Automated Smart Blinds

Ryan Guidice
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO 80523 USA
rguidice@colostate.edu

Andrew Helmreich
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO 80523 USA
achelm@rams.colostate.edu

***Abstract*— Opening and closing the blinds in a room is a monotonous task completed far too often in modern life. In this paper we will explain our attempt to erase this hindrance from modern life. We will show the method by which we have developed a system utilizing a single board computer to do so. Hardware used includes a Raspberry Pi (3B for development, Zero W for production), simple electrical circuit to sense light, stepper motor, and 3D printed blind attachment. In addition, Python scripts were written to control the motor and circuit. A Python webserver was also implemented to run these scripts, and a frontend web application was created to improve user experience. We will describe the difficulties encountered in our progress to our solution, as well as further possible improvements to our project for a later date.**

Keywords—Smart blinds, home automation, Python, Flask, React, Raspberry Pi

I. Introduction

While not a horrible inconvenience, opening and closing the blinds every morning and evening can be an annoyance. In our project, we attempted to resolve this issue. We used a Raspberry Pi single board computer to control a stepper motor to open and close the blinds. Our solution allows the blinds to be opened or closed through user input or automatically, based on ambient light received by a sensor. Python functions were written to open or close the blinds as well as activate the automatic control based on a sensor circuit. In order to improve the user experience, a user-facing web application was developed using a Flask (Python) backend and a React.js, HTML, and CSS frontend. Using these various technologies, we were able to complete our project and fulfill the goals we set out to accomplish. We believe this device could be a great addition to the modern smart devices present in many houses.

We chose to use a Raspberry Pi 3B for the development process and a Raspberry Pi Zero W for the deployed product, as the 3B is faster and easier to write code on, but the Zero W is much smaller and still Wi-Fi connected. We wrote our initial scripts to interface with the stepper motor using Python 3, so we decided Flask was the right web backend choice since it is also in Python. From there, we decided on React.js and HTML/CSS to build the frontend due to our experience with it in the past.

II. Problem Characterization

This project had four main issues to resolve. First, the project required a circuit to sense the ambient light of a room. Second, functions to open, close, and automatically control the blinds were required to complete the project. Third, the mechanics of a mounting mechanism for the machine had to be figured out, and the stepper motor must be able to drive the blinds shaft. Finally, a webserver and web application were required to provide a user-friendly interface for users to interact with.

The first goal of the ambient light circuit was that it needed to be powered off the Raspberry Pi's 5V power supply to save space. From there, a change in the ambient light of the room or out a window must be detected by a change in the characteristics of the circuit, which would then be sent as input over the general-purpose input/output (GPIO) ports of the Raspberry Pi. This detection of a change in light must also be easily processed in the Python script.

The open, close, and autocontrol functions provided a second obstacle to overcome. All three functions required use of the GPIO on the Raspberry Pi. In order to implement the open and close functions, an understanding of the necessary signals to drive the motor stepper was required. The autocontrol function also required use of the GPIO in order to receive the data from the light sensor circuit. Our design of these functions also introduced some user experience-oriented problems. For example, in the autocontrol function, at a specified change-over point between when the blinds should be open and closed, two consecutive readings may cause the blinds to open and then close. This could happen repeatedly and be highly annoying for a user. In addition, opening the blinds when they are already open could break the blinds, and thus should not be possible. Our solution to these issues will be described in section 3.

Next, we had to figure out how to physically arrange the stepper motor and Raspberry Pi in order to open and close the blinds. This process involved two key mechanical problems: how to keep the Raspberry Pi and stepper motor suspended in the air, and how to make the blinds shaft turn with the stepper motor.

Finally, the web application posed many significant difficulties in our project. We started by writing our code for the stepper motor locally, then later realized that we had to migrate it to Flask using requests from a React frontend. The key difficulty in implementation of the web application was to synchronize the state of the frontend with the data stored in the backend.

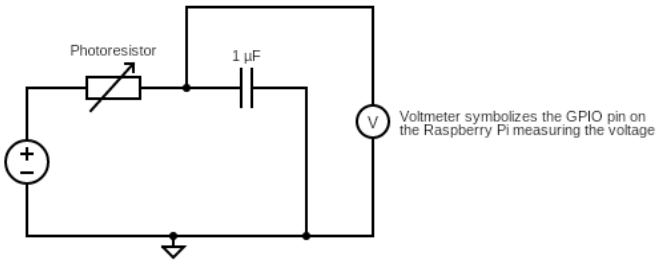

Fig. 1. Schematic of the light-sensing circuit

III. Proposed Solutions and Implementation Strategy

In order to build our ambient light sensor circuit and parse the data in Python, we referenced an excellent guide by Gus on the PiMyLifeUp website [1]. The circuit uses only two components, a photoresistor that changes its resistance based on the amount of light the sensor absorbs, and a 1uF capacitor. A diagram of this circuit is shown in Figure 1. The voltage across the capacitor then changes based on this resistance set by the photoresistor, and we can measure the time it takes for the capacitor voltage to charge up to a digital value of 1. This is what is measured through the Raspberry Pi's GPIO pins. Our Python code then counts from 0 until the signal goes high, and this is now an integer value that is dependent on how much ambient light is available to the sensor.

The Python script controls the stepper motor through a series of outputs over the GPIO ports. The stepper motor controller has 4 control pins and 2 power pins, and requires 5V to run. In our script, we made two sets of arrays containing the correct opening and closing output sequences for this stepper motor based on a guide that we found by Keith Weaver [2]. Our open and close functions work by iterating over a range that we determined was the distance required to set our blinds between open and close, and sending the movement values from the arrays sequentially. A time delay between each sent pulse controls how fast the motor is spinning, and we found that delaying for 0.0007 seconds between each pulse gave us the fastest possible speed without compromising on the torque of the motor.

In the autocontrol function, we utilize the ambient light circuit and the integer value it produced. We found that values below 1700 represented ambient light in a room during the day, and above, ambient light in a room at night. This is the reading we used throughout the function. The autocontrol function then runs as an infinite loop until the user disables it. In order to prevent the constant switching problem described in section 2, a counter was created. This counter requires five consecutive readings in the same range in a row to open or close the blinds. Once the five readings are reached, the correct open or close function is called, and the blinds will open or close.

In order to drive the blinds with the stepper motor, we designed and 3D-printed an adapter piece to attach the end of the stepper motor to the blinds shaft, making them rotate as one. Glue was used on the inside of this adapter piece to secure the blinds shaft. Once the stepper motor could rotate the blinds shaft, we had to find a way to secure its base so it would not rotate in mid-air.

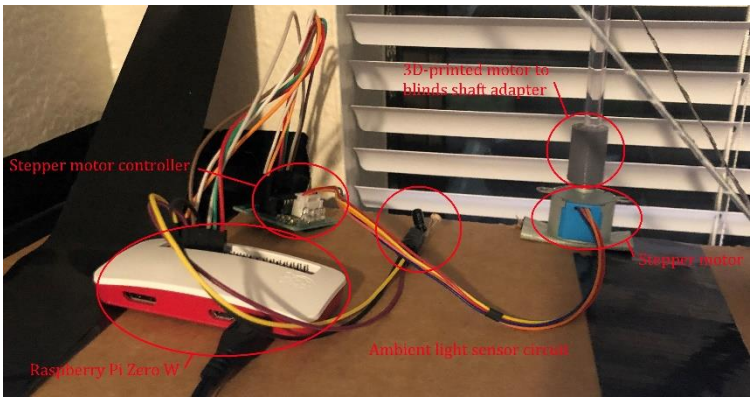

Fig. 2. The current mechanical setup of our device

We achieved this by taping a piece of cardboard to the wall at the appropriate height, then securing the Raspberry Pi, stepper motor driver, and stepper motor to the cardboard. Our full setup, including the 3D-printed adapter, can be seen in Figure 2. This is far from a "professional" solution, but due to lack of a 3D printer at the end of the semester, this is what we ended up going with to make the project work. A higher quality and removable solution is described in section 5.

The implementation of the web application provided the largest issues in our project. This webserver is hosted locally, and a user must navigate to the IP address of the Raspberry Pi and port 3000 to see the UI. The final user-facing UI we designed can be seen in Figures 3-5. The UI prevents a user from opening/closing the blinds twice in a row or opening/closing the blinds when autocontrol is enabled, which were problems we had trouble fixing in the backend Python code. When these buttons are pressed, a fetch request is sent to the Flask webserver, which then runs the corresponding script. In order to run autocontrol correctly however, the function must know if the blinds are open or closed. To solve this, a local file on the Raspberry Pi was created, storing a 1 to represent the blinds being closed, or 0 to represent the blinds being open. The UI is also rendered based on the value read from this file. The useState and useEffect functions of React.js were used to update the state of the open/close sliding button to render in the correct position on load of the webpage, and on each press of the button. This allows a user to visit the web application on separate devices and have the interface settings still be correct. This problem plagued our development process, and fixes that helped us along the way were discovered on StackOverflow and CodeSandbox.io [3][4].

The libraries we used in our Python code were the GPIO and Flask modules. These modules were used to simplify GPIO communication and webserver tasks respectively. The rest of the remaining code is ours, unless otherwise denoted by references in this paper or in the comments of our code.

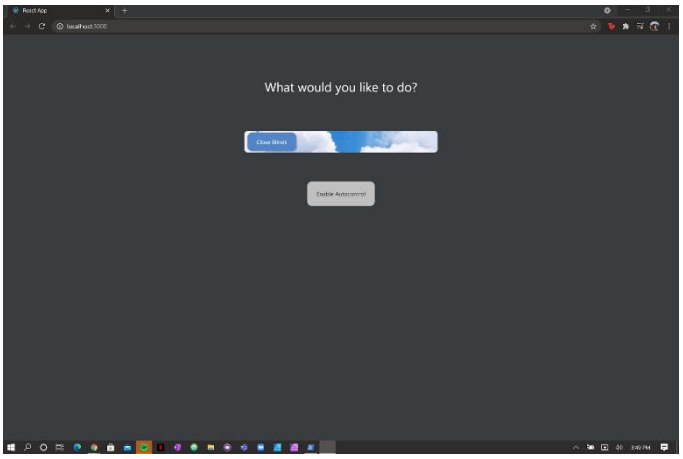

Fig. 3. Web application UI when the blinds are open

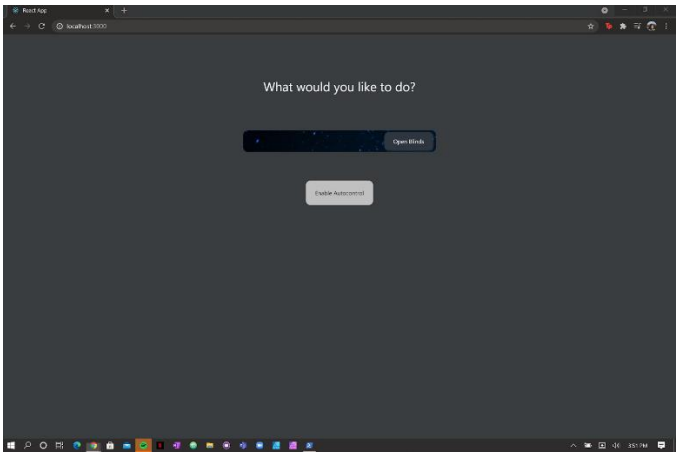

Fig. 4. Web application UI when the blinds are closed

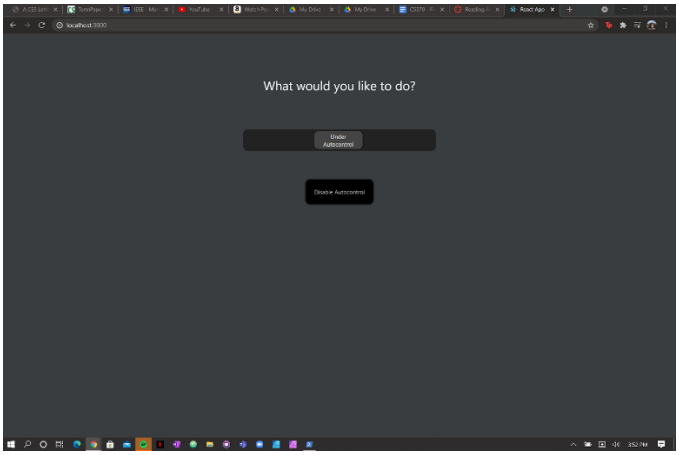

Fig. 5. Web application UI when the blinds are under autocontrol

IV. Evaluated Attributes

The two main attributes we investigated for this project were the security and marketability of our device. On the security side, we do have a security vulnerability since any user on the network can control the blinds by visiting the webpage. This could be fixed by implementing an authentication database that would manage authorized users. This vulnerability is not major though, as attackers would have to already be on the network and know what IP address and port to connect to, and even then could only adjust a set of blinds.

As for the marketability of our device, we think this could be a valuable addition to the wide range of smart home devices available today. To make the product more visually appealing, we would move the circuit components to perfboard and enclose them in a 3D-printed case alongside the Raspberry Pi. As for cost analysis of this device, our per-unit costs are:

- Raspberry Pi Zero W: $10
- Power supply: $7
- MicroSD card: $10
- Stepper motor and controller: $5
- Ambient light circuit components: $1
- Estimated 3D-printing filament: $5

This results in a total cost of $38. Assuming an initial fixed cost of $250 for a budget 3D printer, we have plotted the breakeven line of fixed costs, manufacturing costs, and sale value in Figure 6. For example, if we were to charge $60 for our product, we would only need to sell 12 units before breaking even. With the recent explosion of the smart home market, we think a cost of $40 would be more appropriate however, and the graph shows that a $40 sale cost drastically increases the number of units that must be sold to break even. This means that we must decrease our operating costs in order to sell our device at the target price point of $40. This could theoretically be achieved by purchasing components in bulk or switching to cheaper versions of the same components.

V. Future Improvements

Our project has fulfilled the goals we set out with, but it could still be improved. A custom-designed case and more compact and attractive attachment to the blinds may make the product more marketable. Our plans for this part included a 3D-printed mount that attaches to the wall next to the window then extends an arm out with the stepper motor to latch into the top component of the blinds. This would remove the blinds shaft from the equation and drive the blinds from the very top of the assembly for a cleaner look.

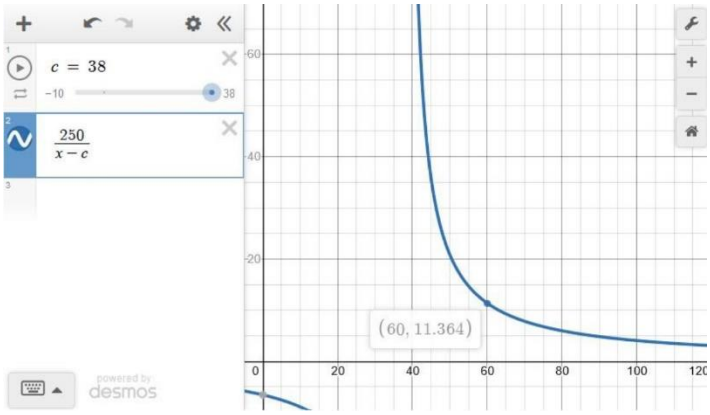

Fig. 6. Cost analysis of our device

Additionally, the motor is only controllable from the same network as the Raspberry Pi and has no user authentication. We felt this was sufficient, as no major data breaches or harm can occur with our device. However, this does not account for the fact that other members of a household may close or open the blinds without the user's consent. We believe an authentication system as described in section 4 could resolve this problem.

VI. Conclusion

This project furthered our experience in many areas of computer science. We had to learn how to get our local Python code running on a Flask webserver, then we had to figure out how to communicate with it from a React frontend. This process was a great experience for our future careers, as the interconnect between the frontend and backend of the web development experience is imperative in the real world.

In addition, communication between hardware and software was also explored in this project. Our project required extensive use of the GPIO on the Raspberry Pi. Using this package, we were able to communicate with external devices, taking input from the circuit, and sending output to the stepper motor. While the GPIO package obscures deeper level interactions between the operating system and the hardware, we were still able to draw connections between the content learned in CS370 and this project.

In conclusion, as well as furthering our experience with CS370 content and software development, we were able to achieve our goals in this project. We were able to create a webserver, web application, Python scripts, and a simple circuit to control the opening and closing of blinds. Our project runs on the Raspberry Pi Zero W single board computer. The blinds can be remotely controlled to open or close, and an autocontrol mode allows them to adjust based on an ambient light sensor circuit.

References

[1] Gus. (2016, January 30). *How to Setup a Raspberry Pi Light Sensor*. Pi My Life Up. https://pimylifeup.com/raspberry-pi-light-sensor/

[2] Weaver, K. (2018, May 28). *Controlling Stepper Motors using Python with a Raspberry Pi*. Medium. https://keithweaverca.medium.com/controlling-stepper-motors-using-python-with-a-raspberry-pi-b3fbd482f886

[3] zemirco. (2018, November 14). *4zq852m7j0*. CodeSandbox. https://codesandbox.io/s/4zq852m7j0

[4] zemirco, & Khatri, S. (2018, November 14). *javascript - How use async return value from useEffect as default value in useState?* Stack Overflow. https://stackoverflow.com/questions/53298626/how-use-async-return-value-from-useeffect-as-default-value-in-usestate