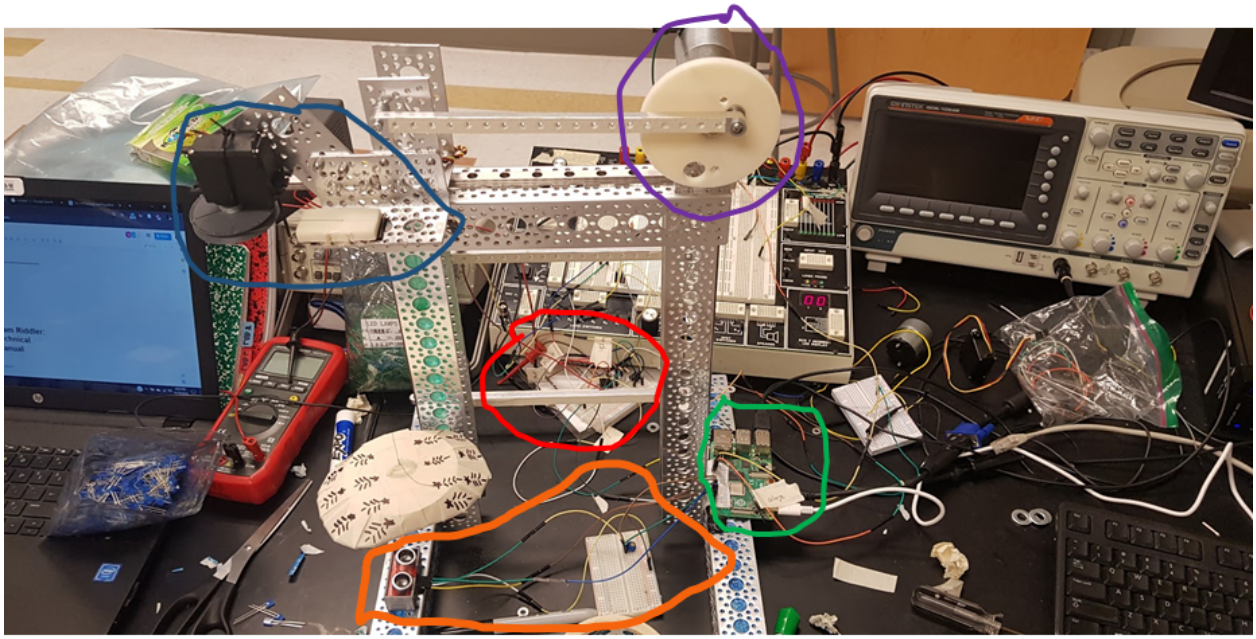


Acoustic sensor box:



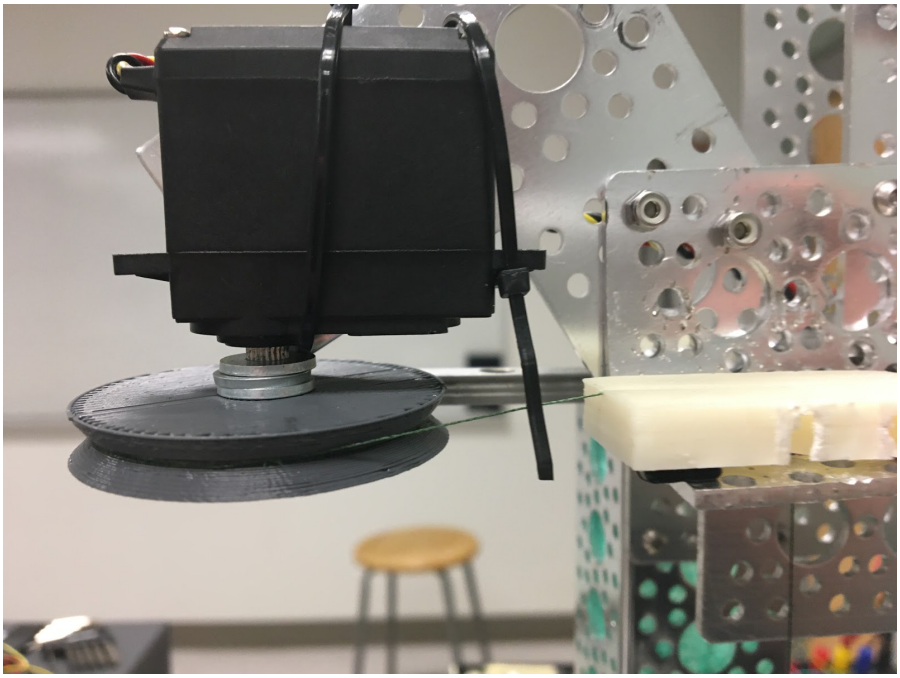
Pendulum:


Ø80
R11
10
R7
Dept.
Technical reference
Created by
Jacob Wacht 12/11/2019
Approved by
Document type
Document status
Title
coolpendudude
DWG No.
Rev.
Date of issue
Sheet
1/1

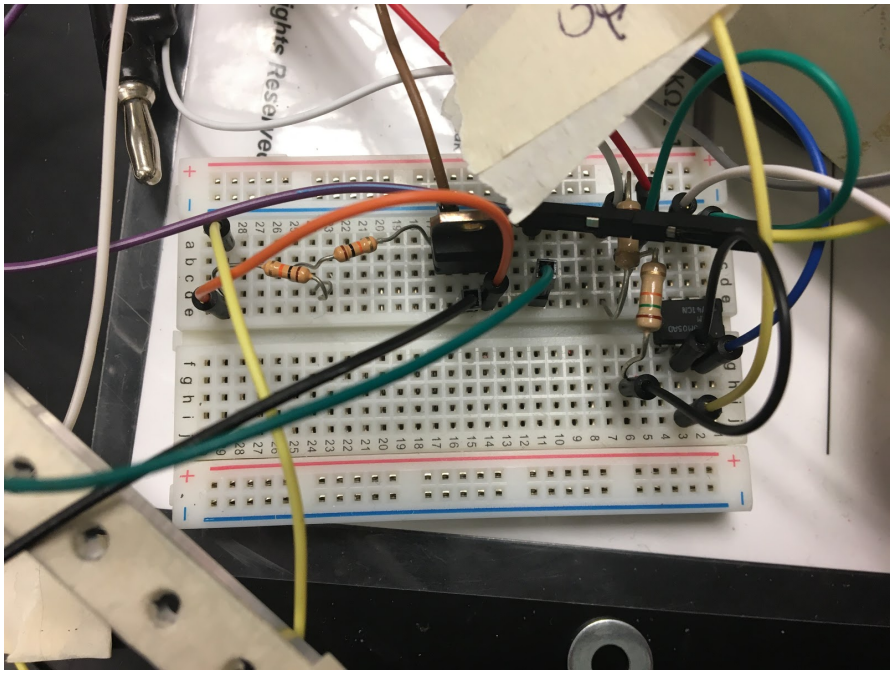


Blue- Servo and Height unit on reduced friction rail
Red- motor control unit
Green- Pi
Orange- Frequency detection unit
Purple- DC motor assembly with solid body rotation oscillator

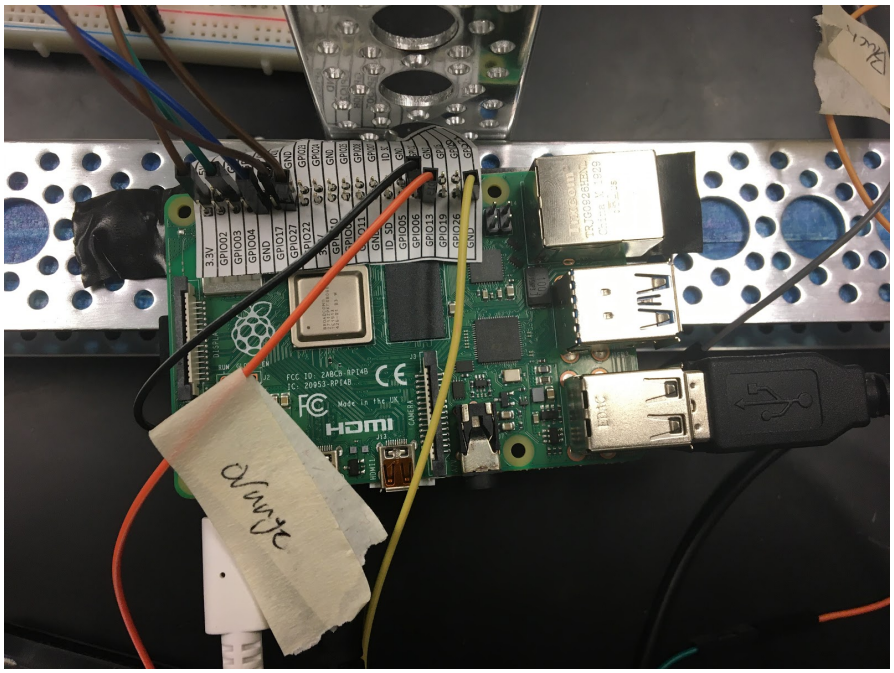
Servo Motor with String and Spool



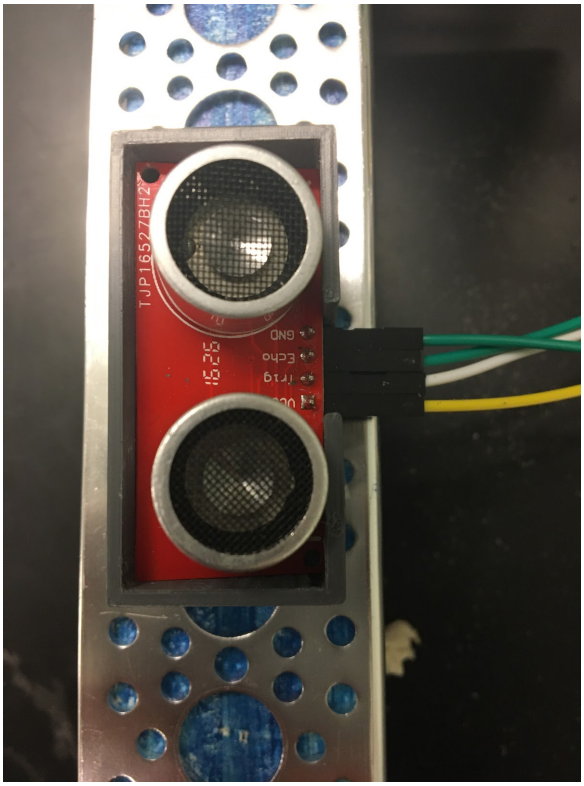
DC and Servo Motor Control Circuit



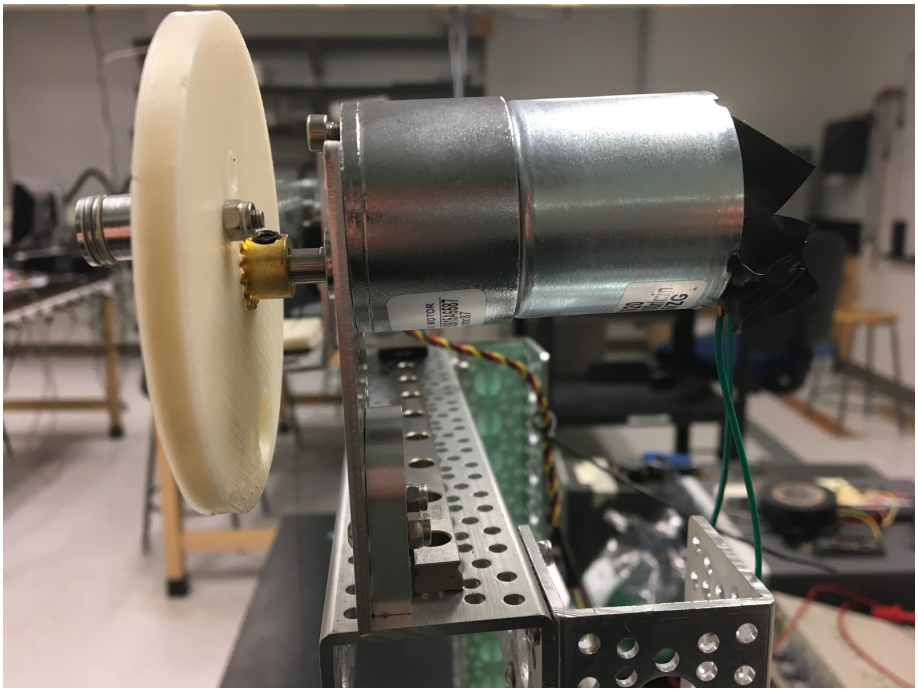
Raspberry Pi 4



Frequency Detection Acoustic Sensor

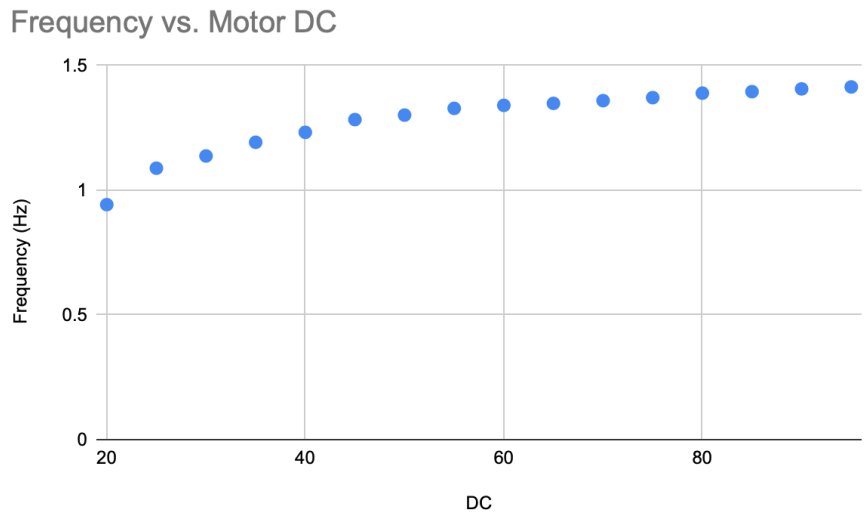


DC Motor with rotating Wheel and Arm



Finding Duty Cycle to Frequency Conversion

Motors usually operate under duty cycle conditions to change the rotation rate. However, each motor is teched differently, so one must undergo experiment to find the specific conversion rotation rate. This task can be done multiple ways using many sensor techniques, but for this pendulum, the team utilized the photodiode. The photodiode increases its voltage output in the presence of more photons, as it converts it to electrical energy. A solid wheel with a small hole can be attached to the given motor, and a small circuit setup of an order of the voltage source, photodiode, resistor and ground connection may be used to find the rotation frequency with the help of an oscilloscope. One places this circuit on one side of the wheel with an oscilloscope probing the resistor. One must make sure that the hole passes the photodiode once per cycle. The other side of the motor's wheel has a strong light source, which will be used to ping the photodiode. After activating a low DC to the motor, one measures the frequency of pings shown in the oscilloscope. Repeat the process for multiple DC signals to find a curve of given duty cycles and rotation rates, which results in a curve shown below. The curve below is for the team's specific motor.



This curve is then interpolated within the code to determine the necessary PWM signal to drive the pendulum at the chosen frequency.

Software Programs

GPIO_PWM_servo_dc_finalproject_v4_FINALCODE.py

```
import time
import RPi.GPIO as GPIO
import numpy as np
import math as m
import matplotlib.pyplot as plt
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
startL=.23 #meters
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
#interpolating the points to be chosen
DCmade=np.zeros(16)
for i in range(0,16):
    DCmade[i]=20+5*i


Freq=[.942 ,1.088, 1.137, 1.192, 1.232, 1.283, 1.301, 1.328, 1.34, 1.348,
1.359, 1.371, 1.389,
      1.395, 1.406, 1.414]
points=np.linspace(20,95,300)

Finterp=np.interp(points,DCmade,Freq)
mindc=np.zeros(300)
```

```
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
#servo setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.OUT)
PWMControlservo = GPIO.PWM(12,50)
PWMControlservo.start(0)
dc=4 #tuned for the motor 7.5= 90 degree rotation
rotate=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
#change in length
#on a circle theta*r=arc length=dl
#15 degres per integer 1=15,2=30.....
#delta theta= 15*pi/180 = angle in degrees
#dl= delta theta* number along servo
r=0.025 #radius of wheel in meters
theta=15*m.pi/180 #angle of rotation
#create array of actual lengths
Length=np.zeros(12)
for i in range(0,len(Length)):
    Length[i]=.23+theta*r*i
print('length',Length)



print(rotate)
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
#dc driver setup
#warnings.filterwarnings("ignore")
GPIO.setmode(GPIO.BCM)
GPIO.setup(13, GPIO.OUT)
PWMControl = GPIO.PWM(13,60)
PWMControl.start(0)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(27, GPIO.IN)
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#best fit equations
def distance(tbegin):
    GPIO.output(18,1)
    StartTime=time.time()
    StopTime=time.time()
    while GPIO.input(27)==0:
        StartTime=time.time()
    while GPIO.input(27)==1:
        StopTime=time.time()
    dt=(StopTime*10**(6))-(StartTime*10**(6))
    while dt<70:
        while GPIO.input(27)==0:
            StartTime=time.time()
        while GPIO.input(27)==1:
```

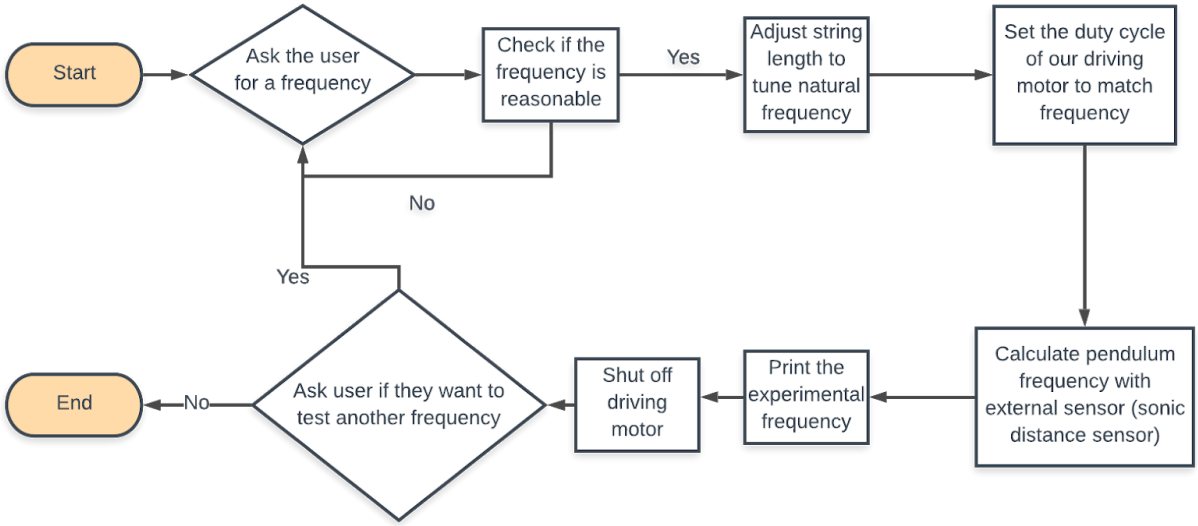
```
            StopTime=time.time()
        dt=(StopTime*10**(6))-(StartTime*10**(6))
    GPIO.output(18,0)
    print(dt)
    timestamp=StopTime-tbegin
    distance=dt/58
    return distance, timestamp
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#input your desired frequency from blank to blank
while(1):
    tbegin=time.time()
    dis=[]
    times=[]
    DESFREQ= input('What Freqency do you want betweeen .91 and 1.06 (Hz)')
    DESFREQ= float(DESFREQ)
    DESFREQrad= (2*m.pi*DESFREQ)**2
    # choosing the proper length first
    #freq=sqrt(g/l)
    #G/FREQ^2=L
    newL= 9.81/DESFREQrad
    print('newL',newL)
    while (DESFREQ>1.06 or DESFREQ<.91):
        DESFREQ= input('TOO LONG BRO. What Freqency do you want betweeen
.91 and 1.06 (Hz)')
        DESFREQ= float(DESFREQ)
        # choosing the proper length first
        #freq=sqrt(g/l)
        #G/FREQ^2=L
        newL= 9.81/DESFREQ;
        print("newL: ",newL)

    deltaL=startL-newL
    #solve for rotate
    #dl=r*theta*rotate[i]
    #Rotateangle= deltaL/(r*theta);
    #starting from the max length at postion 1:
    #rotate=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
    #15 degree increments moving to shorten the length
    #1=longest
    #transfer rotate into degreees
    #rotdegree=[0,15,30,45,60,75,90,105,120,135,150,165,180]
    #rotdegree=15*rotate



    value=newL
    #find the nearest value
    findnearest=np.zeros(12)
    for i in range(0,len(rotate)):
```

```
        findnearest[i]= abs(Length[i]-value) #abs since getting smalling
    print('findnearest',findnearest)

    minarray=np.amin(findnearest)

    minarray=np.where(findnearest == minarray)


    minarray=minarray[0]
    minarray=minarray[0]
    print('min array',minarray)
    actualrotate=rotate[minarray]
    print(actualrotate)
    # we have set the new length for the desired natural frequency
    #keep track of the nrew length to make it the start length
    if deltaL > 0 :
        startL = newL
    #set the driving frequency
    #find the closest to our interpolation

    for i in range(0,300):
        mindc[i]=abs(Finterp[i]-DESFREQ)
    minDCval=np.amin(mindc)
    minDCvalINDEX=np.where(mindc == minDCval)
    dc=points[minDCvalINDEX]
    print('DCin', dc)


    #selecting the drivng frequency
    #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    #run the loop
    #while(DESFREQ!=Freq):
    #Will need to create a loop comparing our measured frequency to
desiredfrequency
    #PWMControl.start(0)
    try:
        stop="n"
        while(stop!="y"):
            PWMControlservo.ChangeDutyCycle(actualrotate)
            time.sleep(2)
            PWMControl.ChangeDutyCycle(dc)
            print("assessing motion")
            time.sleep(5)
            n=0
            while n<100:
                n+=1
                dist,tmoney=distance(tbegin)
                dis.append(dist)
                times.append(tmoney)
```

```
            print("measured distance= ", dist)
        print("Measurement Stopped. Exit graph to continue.")
        #plt.scatter(times,dis)
        #plt.show()
        #finding the min of times to cut values
        minvalsensor=np.amin(dis)
        #determine the cut
        cut=minvalsensor+10
        #cut the values
        bf=[]
        bft=[]
        for i in range(0,len(times)):
            if dis[i] <= cut:
                bf.append(dis[i])
                bft.append(times[i])
        #print(bf)
        #print('uncut',bft)
        bftc=[]
        for i in range(0,len(bft)-1):
            if abs(bft[i]-bft[i+1]) >.4:
                bftc.append(bft[i])
        #calc the frequency
        period=[]
        for i in range(0,len(bftc)-1):
            period.append(abs(bftc[i]-bftc[i+1]))
        averageperiod=2*np.mean(period)
        #mult by 2 since measure half oscilation
        print('frequency Calculated', 1/(averageperiod))


        PWMControl.stop()
        stop=input("Do you want to do a new frequency? (y/n)")

        #run dc motor




    except KeyboardInterrupt:
        break
PWMControl.stop()
GPIO.cleanup()

#
#Need to:
#- Write code in terms soley of newL. This will no allow for negative
values.
```

```
#- if change is 15 degrees, length change is
```

Code Flow Chart:



servofunc1.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 30 16:31:11 2019

@author: Bennett        value= 9.81/DESFREQ;

"""
import numpy as np

def servo_set(length, interpolatedata, points):
    """ Taking input frequency """
     DESFREQ= float(input('What Freqency do you want betweeen .91 and 1.06
(Hz)? I will get as close as possible.'))
    value=9.81/DESFREQ
    while(DESFREQ>1.06 or DESFREQ<.91):
          DESFREQ= float(input('NOT IN RANGE \n What Freqency do you want
betweeen .91 and 1.06 (Hz)? '))
        # choosing the proper length first
        #freq=sqrt(g/l)
        #G/FREQ^2=L
        value= 9.81/DESFREQ;
        print("newL: ",value)
    rotate=[i for i in range(1,len(length)+1)]
    rotdegree=[15*i for i in range(0,len(length)+1)]
    print(rotdegree)
    findnearest=np.zeros(12)
    for i in range(0,len(rotate)):
        findnearest[i]= length[i]-value
    #finding the absolute distance from 0 for the findnearest.
    #want the length closest to 0 meters
    for i in range(len(findnearest)):
        if findnearest[i]<0:
            findnearest[i]=-findnearest[i]
    print(findnearest)
    minarray=np.amin(findnearest)
    actualrotate=rotate[np.where(findnearest==minarray)[0][0]]
    print(actualrotate)

    """ Handling interpolation """
    mindc=np.zeros(300)
    for i in range(0,300):
        mindc[i]=abs(interpolatedata[i]-DESFREQ)
    minDCval=np.amin(mindc)
    minDCvalINDEX=np.where(mindc == minDCval)
    dc=points[minDCvalINDEX]

    """ Return calculated values """
```

```
    print('DCin', dc)
    print('rotate', actualrotate)
    return dc, actualrotate
```