Building a Website Controlled Low Profile Walking Pad From an Old Treadmill

January 2024

Leo Berman
Temple University
Leo.Berman@temple.edu

Liam Coleman
Rochester Institute of Technology
RIT
Liam@megster.com

***Abstract*—This paper presents a step-by-step guide on how we converted the Proform XL Crosswalk treadmill into a Low Profile Walking Pad controlled by a website interface. Initially, we performed a diagnostic assessment of the main dashboard and determined how it communicates with the treadmill motor controllers. Next, we built an Arduino control system that receives commands via serial input. Finally, we designed a website to control the Raspberry Pi with a seamless UX.**

I. Procedure

1) Open up the control panel to examine the electronics. **!!!WARNING!!!** In our version of the treadmill the incline control and emergency stop were hooked up directly to 110 volts of AC so be very careful and unplug the machine while working.

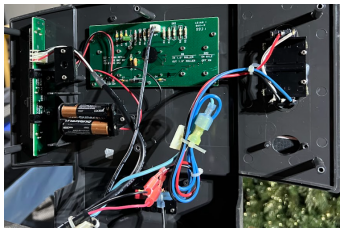

Fig. 1: Treadmill Control Panel

2) Test speed control, incline control, and emergency stop modules to ensure they are working properly.
3) Remove treadmill motor cover.

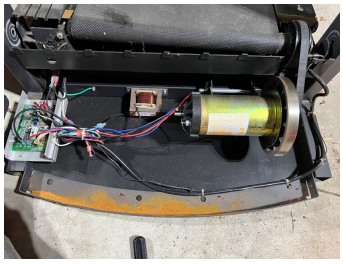

Fig. 2: Motor Controller and Motor

4) Feed the bundled control wires through the arm of the treadmill.
5) Remove the arms of the treadmill using an angle grinder.
6) (Optional) Place electric tape or other safeguard on sharp ends of your cut.

Fig. 3: Treadmill Without Arms

7) Implement a two channel relay module to control incline control. We chose a two channel optical isolated relay module in order to add an extra safeguard so that we don't connect the two output wires that connect to the incline motor. The logic itself comes directly from the Arduino which totally separates the Arduino and the 110 Volts of AC coming through the board so you significantly reduce the chance of frying your board!

Fig. 4: Two Relay Module

8) Implement a function to control voltage output of a PWM pin on an Arduino (we chose to map to a scale from 0-100 instead of the default 0-255).

```
void run(int percent_power) {
   int val = map(percent_power, 0,
      100, 0, 255);
   analogWrite(output_pin, val);
}
```

9) An optional feature for us was using a non-inverting operational amplifier to create a gain of 2.4 in order to map the 5 volt capacity of the Arduino onto the 12 volts of the treadmill.

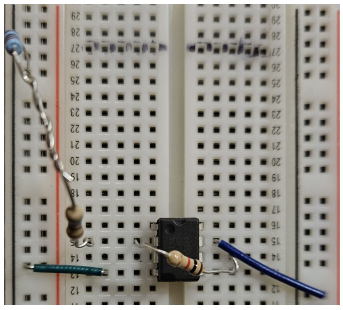

Fig. 5: Non-inverting operation amplifier (Not attached to Arduino or Power Supply)

10) Implement a function to control incline logic using two digital pins and the 5 Volt power supply from the Arduino.

```
// Incline logic to select which
// gates should be open
void change_incline(int input){
   if (input == 1){
      incline_up();
   }
   else if (input == 2){
      incline_down();
   }
   else if (input == 3){
      off();
   }
}

// Enable gate 1 (IN1) on the relay
void incline_up(){
   digitalWrite(IN1,LOW);
}

// Enable gates 1(IN1) and 2(IN2) on
// the relay
void incline_down(){
   digitalWrite(IN1,HIGH);
   digitalWrite(IN2,LOW);
}

// Disable gates 1(IN1) and 2(IN2)
// on the relay
void off(){\usepackage{xcolor}
\usepackage{environ}
\usepackage{eso-pic}
   digitalWrite(IN1,HIGH);
   digitalWrite(IN2,HIGH);
}
```

11) Implement parsing from serial communnication in the main loop.

```
// When input comes in
if(Serial.available()){

    // Parsing serial input
    String input_command =
        Serial.readString();
    String parses[2];
    int index =
        input_command.indexOf(' ');
    parses[0] =
        input_command.substring(0,index);
    parses[1] =
        (input_command.substring(index+1));

    // Checks if incoming statement
    // is speed(S) or incline(I)
        related
    if (parses[0] == String("S")){

        // set treadmill speed
        run(parses[1].toInt());
    }
    else if (parses[0] ==
        String("I")){

        // Set incline to up or down
        change_incline(parses[1].toInt());
    }
    else {

        // Stops the treadmill in case
        //of extraneous case
        run(0);
    }
}
```

12) Don't forget to initialize the relays and speed control!

```
void setup(){

    // Set the pins controlling speed
    // and incline logic to be output
    pinMode(speed_pin, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);

    // Begin communication with
    // serial input
    Serial.begin(9600);

    // Set the treadmill to off
    analogWrite(speed_pin,0); // Speed
    off(); // Incline
}
```

13) Next implement any interface with the Arduino, we went with two interfaces, one written in pure HTML and the other with Svelte.

Fig. 6: HTML Site Design (Try it out!)

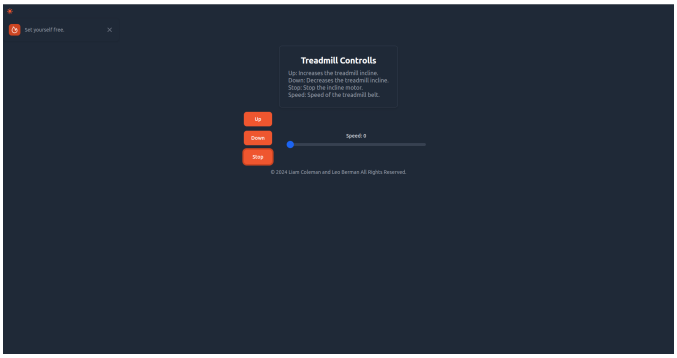

Fig. 7: Svelte Site Design (Try it out!)

14) For the full Arduino code check out this portion of our GitHub and for the Svelte and HTML, check out this portion of our GitHub

15) Finally we need to connect the Arduino and the website. We can do this by using a Raspberry Pi on our local network to host the website! We used Python3 Flask

16) First we initialize the serial port on the Raspberry Pi.

```
# sets the serial port
ser = serial.Serial(
   port='/dev/ttyACM0',
   baudrate=9600,
   parity=serial.PARITY_NONE,
   stopbits=serial.STOPBITS_ONE,
   bytesize=serial.EIGHTBITS,
   timeout=0
)
```

17) Next we build our sites!

```
# serving index.html to the main website
    endpoint
@app.route("/")
def index():
    return render_template('index.html')

@app.route("/fancy")
def base():
   return
       send_from_directory('my-app/build',
       'index.html')

# Path for all the static files (compiled
    JS/CSS, etc.)
@app.route("/<path:path>")
def home(path):
   return
       send_from_directory('my-app/build',
       path)
```

18) Finally, we initialize our endpoints using this format. If you're interested in our full Python Flask code check out this portion of our GitHub

```
# Generate endpoint for
@app.route('/speed/incline/<int:value>')
def speed(value):
   # Prints to the local console
   print(f"Setting speed/incline to
       {value}")

   # call set speed/incline function
       given value read from endpoint
   set_speed/incline(value)

   # return value read to api
   return f"{value}"
```

19) Connect everything together!

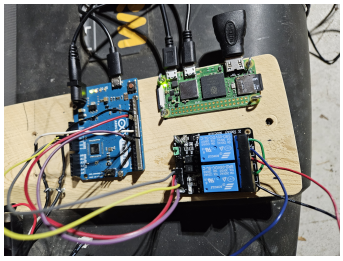

Fig. 8: Everything Connected

II. Launch Guide

Refer to our GitHub Readme for our quickstart guide.

III. Conclusion

In the bittersweet conclusion of this project, we've learned a lot about wiring, circuitry, embedded systems, web design, UX, serial interfacing, Git, GitHub, and much more. Most importantly, we have a working final product that goes above and beyond my initial expectations of the project. Is it possibly over-engineered? Perhaps, but it's a really cool project and if anybody has an old treadmill at home, it's one we would recommend trying. However, when trying this at home, just know that your treadmill could be different so there are no guarantees in terms of how it's controlled or if it will be the same as ours. Furthermore, experimenting with electronics is inherently dangerous and should be approached with caution and care.

Two things we hope to eventually implement are some more safe insulation around the electronics we have on the outside and to cover the sharp edges left over from angle grinding.

For all of our code please access our GitHub Repository and feel free to reach out to our emails with any questions you have in regards to our paper. Without any further adieu, here is the final product. (It's not super impressive to look at but here's a demo video of Leo doing math on the treadmill and Liam showing off the user interface.

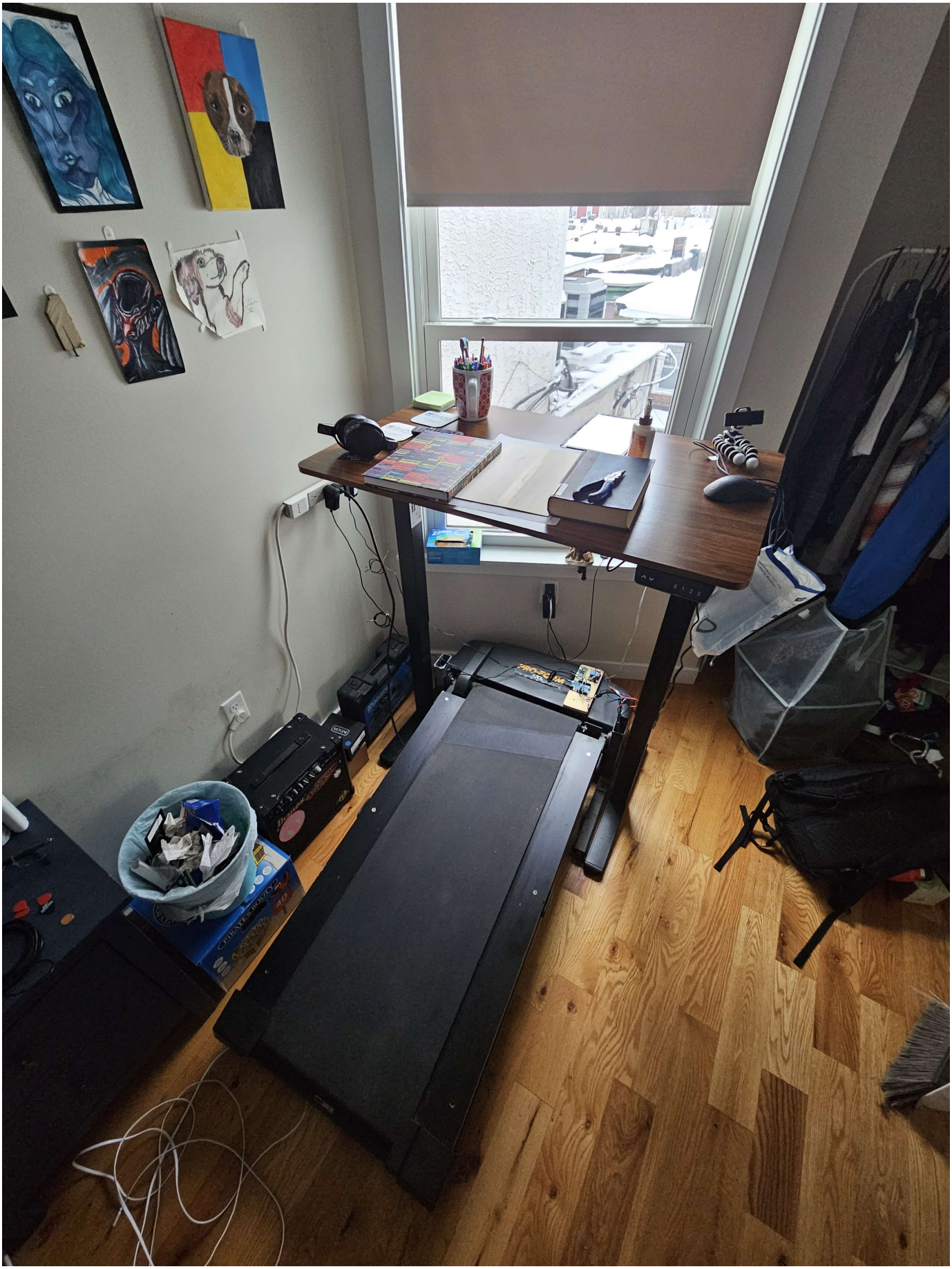

Fig. 9: Final Product