Micro control curtain with weather station

He Wang

December 2020

1 Introduction/Option 1

I have one piece of polyester fabric and a roll of yarn at home; I decided to build a windows curtain with an auto winching system with the motor and microcontroller. When I am working at my desk, I can use micro to open and closed my curtain, and I can manually adjust any height I want to. Besides that, I also made a weather station that will show the temperature and humidity in my room.

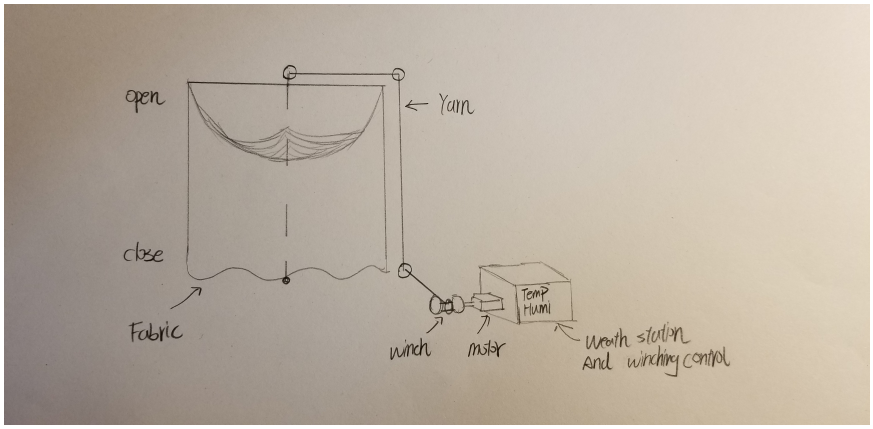

Figure 1: Mechanism Diagram

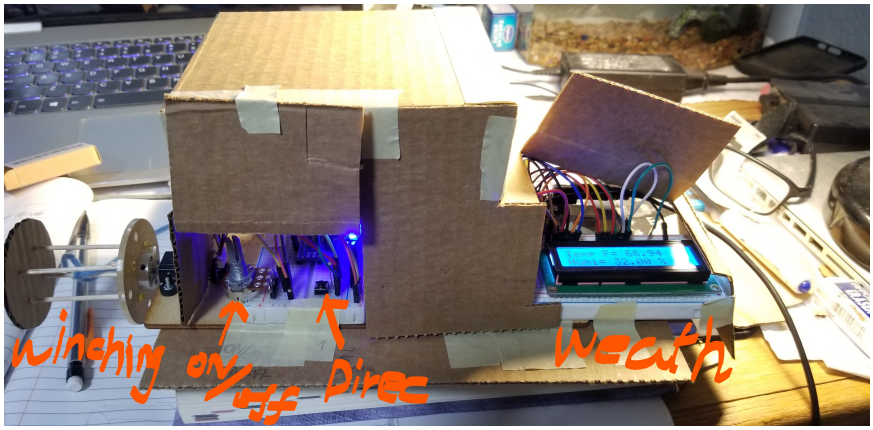

Figure 2: Weather station

2 Electron details

There are two separate microcontrollers. The left side is esp32 with motor drive, potentiometer, and motor. The potentiometer gives control of motor driving speed, and the push button is to switch the direction. One additional LED lights up the housing and indicates power is on; all the commend are processed by esp32. On the right side is a weather station. one DHT 11 temperature sensor and a LED display are connected to Arduino. Once the power on, it will display the temperature.

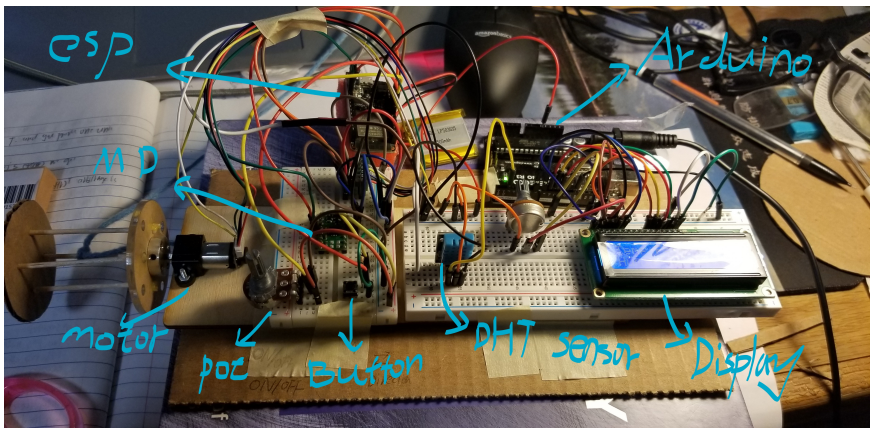

Figure 3: Inside parts

3 Circuit/design

The challenge of this design is to make sure the motor can winch up the fabric and release back. I made a pully with four wood sticks and cardboard and let yarn tight on one bar so that when it is rotating, it will winching up the tarn and pull up the fabric. The fabric is so light it won't drop even the motor rotate back, so I add a couple of weight at the bottom of the fabric. The circuit diagram and coding are similar to what we have done in the lab.

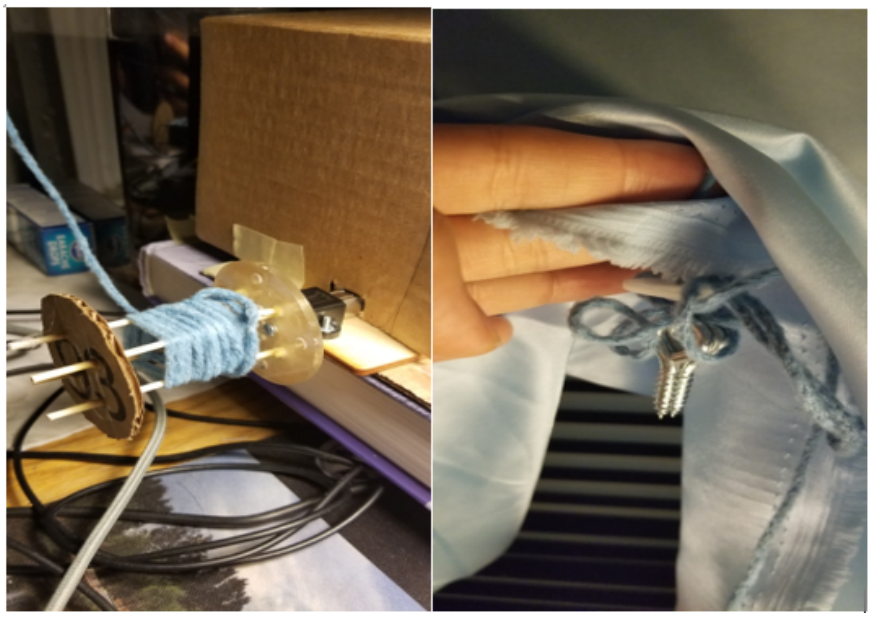

Figure 4: Winching and weighting

There are Wiring diagram and finite state machine,

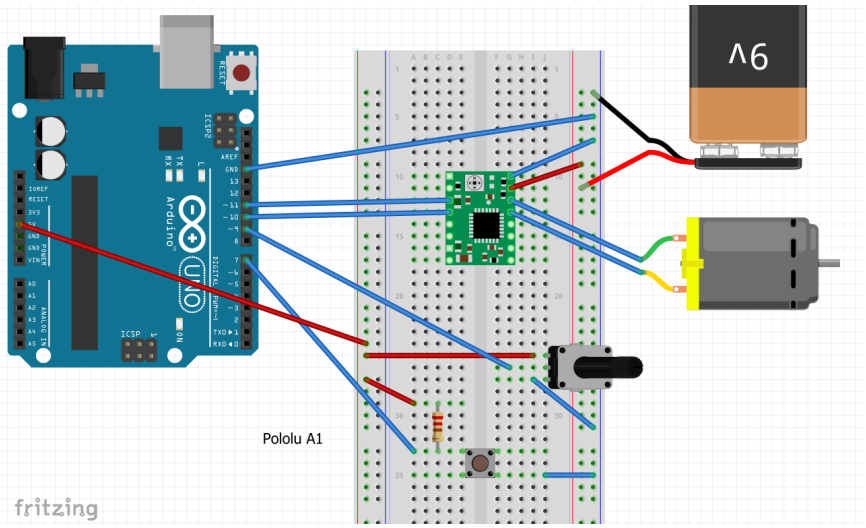

Figure 5: Wiring diagram

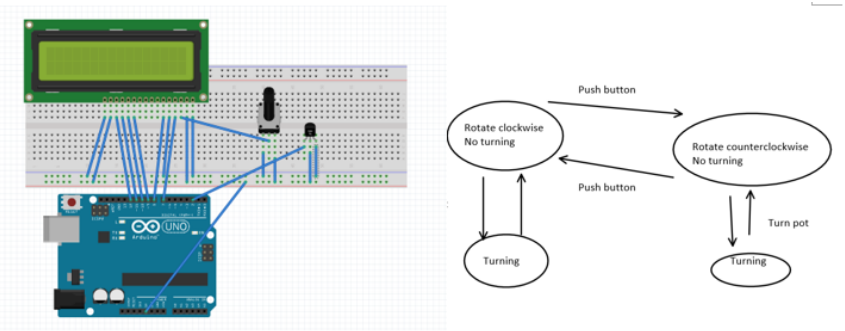

Figure 6: Finite state

```
#include <ESP32Encoder.h>
ESP32Encoder encoder;
int num, num2, motor1=A0, motor2=A1, pinread=33,
interruptPin=27, Debounce=100, Wd, Dw, K=10000, inteG, Dut, newcount;

volatile int totalCount=0, interruptCounter,
buttonTimer=0, buttonTimer_last=0, REdu=1, precount=1;

boolean state=0;
volatile boolean pressEvent=false;
unsigned long angularspeed;
const int freq=20000, ledChannel=0, ledChannel2=1, resolution=8;

void function(){
  buttonTimer=millis();
  if(buttonTimer-buttonTimer_last>Debounce){
    pressEvent=true;
    buttonTimer_last=buttonTimer;
    }
```

```
  }

void setup() {
  // put your setup code here, to run once:
ledcSetup(ledChannel,freq,resolution);
ledcSetup(ledChannel2,freq,resolution);

ledcAttachPin(motor1,ledChannel);
ledcAttachPin(motor2,ledChannel2);

pinMode(motor1,OUTPUT);
pinMode(motor2,OUTPUT);
pinMode(pinread,INPUT);
pinMode(interruptPin,INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(interruptPin),function,FALLING);
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  num=analogRead(pinread);
  num2=map(num,0,4095,0,255);
  switch (state){
    case 0:
      Serive0();
      if (pressEvent==true){
        state=1;
        pressEvent=false;
        totalCount=0;
        }
      break;
    case 1:
      Serive1();
      if (pressEvent==true){
        state=0;
        pressEvent=false;
        totalCount=0;
        }
      break;
    }
}

void Serive0(){
  ledcWrite(ledChannel,num2);
  ledcWrite(ledChannel2,0);
  }

void Serive1(){
  ledcWrite(ledChannel2,num2);
  ledcWrite(ledChannel,0);
  }
```