The Boxer’s Pinata

By Victor H. Zendejas Lopez

Description of product:

This semester I enrolled in an online boxing course. Instead of using punching bags to practice technique and combinations, I used a flashcard hanging from the ceiling (see figure 1). Surprisingly, this flashcard makes for an effective target because it’s lightweight and agile. After a while however, I started getting a bit bored and I implemented some of what I learned in tME 102B to make punching this flashcard a bit more fun. I implemented a very simple system that moved the flash card up and down. This makes hitting the target a bit more difficult. The components that make up this system are an ESP32 mircorontrolloer, a dual motor driver, and a brushed DC motor. I fabricated a pulley that is attached to the DC motor via a hub out of the base of a large water bottle. As engineers, we work with constraints all the time, so I wanted to challenge myself and spend the least amount of money as possible to construct this system. The only supplies I purchased (besides the bottle of water) was the string that would be attached to the flash card and the ceiling mounted pulley.

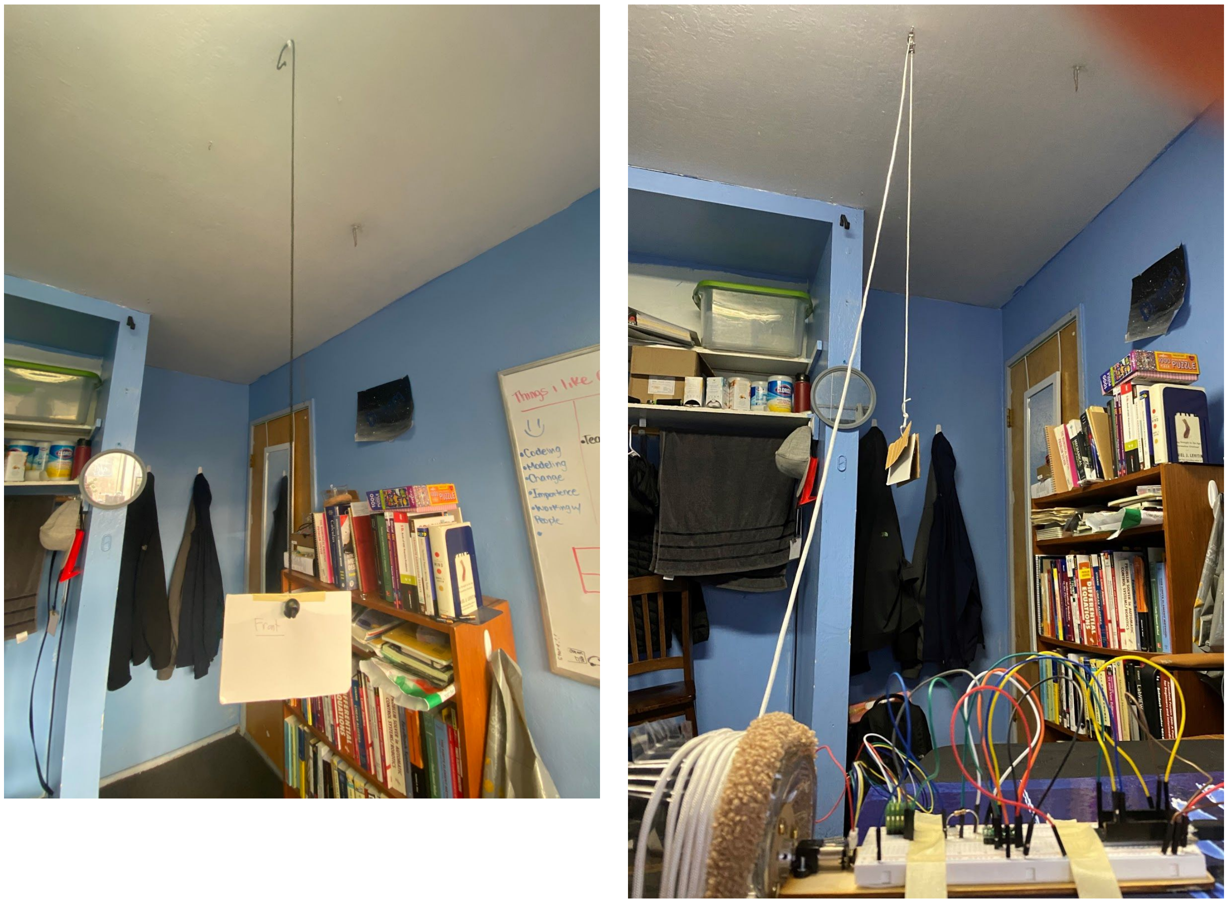


Figure 1. Left - before the automated system, this was the target used punching. Right- automated setup that moves the target up and down.

Electromechanical details:

The paper target is actuated up and down by the clockwise and counterclockwise rotation of the drive pulley attached to the DC motor. This setup uses no sensors for a feedback control strategy. Instead, a target speed is used to control the angular velocity of the pulley, and a delay is used to control the length of time the pulley will spin.

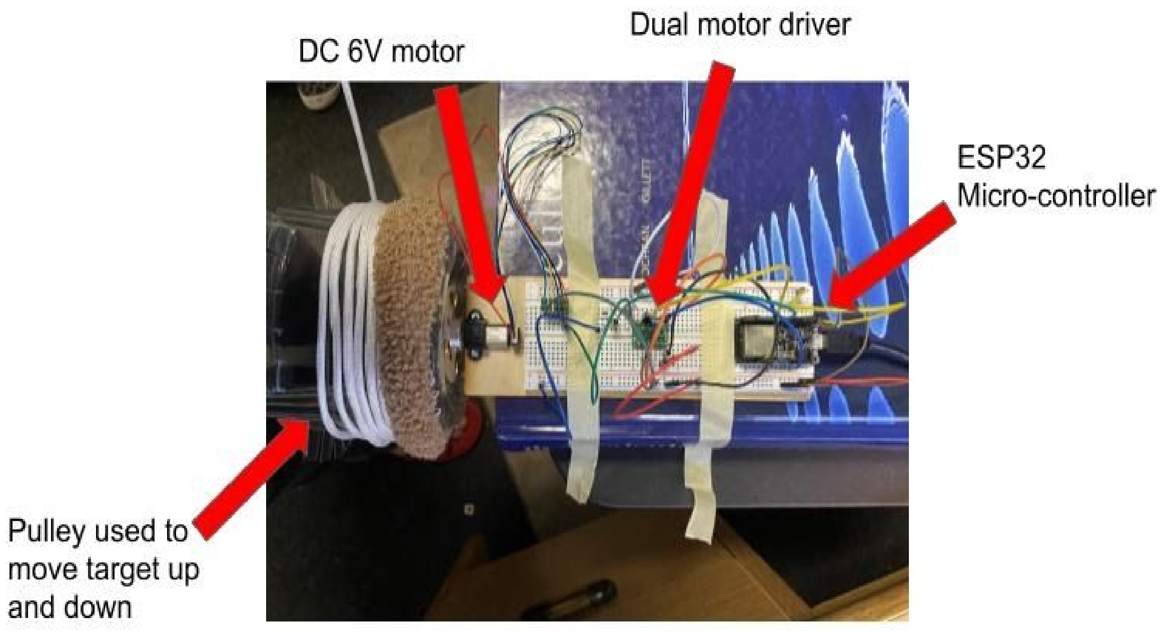


Figure 2. Micro-controller sends signals to dual motor driver to actuate DC motor

The custom part of this set up is the pulley that is attached to the hub that is actuated by the DC motor. The bottom of the water bottle was cut and a lip was formed from the side of the water bottle where the cut was made from. Holes were cut for the copper pins that would be used to attach the water bottle to the hub. The copper pins are ductile and were perfect for mimicking the shape of the inner lining of the water bottle. A hair scrunchy was used to keep the string aligned and on the pulley.

Because we are using a string to transfer motion and I'm assuming that under loading, it's not going to change in length, I can theoretically place the motor and pulley system anywhere in the room. The only torques acting on the pulley are the result of changing angular velocity and from the rope tension. None of these are functions of distance away from the where the ceiling pulley is located which lifts and lowers the target.

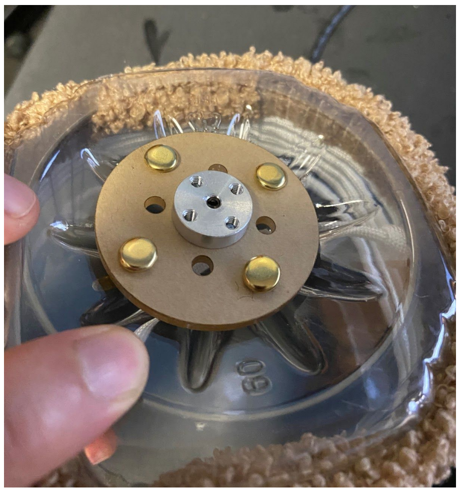
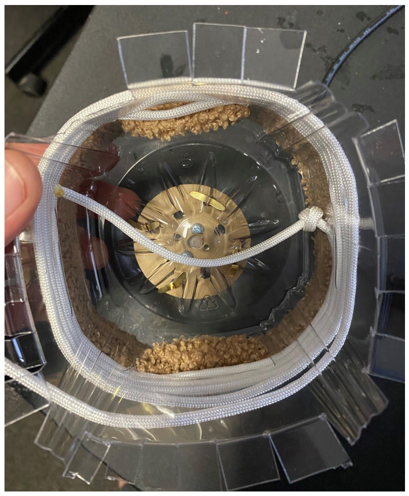



Figure 2. Plastic hub is attached to the aluminum hub by copper pins. The Aluminum hub is attached to the DC motor shaft

Circuit:

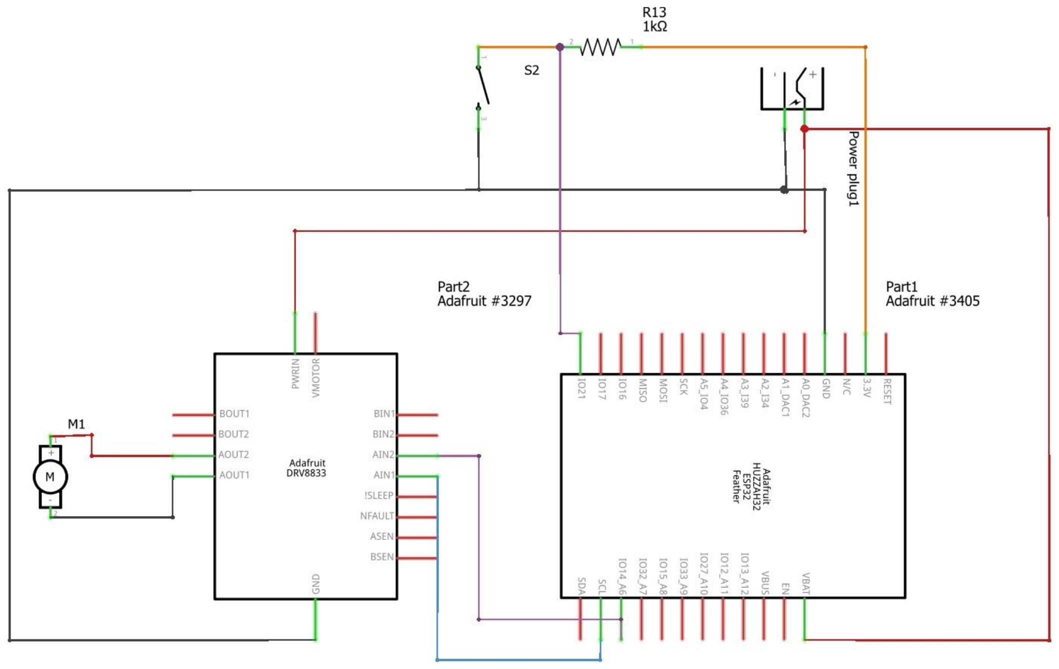


Figure 3. Circuit diagram of open-loop control strategy

There are three main elements that comprise the circuit that drives the pulley. The DC motor, the microcontroller and the motor driver.

1. The motor is the actuator of the system. It directly drives the pulley. I decided to use the DC brushed motor that was supplied in our lab kit. The motor is a gearmotor with a 75:1 gear ratio. It's maximum speed is 318 RPM which is more than fast enough for my purposes. The motor runs off the supplied 5 volts and is controlled by a motor driver.
2. The motor driver is composed of two H-bridge drivers which can drive two DC brush motors. A motor driver allows the flexibility of changing the motor polarity which changes the direction the motor spins. In my case, I wanted clockwise and counterclockwise rotation, so a motor driver was a good choice rather than using a single MOSFET. The motor driver is controlled by pulse-width modulation (PWM) from the microcontroller.
3. The microcontroller is a Adafruit HUZZAH32 – ESP32 Feather Board. The microcontroller actuates the motor driver and is responsible for controlling the motor effort. Since there is no feedback control strategy, the microcontroller relies on the user input (through Arduino IDE code) to control the motor effort through the motor driver.

The entire system runs on 5V. The DC motor will run on 3.3V which is the maximum voltage the microcontroller can output directly. However with this approach, the amount of power the DC motor can output is limited which will not meet the demands of the application. Also, driving the motor directly from the microcontroller can damage its internal circuitry because of high current demands under heavy loads. Peak current from my system occurs as soon as there is a change in direction and the motor needs to rapidly spin from a standstill. All electrical components used for this system were provided in the lab kit for this course.

The switch in this circuit diagram was not used in the final product. It was used for diagnostic purposes.

Finite state machine:

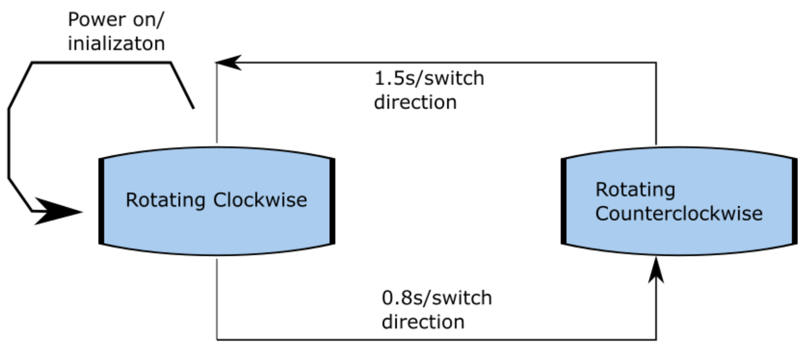
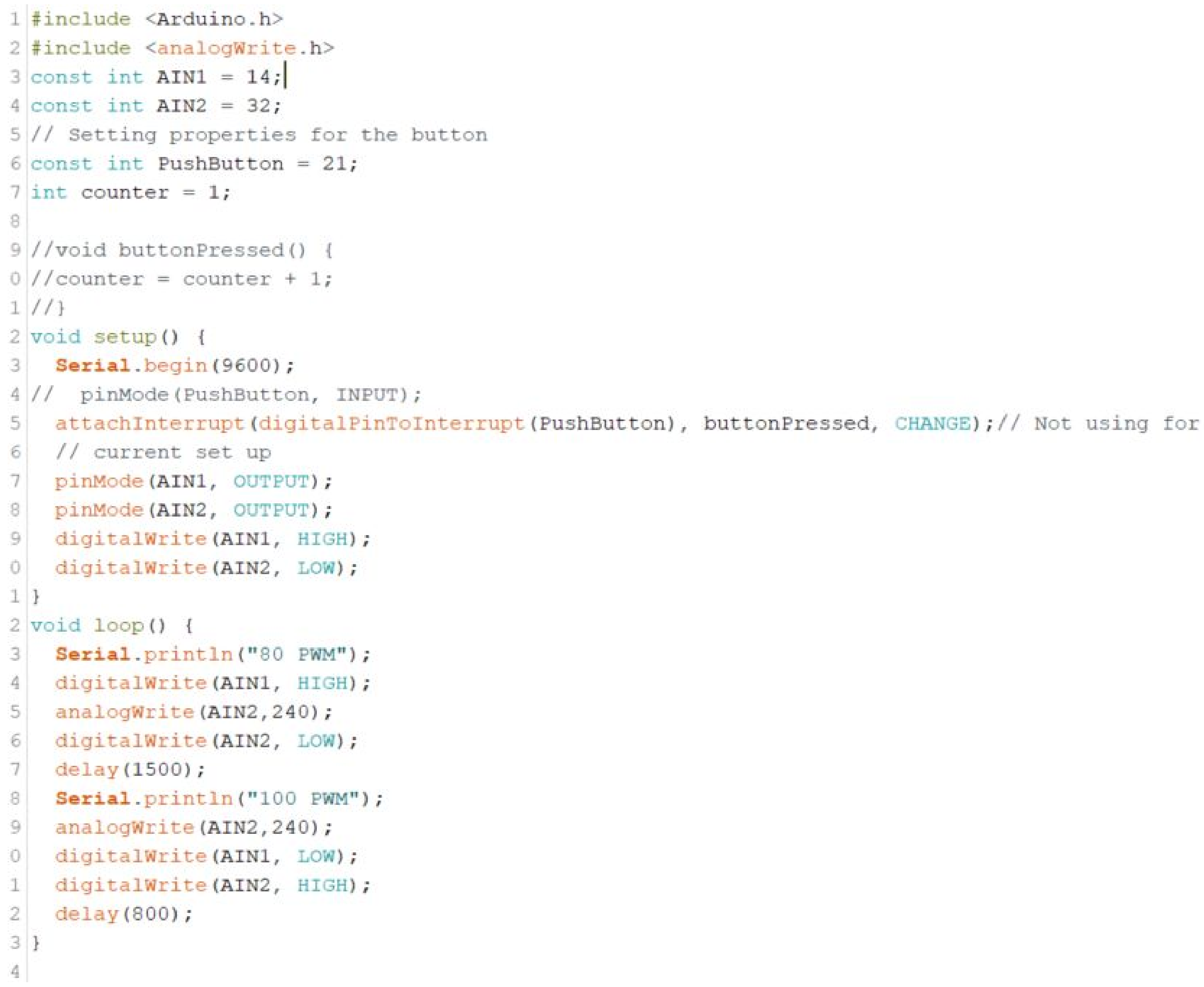


Figure 5. Finite State machine for open loop control

The behaviour of this system is very simple. It's either rotating at a fixed speed clockwise or counter clockwise. There are no sensors that provide the system with feedback to adjust the effort of the motor. The difference in the time the motor is spinning in a particular direction was dictated from the motor needing extra time to lift the target because the motor can't produce enough torque instantaneously. As a result, to do the same amount of work as if I had a more powerful motor, the weaker motor needs more time. The Arduino IDE code is located in Appendix I.

Appendix I: Arduino Code



```
#include <Arduino.h>
#include <analogWrite.h>
const int AIN1 = 14;
const int AIN2 = 32;
// Setting properties for the button
const int PushButton = 21;
int counter = 1;

//void buttonPressed() {
//counter = counter + 1;
//}
void setup() {
  Serial.begin(9600);
//  pinMode(PushButton, INPUT);
  attachInterrupt(digitalPinToInterrupt(PushButton), buttonPressed, CHANGE);// Not using for
  // current set up
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
}
void loop() {
  Serial.println("80 PWM");
  digitalWrite(AIN1, HIGH);
  analogWrite(AIN2,240);
  digitalWrite(AIN2, LOW);
  delay(1500);
  Serial.println("100 PWM");
  analogWrite(AIN2,240);
  digitalWrite(AIN1, LOW);
  digitalWrite(AIN2, HIGH);
  delay(800);
}
```