SwingingFruit

CUSTOM PENDULUMS

Established 2019

Table of Contents

Introduction

Congratulations! You are now the proud owner of your very own automatic Smart Pendulum™! We poured our hearts into this product, and we hope that whether you got it as a smart addition to your sitting room furniture, or the latest addition to your spiritual healing assembly, that you will love it as much as we do. Since we started SwingFruit LLC., the Uber of pendulums, a month and a half ago as a group of undergrads looking for a good project grade, we have always had one thing in mind: delivering a product that can be seamlessly integrated into your life, and will bring you never ending joy. In pursuit of this goal we spent many weeks working with the most advanced technology an undergrad can get their hands on, and pushing it to its limit and beyond in pursuit of excellence. Our team of five highly dedicated yet comically underpaid students have worked to bring you a design with the best circuitry we could integrate, a strong and sturdily engineered frame, with intuitive programming so that all you have to say is what you want and it will take care of the rest.

The theory behind the mechanism of our pendulum is based on the idea of a perturbative expansion of the simple pendulum model. A brief description of the mechanics of our pendulum will be included here.

We can determine the behavior of a simple pendulum (i.e., how it moves) given the solutions to the equations of motion – available in any introductory physics textbook or the internet. However, what if we wanted to have a pendulum that would never stop moving, and at a specified frequency? A perpetual pendulum of sorts. We would then have to add a force to the pendulum, thus resulting in forced oscillations. Think about moving your hand back and forth with a string tied around a ball. If you move the string with some frequency, you can drive the pendulum to swing. If you do this correctly – as in with the proper timing of movements – you can drive the pendulum forever (well, until your falls off). We can determine the behavior, similarly to before, with a small angle approximation with the equations of a forced oscillator, given as follows:

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \omega_0^2 x = \omega_0^2 d_0 \cos\omega t$$

Where γ is the damping factor (or the force counteracting the motion resulting in the decay of the motion and the eventual stopping of motion), ω is the frequency (and ω_0 is the natural frequency, or the frequency the pendulum would swing at if no forces were acted upon it), and d_0 is the horizontal distance we move to drive the pendulum. We can therefore solve this equation to determine the position, acceleration, frequency, and distance with respect to time that we should expect from the pendulum. However, we should note that because we used the small angle approximation ($\sin\theta \approx \theta$), we expect some difference in what our approximated model gives and that which we observe. Herein lies the ingenuity of our device. We use the techniques developed in electronics class to construct a feedback-loop that will self-correct for the difference in the approximation and reality. Specifically, we can use an accelerometer on the pendulum that will read the values of acceleration of the pendulum mass at any given time and relate those values to the frequency of the pendulum. Comparing the two values, we can determine if the frequency is too low, too high, or just right. If the values are not equivalent we can then increase or decrease the driving mechanism of the pendulum to increase or decrease the frequency. We can repeat this process continually to keep the frequency equal to the input frequency – continually increasing or decreasing the driver as necessary. As a result, we get a pendulum that oscillates forever.

One of the key components of our setup is the bluetooth enabled pendulum, that remotely sends data to the controller to allow real time readout of the frequency, as well as feedback control of the pendulum. To obtain frequency measurements we decided to use an accelerometer (the Adafruit LIS3DH 3-Axis Accelerometer) placed on the end of our pendulum. The basic idea was to align one of the axes along the length of the string and then measure the frequency with which the greatest value the acceleration occurred along that axis. This should have been when the accelerometer axis was aligned vertically. Instead of running four connection wires down our pendulum which would invalidate the assumption that our string is essentially massless, we chose to send acceleration measurements to the Raspberry Pi 4 using Bluetooth. We purchased a bluetooth adapter (the Adafruit Bluefruit Low Energy UART Friend) which we connected to an Arduino UNO microcontroller along with our accelerometer. We powered the Arduino with a battery pack that was taped to the bottom of it. The Arduino read measurements from the accelerometer and then wrote them out through the Bluetooth adapter, both of which were also taped onto the Arduino. This helped make the Arduino block double as the pendulum's hefty mass. In addition, Arduino block was so effective that it was able to continue sending data to the Raspberry Pi even if it was in another room. On the Raspberry Pi we used the bluepy module contained in the BlueZ software package to connect the Raspberry Pi and obtain measurements through Bluetooth.

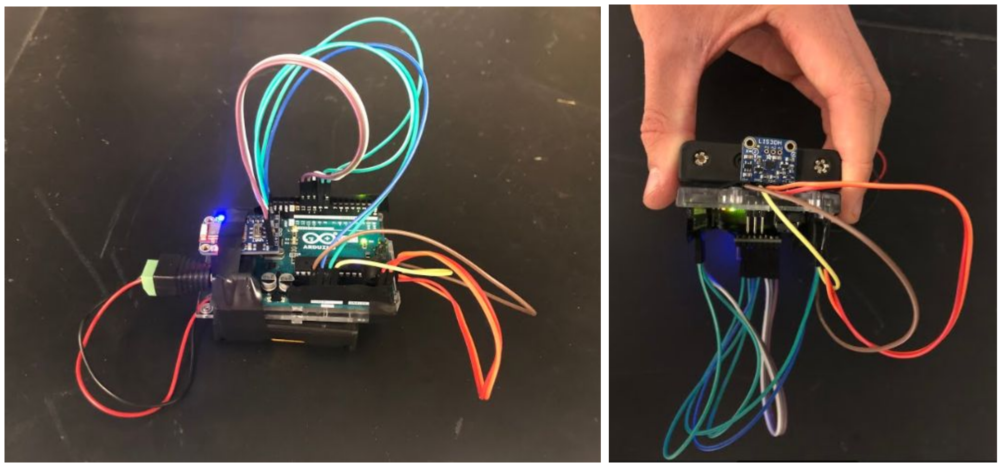

Fig. 1: The Bluetooth Accelerometer setup. The Arduino sits on top of the battery pack and the Bluefruit UART LE Friend is taped on top of the Arduino as shown in the left image. The blue light indicates that it is connected to the Raspberry Pi. The right image shows the LIS3DH which is hidden underneath the right side of the Arduino in the left image.

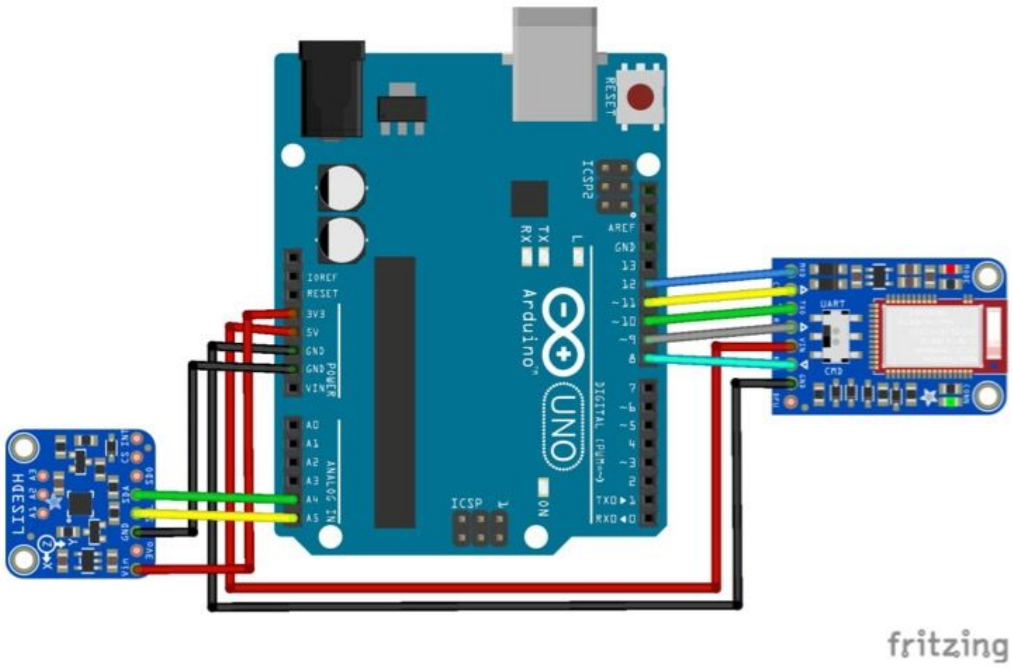

Fig. 2: Connection diagram for our Bluetooth Accelerometer. To the left of the Arduino is the LIS3DH and on the right is the Bluefruit UART LE Friend.

We are able to log in to our Raspberry Pi remotely and run our programs since it is connected wirelessly to the internet. We also added user input prompts on the command line so that we may specify a new length or frequency for the pendulum while it is running. For ease of use, we also connected Siri Voice Control to our Raspberry Pi so when our code is running we may ask Siri to write a note specifying the desired frequency or length of the pendulum and our pendulum will adjust accordingly. The note is sent to the Raspberry Pi through a designated email account.

Another important aspect of our device is the motor driving system. In order to actually raise and lower the pendulum, as well as modulate its amplitude to keep it swinging we needed to have a pair of motors that could be remotely controlled. Our initial idea was to use a mosfet, however, we dropped this idea because it would be difficult to integrate it with the stepper motors we wanted to use. Stepper motors require multiple input channels and a mosfet could only handle a single channel, and it also had limited current supply which could not support the requirements of the stepper motor. Our next idea was to try and use number of h-bridges to run the motors, however, this idea was dropped due to confusion over how they operated. For a long period of time we attempted to use AdaFruit motor shields with the Raspberry Pi, and although a degree of success was obtained, with a motor managing to rotate a small amount at one time, we inevitably dropped this idea due to multiple boards catching fire.

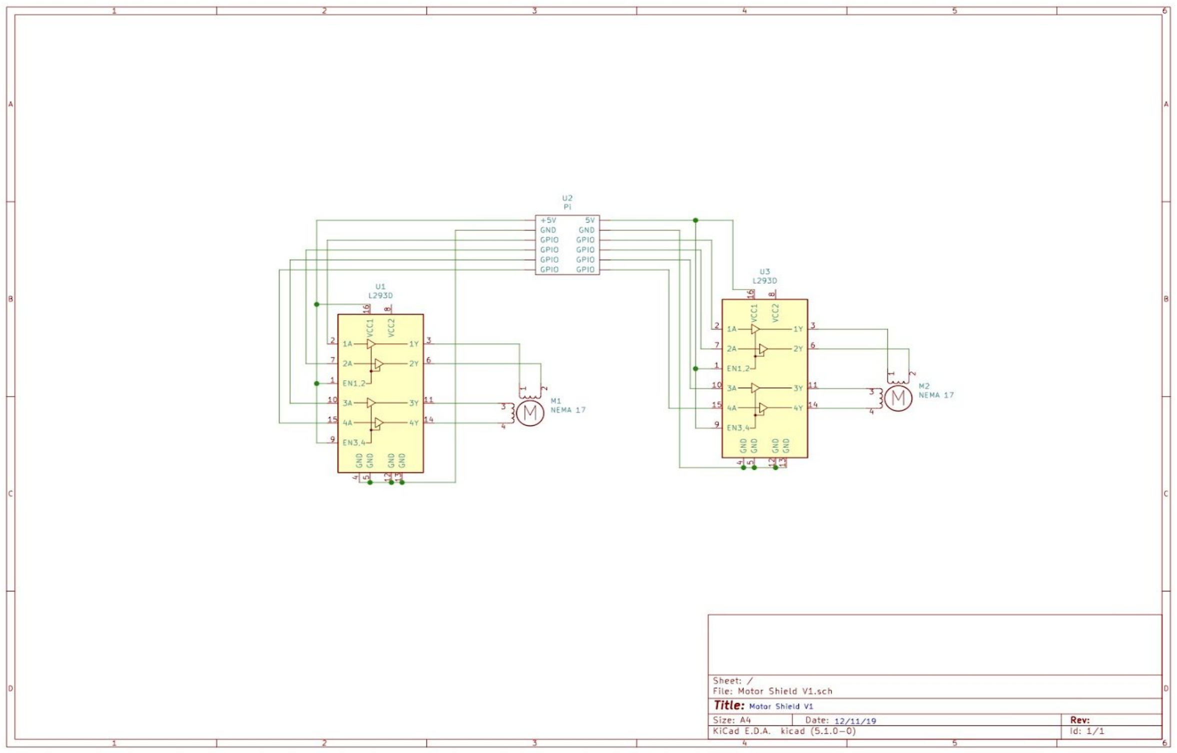

Fig 3: Circuit Diagram for the attempted L293D based motor driver. The motors in this setup were NEMA 17.

Our next attempt at motor driving was with a pair of L293D dual h-bridge motor drivers connected to GPIO outputs on the pi. The benefit of the L293D driver was that it could easily translate sequences sent from the Pi into movements, and it had a high enough rated current supply to run the motors happily. Although this solution worked for a while, and was able to successfully control each individual motor, when we attempted to introduce higher supply voltage to achieve the necessary torque, current backflow became a major issue, at times causing a total crash of the Pi. Attempts to rectify this issue with current protection allowed the system to operate, but fine motor control was lost. This led to a last minute scramble where we decided upon using an Arduino UNO with an AdaFruit MotorShield V1 slaved to the GPIO output from the Raspberry Pi as our motor driver. This motor shield is based upon the same basic design as the motor shield we attempted to create, with better protection and control. After some troubleshooting, we were able to achieve flame free control of both both motors with the necessary torque to operate the pendulum.

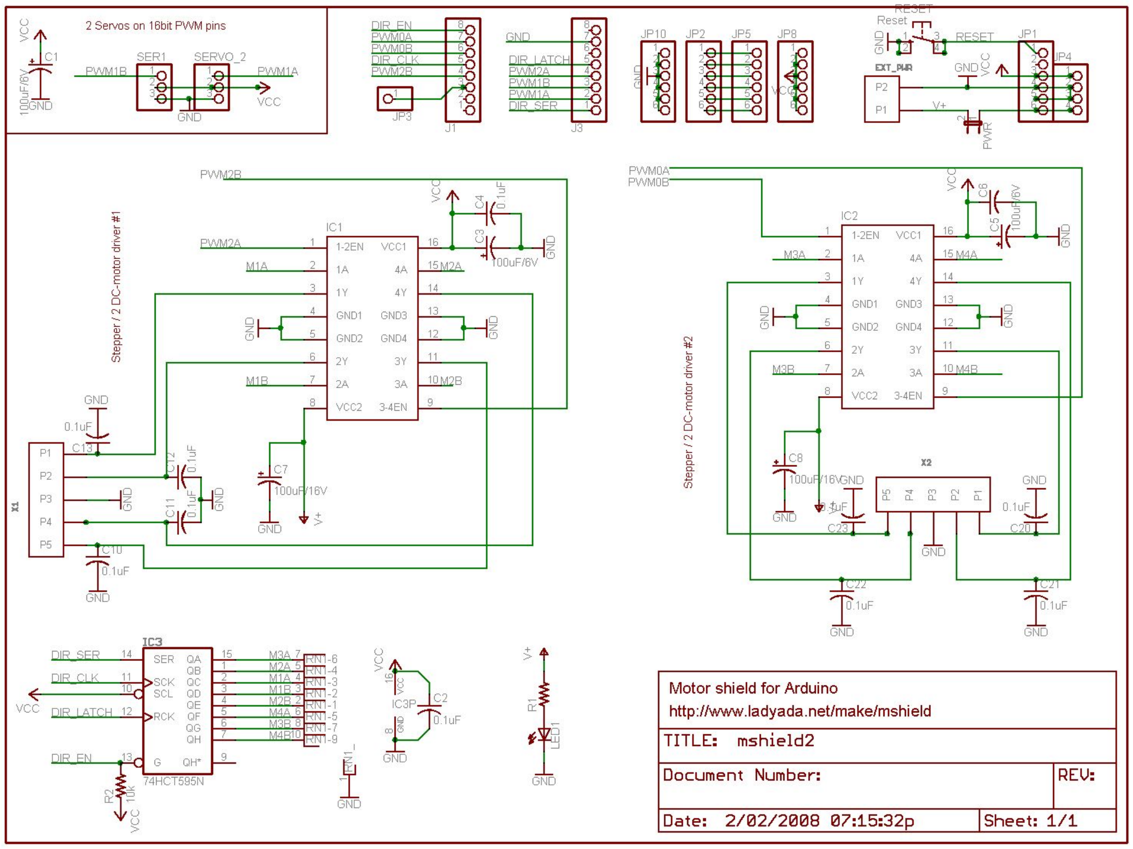
2 Servos on 16bit PWM pins
Stepper / 2 DC-motor driver #1
Stepper / 2 DC-motor driver #2
IC1
IC2
IC3
74HCT595N
Motor shield for Arduino
http://www.ladyada.net/make/mshield
TITLE: mshield2
Document Number:
REV:
Date: 2/02/2008 07:15:32p
Sheet: 1/1

Fig 4: Circuit diagram for the AdaFruit Motorshield V1 that we used in the final version of the design.

Our design is based off of a shaker table used to drive the pendulum and a rugged support structure to dampen noise. We designed a moving stage/sled with a mount for the stepper motor that controls the length of the pendulum and a gear rack. We then mounted this stage on two cylindrical lengths of aluminum with two stops on the ends. These stops were attached to the top of our support structure. The stage was moved by a gear mounted on a stepper and whose mount was fixed on the support structure as well. The gear-stepper was held in place with a simple clamp that used the same holes as the base to reduce the number of fasteners. Both the spool and the gear were mechanically fixed to the shafts of the steppers using collars and flat-bottomed screws. Due to the high torque of the system Lithium Grease was needed on the aluminum lengths so that we could use lower power stepper motors. Our support structure was built of plywood and aluminum 80-20 extrusion, however in principle these could be changed for cheaper/nicer materials depending on how much money people give us. 80-20 is rugged and sturdy, although a bit pricey, and was used for the vertical supports and the crossbeams. Plywood was used for the base and the top as it is easy to mount onto and it has a high degree of durability. It also allows for us to directly mount the Raspberry Pi and other electronics directly to it as it is non-conductive. The cross beams were added to reduce noise in the system and for the protection of those around the pendulum as it has the tendency to fly into pure chaos when in use.

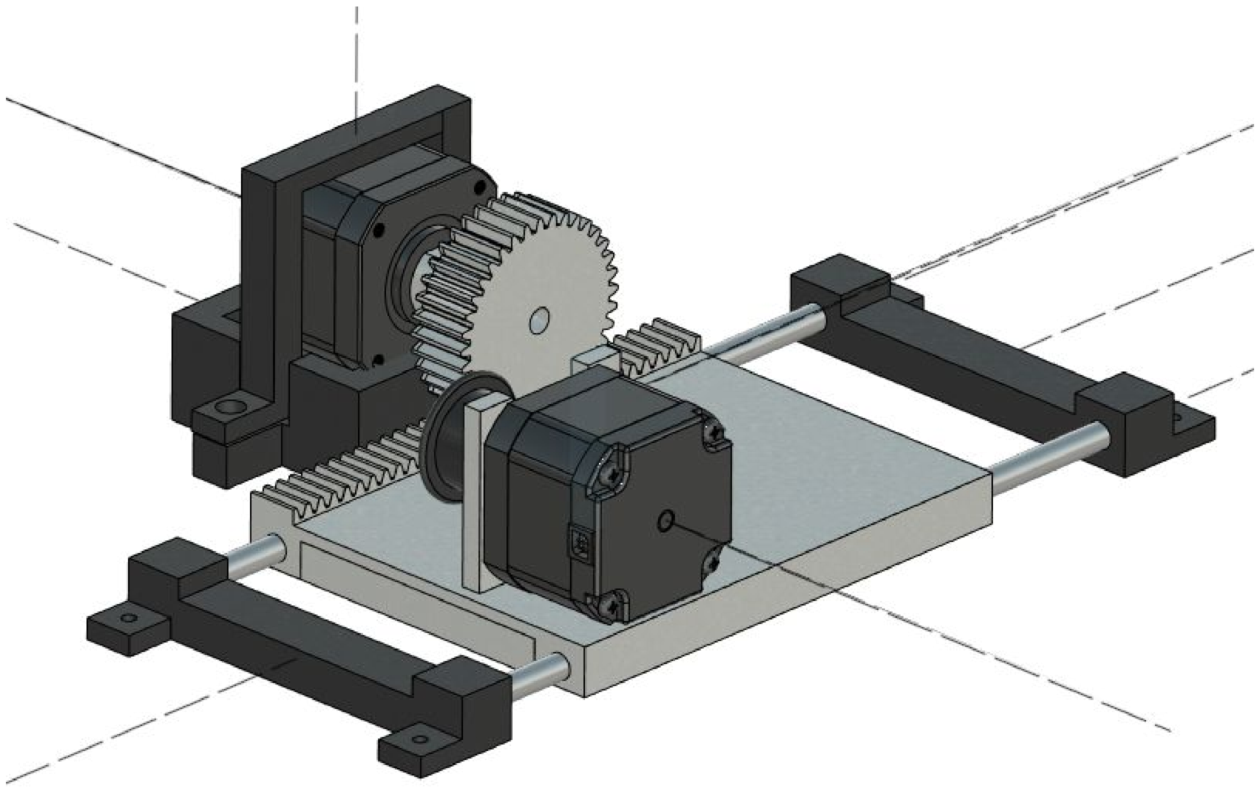

Fig. 5: CAD model of our shaker table used to check compatibility of 3D printed parts with motors and other hardware.

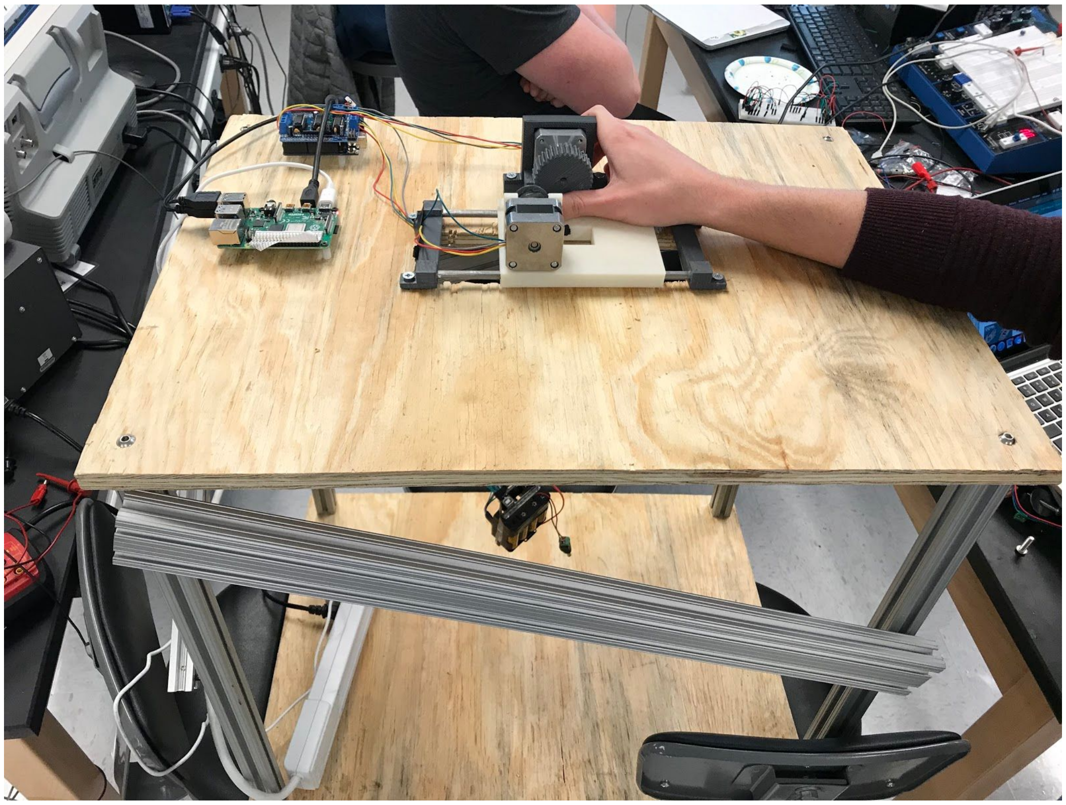

Fig. 6: Top down view of our Version 4 prototype including mounted electronics and hanging pendulum.

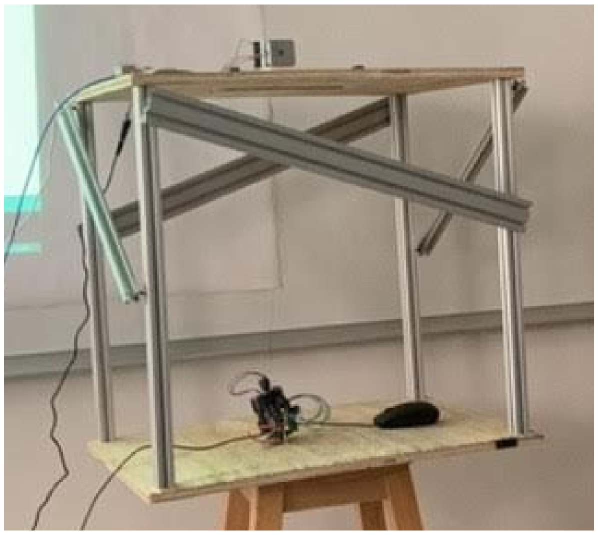

Fig. 7: Side view of the pendulum highlighting support structures.*Mouse not included

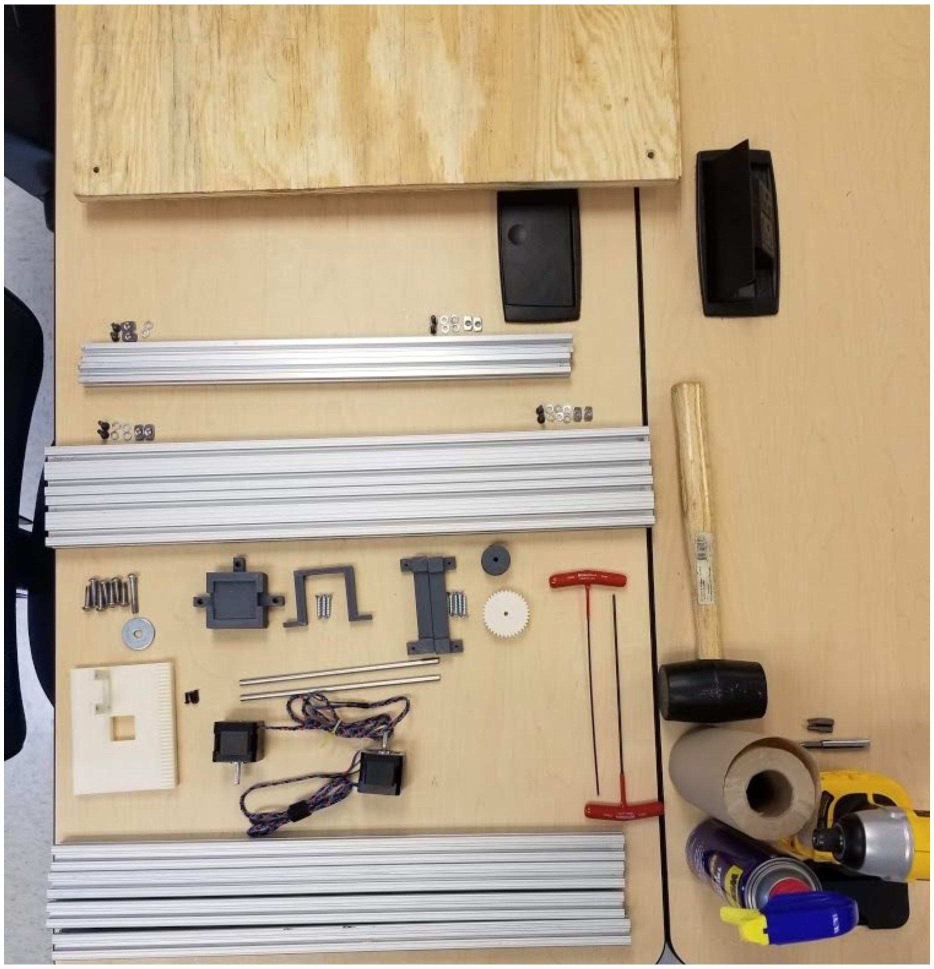

Fig. 8: View of our hardware before the assembly of the prototype.

<u>Specifications</u>

Here are some highlights of our device:

- It can obtain accurate bluetooth measurements from an accelerometer with the controller as far away as 30ft. Long distance remote measurement!!!

- The controller can take voice control commands when its setup on your home WiFi. Don't feel like getting up to get the controller, you don't need to, just tell it what to do! You don't even have to be in the same room!
- Easy to set up, just plug the complete device into a wall outlet, and you are ready to go!
- Looking for variety? Our device can handle a wide range of low frequencies handling around .75 - 1.5 Hz
- Our amplitude modulation technology keeps the pendulum swinging as long as you want. You could forget about it for a whole year and it would still swing! Endless Fun!
- You can sit on it! You can kick it! You can even hit it with a mallet! It would survive a nuclear holocaust!*

*We assume it will but have not actually tested this due to lack of access to nuclear warheads.

Tutorial

To get you started with using our device, we have included a step by step tutorial on how to get the device set up and running as you want it!

1. Begin by plugging the device in, and once plugged in give the device a few moments to startup.
2. Once the device has started up, make sure your mobile device has the app Shelly installed, which allows you to interface with the device by mobile phone. Plug a mouse and a monitor into the control computer on the top of the device.
3. Once on the desktop, navigate to the toolbar and find the network icon. Scroll over it to identify the devices IP address. Record this IP address and then input it on Shelly.

4. You can now disconnect everything from the control computer, and return to Shelly on your phone. You are now on the command line of the controller. Use the following commands to navigate yourself to the Pendulum Control program:

```
cd Documents
cd Python / Projects\
python3 Threaded_MotorRunner_BLE.py
```

   This will run the Pendulum Control program, and prompt you to input the desired frequency. Once you have input your desired frequency, the program will take care of the rest, changing the length of the pendulum such that it will swing at the right frequency, and the motor will modulate the amplitude to keep it swinging. When you are done just type "Done" on the line. If you want to do a different frequency just rerun the program and input the new desired frequency.

5. To connect the Raspberry Pi to Siri on your phone first create an email account for your Raspberry Pi and type the username and password into the siri_control.py file in the ~/Documents/Python\ Projects/ folder. Then, in settings on the Raspberry Pi's email account make sure that IMAP access is enabled. On your iPhone add the Raspberry Pi's email account and make it the default account. Also, allow it to access notes on your iPhone. When you want to send a command to the Raspberry Pi through Siri, simply ask Siri to write a note and then have her write your commands in that note. You can specify either frequency or length of the pendulum with Siri Voice Control. For further more detailed instructions, see Sanjeet Chatterjee's online example at https://www.deviceplus.com/how-tos/add-siri-control-raspberry-pi-project/.

Troubleshooting

1. Is it on fire?
    a. Yikes, but not an uncommon problem. The easiest way to fix this is to run and hide and find a real adult. Don't forget to put it out before running away!
2. Is it not working for some non-fire related reason
    a. Hmm, we haven't experienced problems with this before so this is likely your fault. I'm not IT but maybe turn it off and back on again. Didn't work? Hit it with a mallet until it does.

Appendix

Team Members and Roles:
Doyle Weishar: Pendulum and motor supports design and construction
Alex Fay: PLL Coding and motors
Gaëtan Poirier: PLL circuitry and motors
Rodrigo Arias: Bluetooth accelerometer
Matthew McEneaney: Bluetooth accelerometer and Siri Voice Control

List of components:
- Adafruit LIS3DH Triple Axis Accelerometer
- Adafruit Bluefruit LE UART Friend
- Adafruit V2 Motor Shield
- Arduino Uno
- Arduino Trinket
- Raspberry Pi Zero W V1.1

- Raspberry Pi 4

Sources of components:
W&M MAKERSPACE

List of vendors, prices, availability:
McMasterCarr.com
OpenBuildStores

- $17 Bluefruit UART Friend.
- $0.05/hr * 40 hours = $10.
- (Engineering R&D: $12/hr * 40 hr = $480 -- not included in pricing)
- $10 Plywood.
- $11.62 * 2 Nema 17 Stepper Motor cost.
- $19.50 * 2 Raspberry Pi Motor-shields (R&D cost).
- $7.00 * 8 Structural 80-20 Extrusions
- $25 ABS Filament.
- $15 Fasteners
- $2.77 Wiring
- $22.00 * 2 Arduino Uno
- $20.00 Arduino Motor Shield V1
- $35 Raspberry Pi 4

Sum of hours to complete the project (represented as cost of pendulum)
We estimate the cost of our pendulum at about $150. Each team member spent at least 40 hours on this project resulting in 200 total man hours.

List of software libraries:
Software:
Bluez 5.37 (includes bluepy)

Libraries/Modules:
Adafruit_BluefruitLE_nRF51 (C)
Adafruit_LIS3DH (C)
Bluepy (Python)
AFmotor.h (Adafruit motor shield library)

Code:
Available at https://github.com/mfmceneaney/PHYS351.

.