Breese Sherman, Vivian Lu, Quinn Campagna, and Cody Hammack

Final Report for Team Thanos
December 11, 2019

History

Our road to building a driven pendulum has been a long and troublesome one. During our first team meeting on Friday, November 8th, we tossed around some very creative ideas regarding how we would construct something like this. Among the designs that were discussed were: a marble rolling up and down on an endlessly twirling slope, a motor on one axis rotating another motor on a perpendicular axis, and a large array of photodiodes set up next to a bright bob to detect its exact position. All of us (five at the time) were excited to start working on these constructions the next time we came to the lab.

Quickly, however, we realized we were aiming too high relative to our skill levels. The marble idea didn't count as a viable pendulum, and the photodiode-array would necessitate an extreme amount of wiring and code. Soon, the only surviving idea was the double-axis pendulum, and even that plan fell through when Colm's 3D-printed designs didn't pan out. At this point, all we had was Colm's recommendation to use stepper motors. Then he disappeared, never to be seen again.

The three following weeks consisted of the following cycle, repeated over and over again: the remaining four of us trying to rig up the simplest possible pendulum, failing, and trying something even simpler. Along the way, we had to sacrifice many planned features of our pendulum (including a variable length and variable driving speed) in order to create something that would actually work. We weren't able to control the motor's speed in real-time, not that it really mattered, because the motor seemed to move at a random speed irrespective of what we actually put into the code.

Eventually, we settled on something about as simple as one could get: a bob attached to a piece of fishing line strung up to the original stepper motor we had started out with. Our finalized pendulum works like so: An accelerometer on the bottom of the bob informs a Raspberry Pi the acceleration it feels on the z-axis. If that acceleration is below a certain threshold, drive the pendulum with the motor. If it's above the threshold, let it swing freely. This creates a never-ending cycle of driving the pendulum until it reaches a certain amplitude, letting its swing dwindle for a bit, and then moving it once more. This mechanism prevents the pendulum from starting to swing in a circle, which was a behavior we noticed when we made the motor drive the pendulum constantly.

It's not a perfect machine, but at least it swings back and forth as it's supposed to. It should run until the fishing line wears through from the constant swinging, or when the Pi/Arduino burn out — whichever happens first.

How it's built

Structure Building: We made an 'L' shape metal frame as the stand of the pendulum. We mounted a wooden board on top of the stand to hold the motor, the Arduino and the Raspberry Pi. A support weight was placed on the bottom part of the 'L' shape to ensure the stability of the stand. We fastened the motor to the wooden top with a 3D printed mount (Diagram 1). The motor was mounted such that its shaft extended out to hang the pendulum. A 3D printed mount (Diagram 2) was securely attached to the motor shaft so that a piece of fishing line attached to the shaft could be swung by the motor.

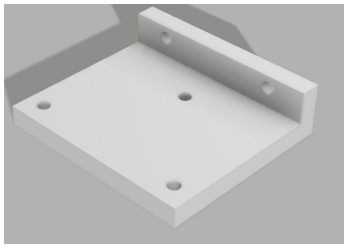


Diagram 1

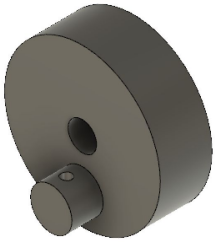


Diagram 2

A bob was 3D printed as a plastic cube (Diagram 3) with a bracket on its bottom to hold the accelerometer. The fishing line was attached to the handle of the bob. We then added small weights to the bob until it was perfectly balanced. We also attached an LED strip around the stand for aesthetic purposes.

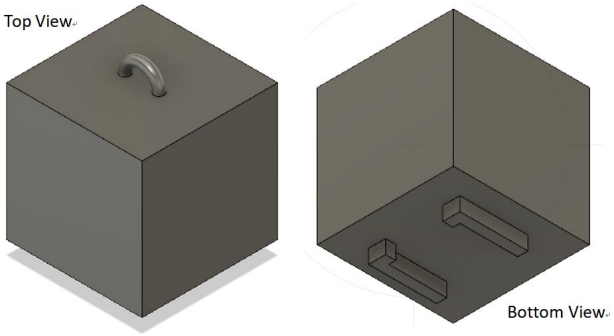
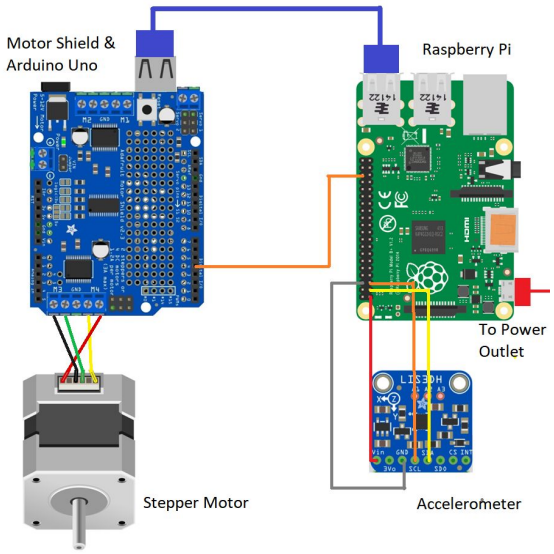
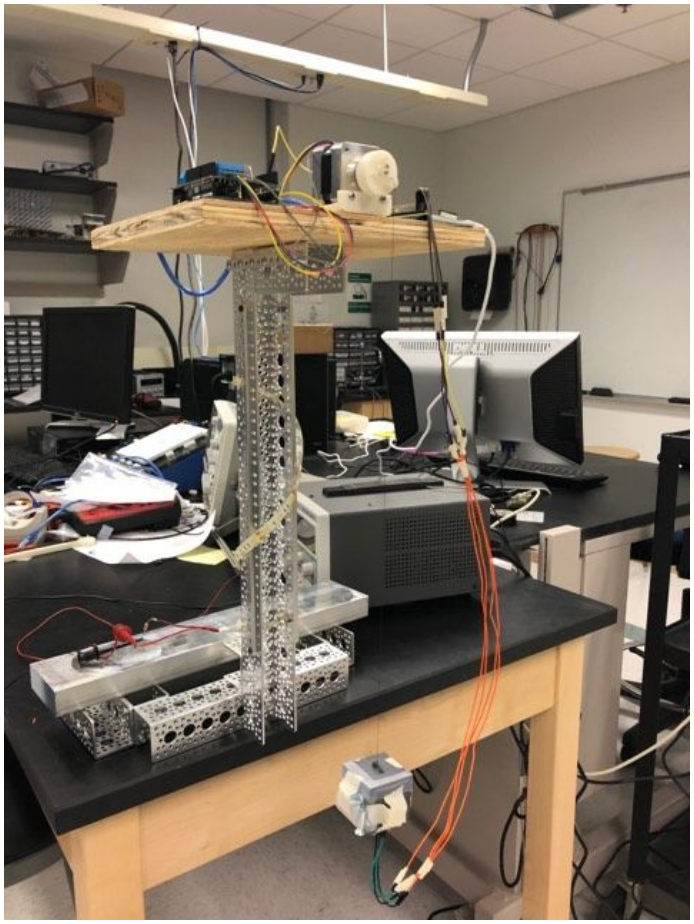


Diagram 3

Circuits Building: We decided to use a motor shield connected to an Arduino Uno to drive the stepper motor and to use a Raspberry Pi to load data from the accelerometer. The Pi was connected to a 110V power source. The Arduino was connected to the Pi through a USB cable for power supply. The entire circuits were shown below:



Picture of the Final Product:



How the code works

The code for this project is split between two programs of different languages: one Python file called `Thanos_frequency_detector.py`, and an Arduino file that directly controls the motor. The code to run the motor is already present and running on the Arduino, and does not need to be loaded or run by the user for the pendulum to work. Here's how both files work in tandem:

`Thanos_frequency_detector.py`, located in the `~/Documents/Python Projects/` folder on the Pi, begins reading the input from the accelerometer by importing `LIS3DHMOD.py` (a library for the accelerometer) and initializing the accelerometer object. On a quick loop, it tells us the z-axis acceleration the accelerometer feels, minus a certain fudge factor to account for gravity. Once per second, the program takes an average of the last 300 measurements made. If that average is below a certain threshold (in our case 0.22), it sends a signal to `DigitalIn` pin 4 on the Arduino (through the central orange wire in the circuit diagram above). Meanwhile, the Arduino code, aptly named `stepper.ino`, constantly checks to see if pin 4 is on or off. If it's on, the code rotates the stepper motor forward and back once by a certain number of steps. If it continues to be on after this rotation is done, it rotates the motor once more, so on and so forth.

In addition to averaging the z-axis acceleration over 300 measurements, the program also finds the maximum acceleration in those last 300 measurements, and notes the time it was taken. Then, every time the average is made, the program also takes the difference between the last two maxima, this being the approximate period of the pendulum.

How to operate the pendulum

For the pendulum to work, the Raspberry Pi must be connected to an external power source, such as a portable battery pack or a wall outlet. For the optional LEDs to turn on, they must also be connected to a 12V power source. Once that's done, all that needs to happen is that the principal Python file on the Pi be run. Given more time, we could have set this up using a physical button on the stand connected to the Pi, but we decided to focus more on getting the actual pendulum to work. Therefore, to run `Thanos_frequency_detector.py`, the user must SSH into the Pi and run it manually.

That's all there is to it! The motor will automatically start and stop based on the amplitude of the pendulum's swing. If the pendulum starts in a resting position, the motor will detect that and begin driving it. Everything is automatic, there is no need for any user input during its operation. Even if the pendulum is jolted such that it begins swinging in a circle, the motor will eventually drive it back to a linear path.

Appendix

Team members and roles

Breese Sherman — Coder/circuit designer/stepper motor wrangler
Vivian Lu — Builder/3D-printer wrangler
Quinn Campagna — Coder/circuit designer/accelerometer wrangler
Cody Hammack — Builder/LED-strip wrangler

Project components, their sources, and their costs

Item	Source	Cost
Metal frame	Makerspace	$10
Scrap wood	Makerspace	$5
Misc. screws	Makerspace	$2
Raspberry Pi	Electronics lab	$35
LIS3DH accelerometer chip	Electronics lab	$5
Arduino Uno	Makerspace	$16
Adafruit Motor Shield V2	Makerspace	$20
Nema-17 stepper motor	Makerspace	$13
Misc. wires	Electronics lab	$2
Fishing line	Makerspace	$3
3D-printed components	Makerspace	Could sell for $10 total

The total cost of all the materials above would be about $120. Given that there were four of us and we each put at least 10 hours into this project, this would come out to $3/hour — but I'd say our labor was worth closer to $15/hour, given how much we struggled with putting this all together.

Codes and libraries

The two files we wrote, `Thanos_frequency_detector.py` and `stepper.ino`, are located in a GitHub repository at https://github.com/brsherm/pendulum351. The former requires the LIS3DH library called `LIS3DHMOD.py`, the Raspberry Pi GPIO interface called `RPi.GPIO.py`, as well as some basic Python libraries such as `time`, `math`, and `numpy`.