Aliya Chambless, Tyler Jacobson, Stephen Sun
Prof. R. Iris Bahar
ENGN 1931I: Design of Robotic Systems
22 April 2021

Final Project Report

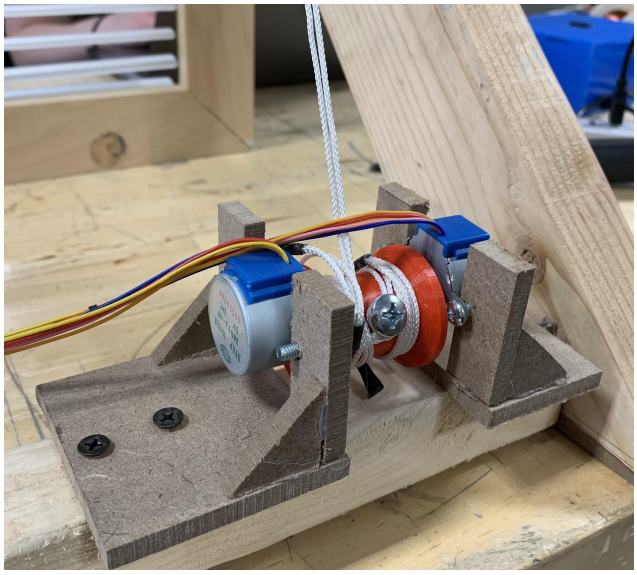
1. A discussion of your design, including its unique and salient features.

Mechanical Build

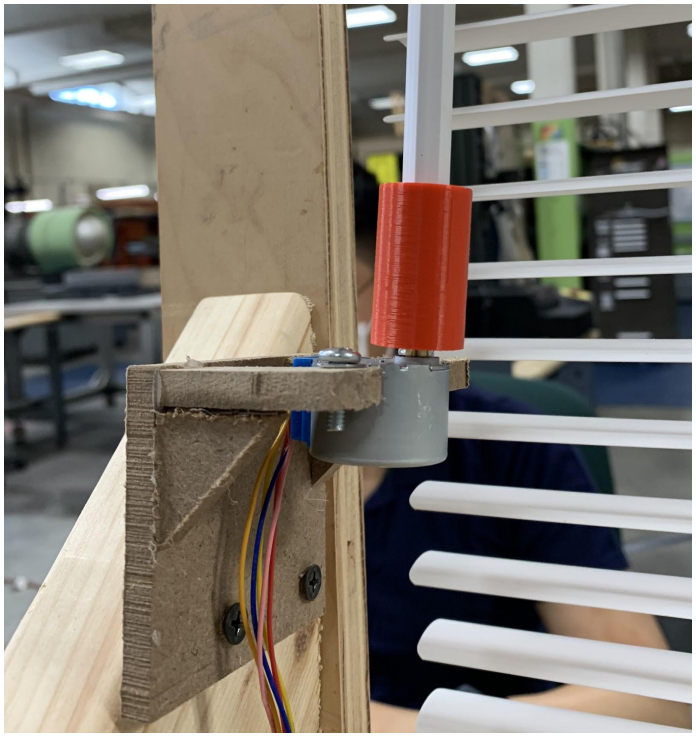
The strategy for physically building a set of smart window blinds was to incorporate as many features from an existing set of blinds as possible. We constructed a basic wooden frame with two platforms bolted to the ends of the blinds to support the blinds from underneath. We also disabled the latching mechanism by hot gluing the latch in the open position. As we found out, the string required to hold the blinds at a set height was held firmly in place by the winch mechanism and did not add any additional strain on the stepper motors, and adding another motor to latch and unlatch the blinds would have added more, unnecessary complexity to the electrical and programming sides.

Stepper Motors

We had 2 main movement functions that we wanted to implement: the raising/lowering of the blinds, and the opening/closing of the blinds which is achieved by the rotation of the slats. The raising and lowering of the blinds is achieved by pulling on the main string of the blinds. We used a winch mechanism that winds up the string and thus raises the blinds. A pair of stepper motors are attached to the winch to rotate this winch.



For opening and closing the blinds, we directly attached a stepper motor to the tilt rod by 3D printing a connector piece. The stepper motor is able to rotate the tilt rod smoothly and in both directions to open and close the blinds.

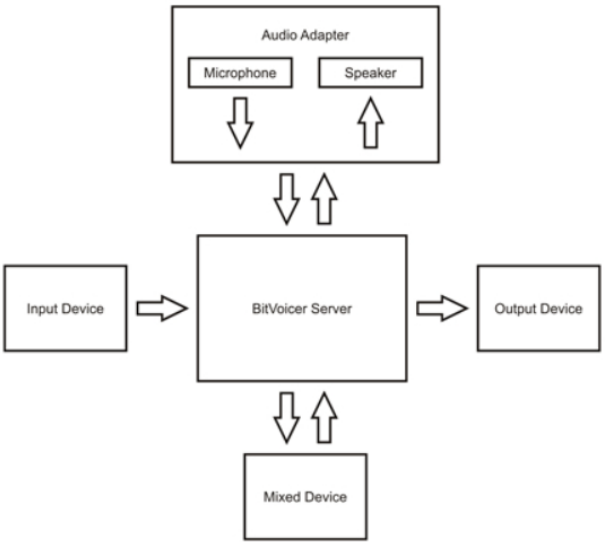


Voice Control and Wireless Communication

The voice control acts as the main control mechanism for our entire smart blinds setup. The commands for the voice control include raising/lowering and opening/closing the blinds, and setting an alarm. The full list of functions is detailed below in the schematic section.

The voice control software that we used was BitVoicer Server. This program makes use of Windows' speech recognition and speech synthesis tools to run speech recognition on recorded audio clips. It can also read out phrases out loud that you set in the program.

BitVoicer provides an Arduino library that allows the Arduino to interface with the program via the Serial port. The PC's system microphone, or a microphone connected to the Arduino can be used for speech recognition. The library allows for two way communication, with first any audio from the Arduino being sent to BitVoicer Server, and then whatever data we set up to send on a voice command being recognized being transmitted back to the Arduino.



The above diagram shows a simplified schematic of the communication flow of BitVoicer server. In our case, we only used one way communication, so the laptop microphone acted as the input device. We set up specific phrases that will trigger functions on the Arduino that each correspond to one function of the smart shades, for instance raising or lowering the blinds. An activation keyword is also required at the start of the command, and our activation word is "Helios". Thus BitVoicer Server parses any voices received from the microphone for the activation word and command phrases. If a command is recognized it then sends binary data to the Arduino, which then calls functions corresponding to the command.

We also made use of the HC-12 wireless communication chip to facilitate communication between our main Arduino, which controls the stepper motors and is connected to a laptop for voice control, and the alarm Arduino, which acts as the alarm clock. We defined a specific format for sending and receiving data between the 2 Arduinos, thus allowing for 2-way communication between them.

Alarm Clock

The alarm clock was designed to control the blinds and allow the user to more easily snooze and stop the alarm. The most important aspect of the clock is the lcd screen, which in addition to displaying the current time also indicates if an alarm is set and what time it is set for. When the alarm clock receives a time increment message from the blinds, it calculates the correct alarm time to display. The lcd screen brightness is controlled by a potentiometer, but the lcd screen was only readable at a high brightness so we did not expose the potentiometer to the user.

To keep the design as simple as possible, we only included two buttons, which increment the hour and minute to set the clock. However, if the alarm is going off, the buttons snooze the alarm for 9 minutes or stop the alarm completely. One minute before the alarm activates, the alarm clock sends a message to the blinds to open, then sends a message to raise the blinds when the alarm goes off. If the user snoozes the alarm, the clock will first send a message to lower the blinds then close them.

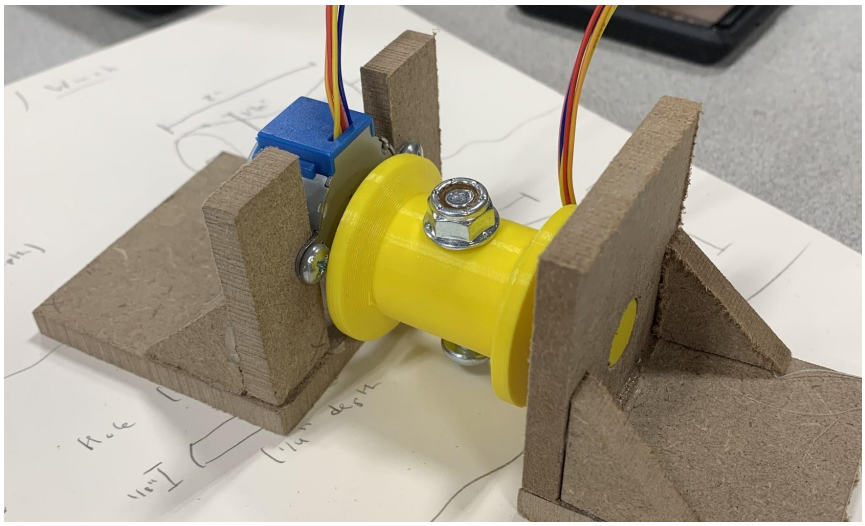
Modified Tone library

To incorporate timer interrupts into our project, we merged Ken Silverman's audio code into the default Arduino tone library. By integrating the wave look up table code into the tone library code, we were able to use different wave types for the tone() function. While we tested various wave types, we found that the sawtooth and triangle waves generate very distorted sound, hence we set the tone() function to use a sine wave.

2. The aesthetic and technical challenges, successes and failures. Which of the challenges and constraints did you anticipate, and which surprised you?

Mechanical Build

One of the first we faced was that the stepper motors were not powerful enough to rotate the winch and raise the blinds. We first designed and tested the winch mechanism with one stepper motor.



However, this winch was unable to raise or lower the blinds, even after increasing the voltage to a 9V supply. We then cut off the bottom 10 slats of the blinds to decrease the tension required to raise the blinds.

This was still unable to raise the blinds. We added a counterweight to the blinds to reduce the tension. While raising was somewhat successful, unwinding was not, as slack built up in the string behind the weight, and the blinds remained stuck in the same position.

We then performed some calculations (See section 5) to see if two stepper motors could be controlled simultaneously to raise the blinds. Confirming that they could, we redesigned the winch and connected it to two stepper motors, which was able to raise the blinds halfway. While the motors were still not powerful enough to raise the blinds the entire height of the frame (the

tension required increases as the blinds raise), this implementation was successful as a proof of concept.

Stepper Motors

In testing our stepper motors in project 3, we faced issues of getting the Arduino stepper library to work with interrupts, as the stepper code is blocking. While we did test other stepper motor libraries, they had similar problems. We thus decided to make use of 2 Arduinos and the HC-12 chip to communicate between them. This worked well, as now the current time display and interrupts are all off-loaded to the alarm Arduino, and the main Arduino only has to control the stepper motors.

However, once it became necessary that we needed 2 stepper motors turning together to control the winch mechanism, the fact that we were using the default stepper library meant that we were unable to control 2 stepper motors directly. A workaround that we discovered was that we could run 1 set of stepper motor pins to 4 pins on the Arduino, then split the 4 pins in parallel between 2 stepper motors. This worked to our advantage, as we wanted both stepper motors to turn at exactly the same rate and for the same number of steps, hence directly splitting the control pins meant that we would not face any misalignment between the 2 stepper motors. Also, since the 2 stepper motors were facing each other, we needed one motor to spin in the opposite direction of the other. This was easily resolved by simply reversing the 4 pins that plugged into one motor.

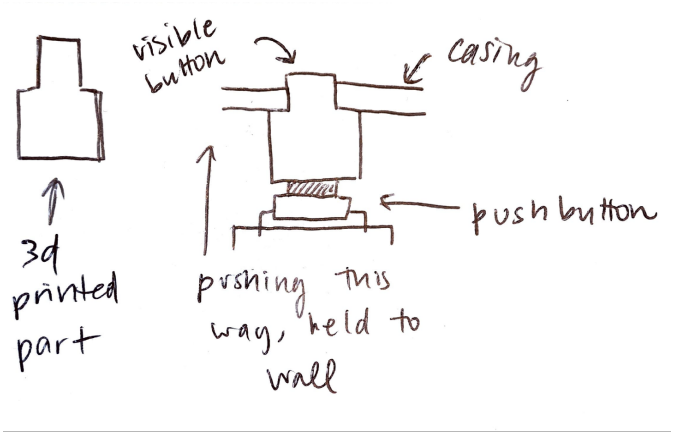
Voice Control

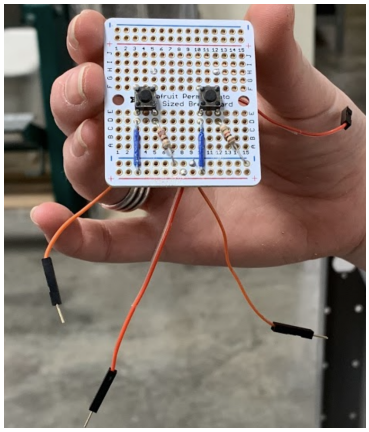
The main difficulty with debugging the voice control functionality was that since the BitVoicer Server library uses the USB serial interface to transmit data, the serial port on the Arduino is constantly occupied as the library periodically checks for new data from BitVoicer Server. This means that we were unable to easily debug if voice commands went through. A workaround was to use the built-in LED on the Arduino to indicate the state of the blinds - whether they should be raised or lowered.

Alarm Clock

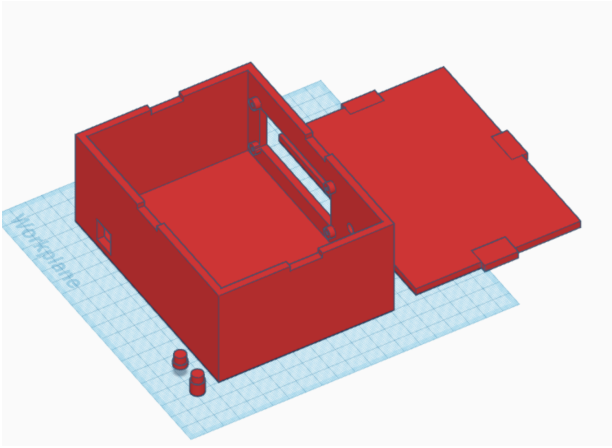
Keeping track of the 12 hour time proved tedious, especially since it was changed in so many places. We ran into bugs when switching from am to pm at 12 and resetting the hour to 1 at 13, especially when an alarm was set and the time passed minute “60” without a chance to properly increment. We had to use slightly different logic when using the hour and minute buttons, because incrementing the minutes past 59 should not change the hour while in time-setting mode. We also had to change the one second delay() to millis() logic because it was preventing the buttons from working while in the delay.

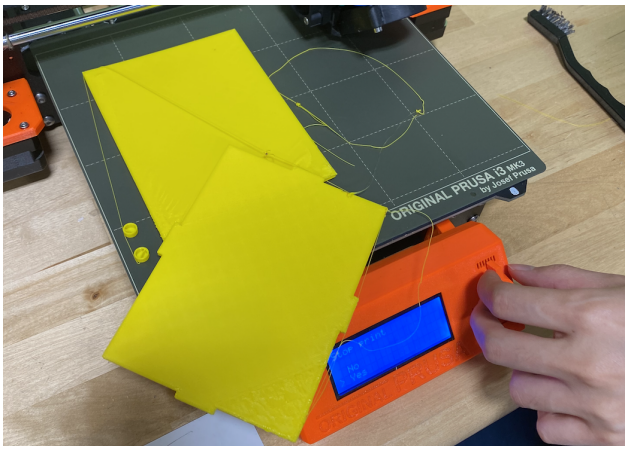
We also ran into material challenges with the buttons - we did not have any raised push buttons that would stick out of the casing, so we had to design our own buttons that would wedge between the buttons. We also had to solder the buttons to a ¼ size breadboard so it would fit vertically in the box beside the lcd screen.





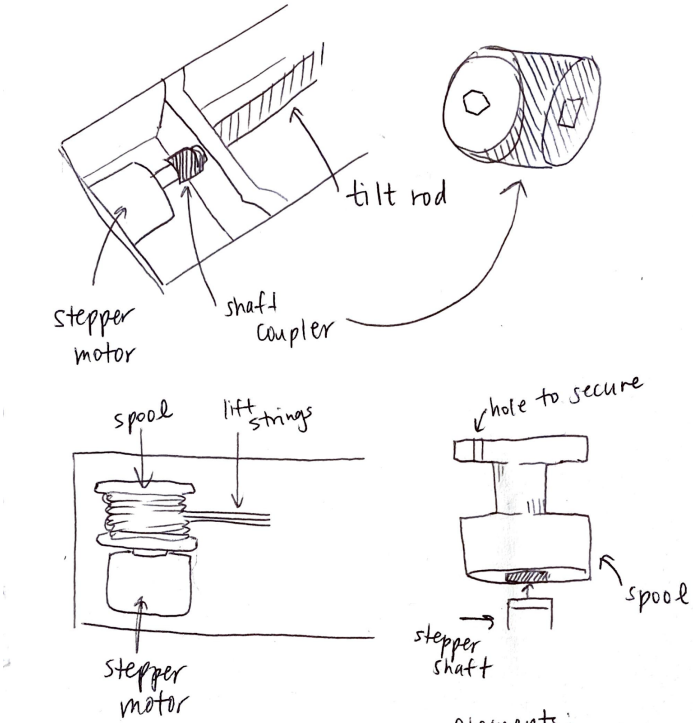
Our original CAD design also had to be scrapped after the 3D print curled and failed. An important part of the 3D model was the extended screw supports that allowed the lcd screen to be flush with the wall of the casing while the back was supported.

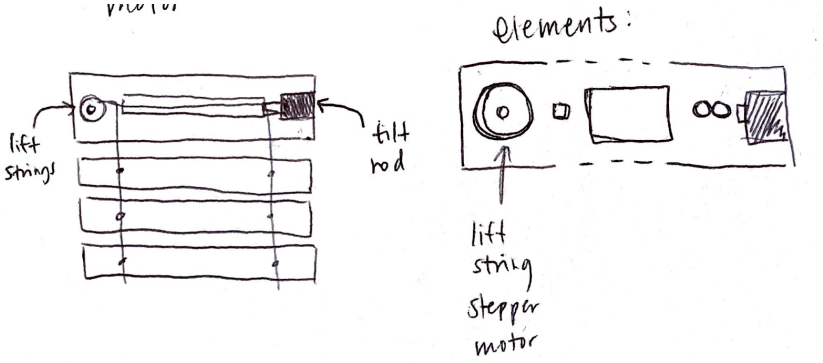


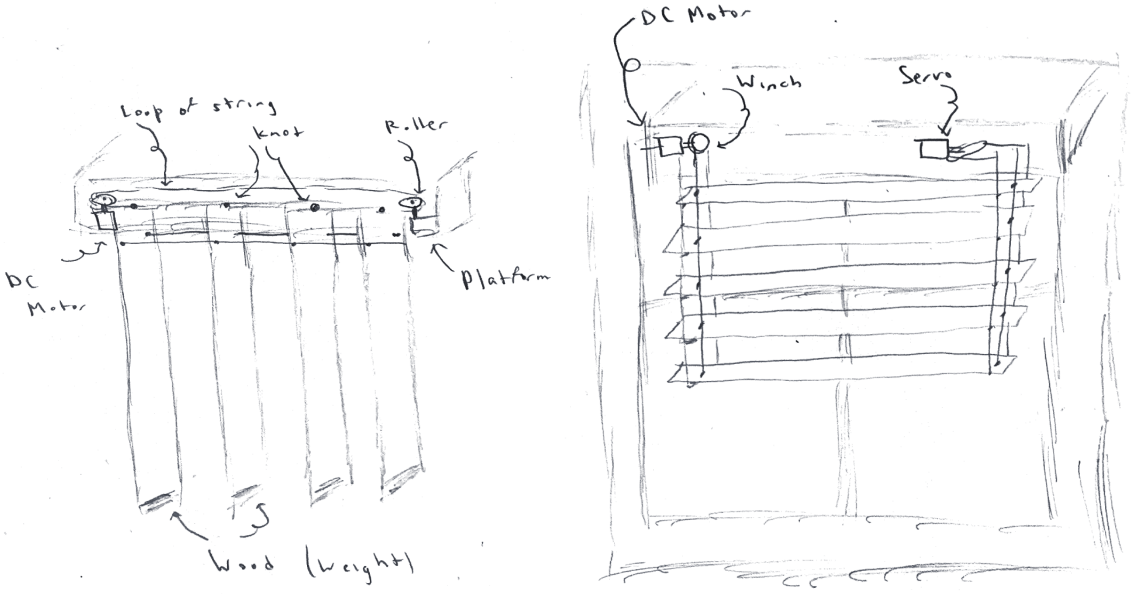


Instead, we had to lasercut the box, and change the hole for the lcd to include a notch so the entire screen could slide through, and the small bolts could reach the backboard of the screen. Finally, we had screen flickering issues when the buttons were pressed, but after checking the wiring we realized it was a power issue and used a 9V plug instead of a laptop for power.

3. Annotated sketches: show the evolution of the design and your design choices. Detail the ways you distilled your design ideas during the ideation phase: which decision frameworks did you use? Which factors influenced your initial design decisions?







Some of the early stage sketches of the design for the smart shades are shown above. We were originally unsure of which style of shades to use as the basis for our design – shades with horizontal slats, vertical slats, or shades with a single sheet of fabric. We eliminated the idea of using a single sheet of fabric after discovering that users want two types of control over the amount of light that enters a room (both control over the rotation of the slats and how many slats are visible). We decided to design our shades with the horizontally oriented slats for simplicity and ease with integration with the motors.

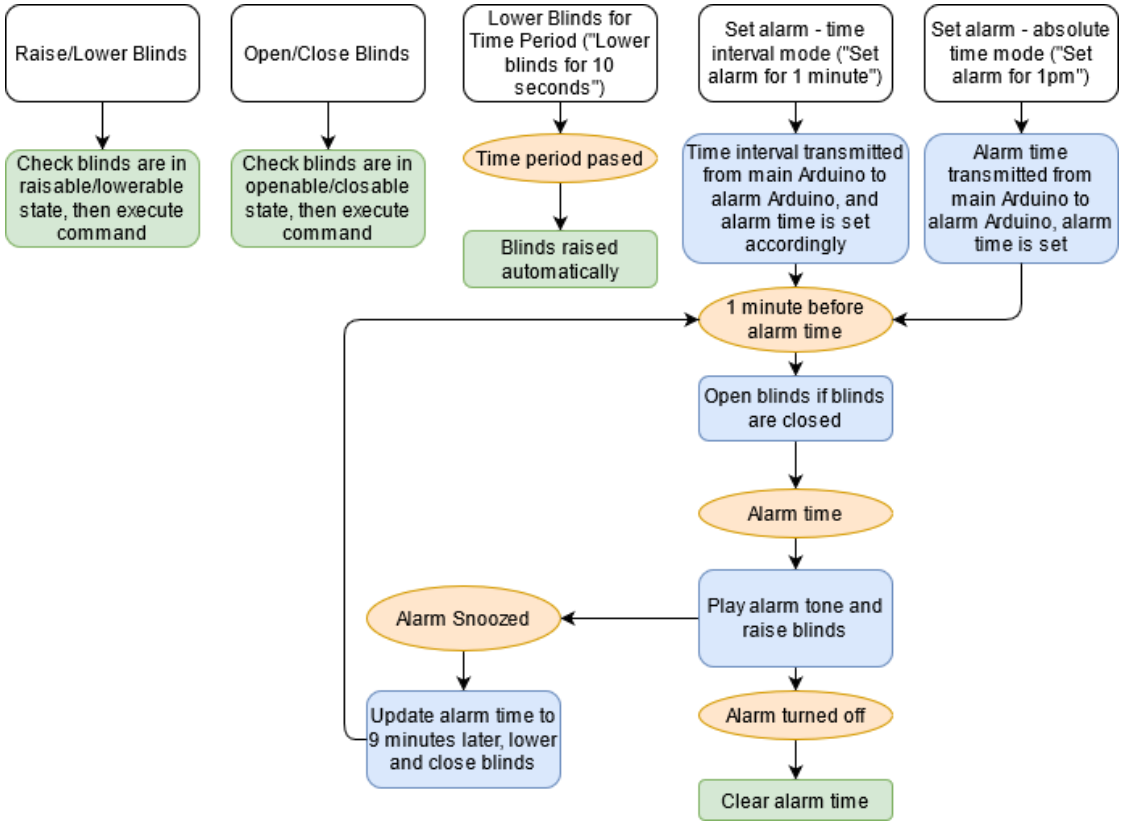
Originally, the design incorporated the winch mechanism used to raise and lower the slats and the mechanism used to rotate the slats into a single box at the top of the shades. The winch mechanism would be controlled by a single stepper motor and the tilt rod used to control rotation controlled by a servo. However, the tilt rod ended up requiring relatively high amounts of torque to rotate directly. Rotating the rod used to the control the tilt rod required much less torque due to gearing inside the existing shades. Therefore, it was much more natural and more easily accessible to build and design a stepper motor mount that rotates the rod using a 3D printed shaft coupler.

It was also much easier to design a stepper motor mount for the winch mechanism free from the size constraint of having to design one to fit inside a box at the top of the shades. The crossbeam used for the frame of the shades provided a perfect spot to mount the winch and stepper motor.

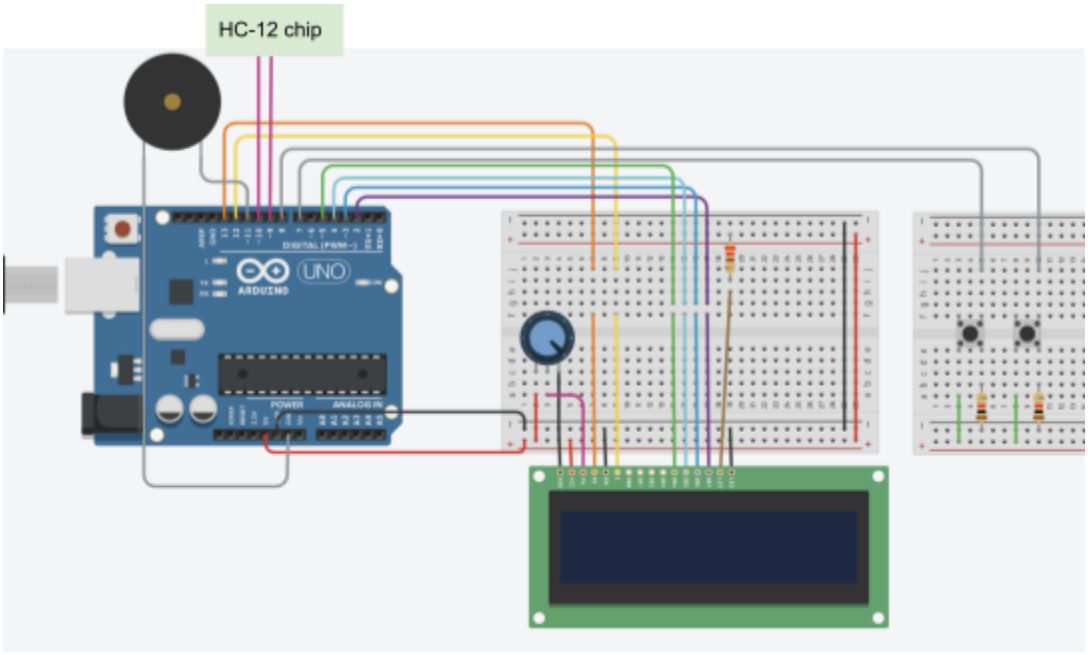
Our original design included the ability to layer in different functionality modes for the smart shades: a photosensor could be used to open the shades in the morning and close them at night and a motion sensor could be used to open the shades when they detected that a person had entered the room. However, due to the limited number of pins on the Arduino available and the number of pins required to control the stepper motors and the HC-12 chip, only the minimum functionality was implemented. We also offloaded the alarm clock functionality to a second Arduino; we discovered through user research that people prefer to have their alarm clock in a different location than beside their windows, often near their bedside.

4. Schematic-level design of your robot showing all components and how they are Connected.

Functionality Flow Chart



The chart above shows all implemented functionality on Helios.



Above: Alarm Clock circuit

BitVoicer - Arduino communication

The table below shows all binary data sent from the BitVoicer Server program to the Arduino. We always send a sequence of 4 bytes, and the first byte is used to indicate the function to be executed. The remaining bytes are used to send relevant data for the specific function as necessary.Note that the raw data being sent is in hexadecimal format, but we have written the bytes below as decimal integers for clarity.

Voice command	Byte 1	Byte 2	Byte 3	Byte 4
Raise blinds	1	0	0	0
Lower blinds	2	0	0	0
Lower blinds for [time]	3	[time in seconds]		
Set alarm for [time interval] (e.g. set alarm for 1 minute)	4	[hours]	[minutes]	[seconds]
Set alarm for [absolute time] (e.g. set alarm for 1pm)	5	[hours]	[minutes]	0
Close blinds	6	0	0	0
Open blinds	7	0	0	0

HC-12 communication between Arduinos
From the main Arduino, we forward all commands detailed above sent by BitVoicer server through the HC-12 chip to the alarm Arduino. The alarm Arduino parses the first byte for either 5 or 6, and sets the alarm according to the time sent in the next 3 bytes.

The alarm Arduino also sends data back to the main Arduino, mainly to tell it to raise/lower the blinds when an alarm is ringing or snoozed. Only 1 byte is sent to indicate the function to be executed. The data format is shown in the table below.

Function to execute	Byte 1
Raise blinds (sent when alarm starts ringing)	1
Lower blinds (sent when alarm is snoozed)	2
Close blinds (sent when alarm is snoozed)	6
Open blinds (sent 1 minute before alarm starts ringing)	7

5. Mechanical and electrical calculations for your design. For instance, as appropriate, include calculations for torque, total current draw, selection of sizes for resistors, capacitors, gain coefficients, etc.

From the data sheet of the 28-BYJ-48 5V stepper motor, as well as various guides, it is estimated that 1 stepper motor draws a maximum current of 240mA when running. This means that 2 stepper motors should use 0.48A when running. Also, it should be noted that the Arduino 5V pin (with the Arduino plugged in to a wall outlet) can supply up to a maximum of 0.8A. As the current draw of the HC-12 chip is negligible, we estimate that we should be able to power at least 2 stepper motors at maximum power without issues.

Once we set up the winch to be controlled by 2 stepper motors, we measured the current draw with a multimeter. The current draw when the steppers are raising the blinds was measured to be approximately 0.74A, which is slightly higher than the predicted current draw. This direspancy might be caused by the fact that other components, like the motor driver board, are also drawing current. Since the maximum current draw is under 0.8A, our system has enough current to run smoothly.

6. As always, a well-commented sketch should be uploaded to Canvas.

Please refer to the attached .ino files, as well as the tone.cpp file for the modified default Arduino tone library. We did not include the code files in the appendix as there are multiple long code files and that will make the length of the report unnecessarily long.

8. (optional) Suggestions for improvements for this assignment (for future years).

More guidance on how to integrate components together will be extremely helpful. For instance, we struggled quite a lot in figuring out how to connect the stepper motors to the winch and the tilt rod, and if we had knowledge of what is the best way to connect motors to components that we want moved, the integration process would have been smoother.