ECE 1000 Final Report: Curtain Clock

Andrew Queener, Weston Taylor, Tennessee Technological University, Cookeville, Tennessee. alqueener42@tntech.edu, wstaylor42@tntech.edu

***Abstract*—Waking up to sunlight is proven to be the best way to wake up in the morning. However, when using curtains for privacy at night, it is difficult to wake up to sunlight. The curtain clock is a device that will automatically open the curtains in the morning, allowing sunlight to enter and wake you up. It was designed to open slowly to simulate a sunrise. You can also use the curtain clock to close the curtains at night. This is all powered by a Raspberry Pi Pico W, stepper motor, and custom 3D printed mechanical components.**

Keywords—Home automation, Raspberry Pi, Micropython, Server-sent events, Sunlight exposure

I. Introduction

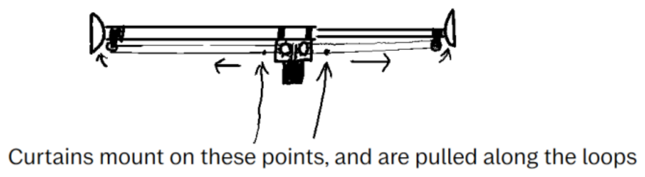

Curtains mount on these points, and are pulled along the loops

Figure 1: Basic design of the system

Curtains are a home appliance that many people have in their homes for privacy. With this added privacy, sunlight, which aids in the ability to wake up, is blocked. The team for this project consists of Andrew Queener and Weston Taylor, both students of computer and electrical engineering respectively at Tennessee Technological University. The goal of this project was a way to automatically slowly open the curtains in the morning at a set time through an online application and a Raspberry Pi Pico microcontroller with MicroPython, creating a sunrise effect and aiding in the feeling of wakefulness in the morning.

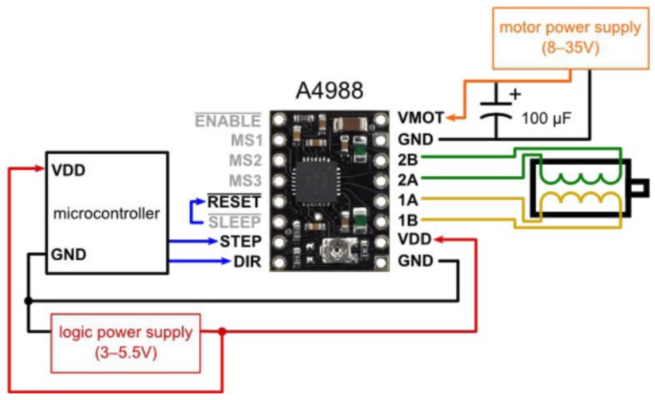
II. Background

The parts used for this project consist of a Raspberry Pi Pico W microcontroller, a 17HS19-2004S1 stepper motor, an A4988 stepper motor driver, a prototyping board to solder the driver board and pins to, a USB battery bank to power the Raspberry Pi (the team couldn't get the parts for a voltage regulator), elastic string for the pulley systems, 2 9-volt batteries to power the motor, and 3D printed mechanical parts that include a housing, mounts, gears, and pullies. Online resources that were used include the MicroPython documentation [1], the 17HS19-2004S1 stepper motor datasheet [2], an A4988 driver pinout sheet [3], and the Mozilla server-sent events documentation [4].

III. Project Description and formulation

The basic circuit design involves the Raspberry Pi Pico W pins 0, 1, 2, and ground. The driver board requires an 8–32-volt connection to power the motor, which is connected back to the ground of the 9-volt batteries. It also requires a 3-5-volt connection for the control. Pin 2 connects to the 3-5volt connection on the driver board to supply the voltage. When pin 2 is high, the driver board can be controlled. Pins 1 and 2 control the step and direction pins respectively. The Pico was programmed with MicroPython and used Server-Sent Events to send data between the Pico and Web Dashboard. The Web Dashboard hosted the event stream route which is what the Pico was connected to and listened in a continuous TCP connection with an SSL wrapped socket. The bytes were streamed and then processed to see whether it should go backwards or forwards a certain amount of steps. The dashboard allowed configuring the step counts, time to automatically open/close, and the creation of temporary access tokens for demonstration purposes.

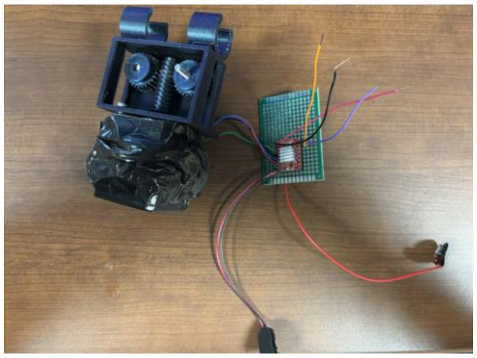
Figure 2: Circuit wiring and connections [3]



IV. Discussion and results

This project ended up being far more difficult than initially expected. The team ended up realizing too late that the 9 volt batteries often struggled to supply the needed power for the motor for long. Connection issues to the server for controlling remotely in environments were tested also ended up being a big hassle. In future renditions, better mounting for additional components would be a must, as well as an improvement to the power supplied to the stepper motor. It often would slip or not be able to provide enough torque required to spin the pullies. On top of this, the stepper motor used was extremely large and heavy to be mounted on a pivoting curtain rod. Due to this it had to be mounted on the bottom in use a worm gear spin the 2 different pullies, which helped with torque, but made the system very slow. In the team, Andrew Queener programmed the networking frontend and backend, as well as the flashed program on the Raspberry Pi itself using MicroPython, and design. Weston Taylor created the design and renditions, circuit design, wiring, and the 3D modelling of the hardware.

Figure 3: Completed device. The Raspberry Pi Pico is missing due to it being returned before the photo was taken, the 4 exposed wires all connected to pins 0, 1, 2 and ground of the Pico.



V. Conclusion

The purpose of this project was to use a microcontroller to be able to curtains in the morning to allow sunlight in. While the project was very finicky at times, when properly mounted to a window it did work well. The main restraints of the project were the size of motor, the poor supply of power the 9-volt batteries supplied, and the at times poor connectivity to the control server. All in all, this project taught us a lot, from how stepper motors work, to digital circuit logic, and prototyping, as well as the challenges vibration and friction can play into the merging of mechanical and electrical components.

References

1. MicroPython documentation - https://docs.micropython.org/en/latest/
2. 17HS19-2004S1 stepper motor datasheet - https://www.omc-stepperonline.com/download/17HS19-2004S1.pdf
3. A4988 driver pinout sheet - https://www.tme.eu/Document/25459777e672c305e474897eef284f74/POLOLU-2128.pdf
4. Mozilla server-sent event documentation - https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events