December 17, 2018

Smart Window Blinds

Aya Mistica & Rui Chen

Design Proposal

Aya Mistica & Rui Chen
1000 E University Ave
Laramie, WY 82071
amistic1@uwyo.edu | rchen2@uwyo.edu

December 6, 2018

Dr. Steven F. Barrett, Professor
University of Wyoming
College of Engineering and Applied Science
1000 E University Ave (EN 2085B)
Laramie, WY 82071

Dear Dr. Barrett:

We, Aya Mistica and Rui Chen, both 4th year computer engineering students, are pleased to commence our senior design project for EE 4820. We would like to propose a project to create and design smart window blinds. This project features both software and hardware implementations, but we are excited to take on a project which will challenge our technical skills and knowledge to further mold us into competent and knowledgeable computer engineers. This cover letter will provide an overview of our proposed idea.

According to a survey conducted by Pew Research Center in January 2018, over 77% of American adults own or use a smartphone [1]. Due to the growth in smartphone users, wireless technologies have become more and more integrated into everyday objects and given rise to the Internet of Things (IoT). By 2025, it is estimated that there will be more than 55 billion IoT devices [2]. For these reasons, we propose to create and design wirelessly-controlled window blinds which will allow users to install the mechanism on pre-existing window blinds and control the blinds from their smartphone. Through the mobile app, users should be able to open, close, raise and lower the window blinds. This project aims to design an independent system allowing users to fully utilize their household blinds with just a tap on their screen.

We feel that this project will not only challenge us both technically and creatively, but will help us gain knowledge and understanding within the growing IoT field. Feel free to contact us with feedback and concerns through our school emails: amistic1@uwyo.edu or rchen2@uwyo.edu. We sincerely appreciate your time.

Kind regards,

Aya Mistica and Rui Chen

The Smart Window Blinds project aims to create a self-contained system which will allow users to fully automate window blinds through their smartphone. The system consists of two main components: an attachable module that provides motorized control to open, close, raise and lower the window blinds and a smartphone app which will connect to the attachable module over Wifi. For design and implementation purposes, the project is divided into three technical aspects: software, hardware, and mechanical.

The objective of the software portion of the project is to create a mobile application through Evothings Studio which will allow the user to communicate to our chosen microcontroller, the ESP8266. Another goal for the smartphone app is to implement a straightforward, clean interface for an easy-to-use experience. We would also like to add security measures between the IoT device and mobile app to protect the privacy and data of our users.

On the hardware side, we will be using the ESP 8266 as the main “microcontroller”. Using a standalone ESP 8266 will still give us full Wifi connectivity, efficient reception of data from the mobile application as well as the capability of sending pulse width modulation signals to the DC motor. The hardware will also include a 9V power supply, voltage regulators, resistors, capacitors, wires and other circuitry components. Upon receiving input from the mobile app, the ESP 8266 should output a corresponding PWM signal to the DC motor which will physically move the window blinds. The PWM signal will be determined by the type of input data received.

The mechanical system will be at the heart of physically moving the window blinds. It will include a DC motor, gears, and other mechanical parts to raise and lower the system. More research needs to be done on specific parts as well as how the mechanical system should move

the blinds. Also, the system should be able to receive the PWM signal and move the window blinds seamlessly, swiftly and efficiently.

Functional Block Diagram

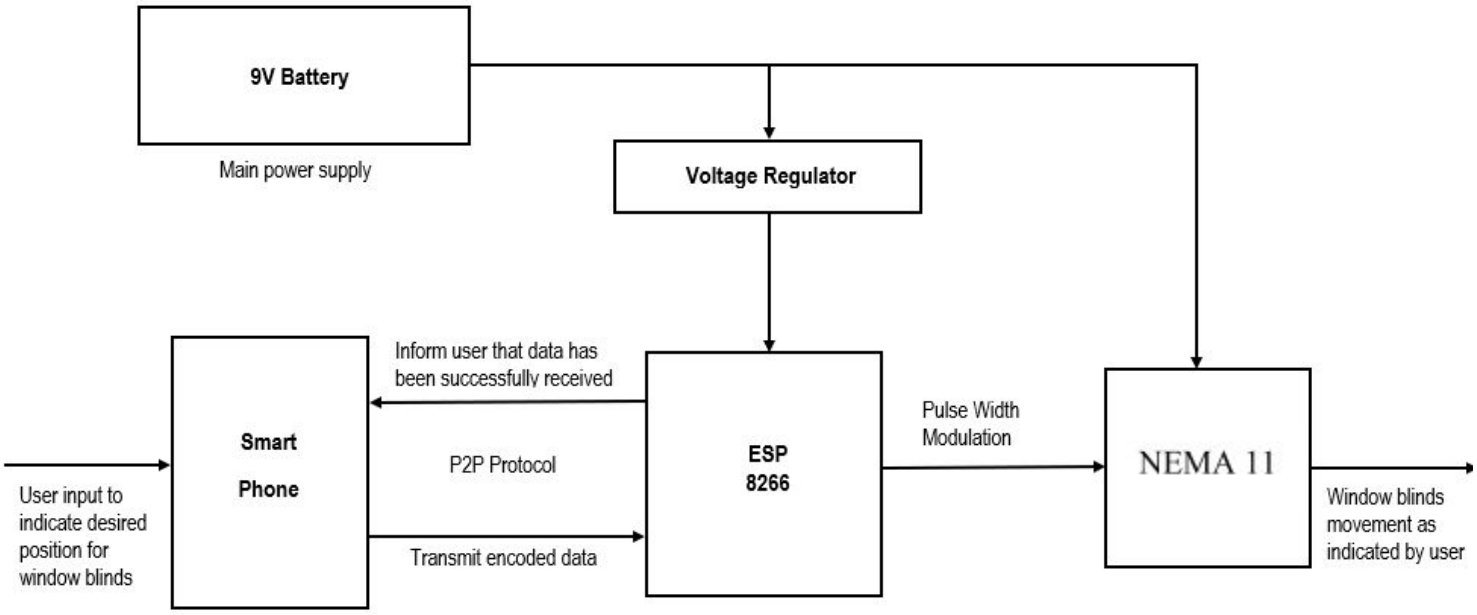


Figure 1. Smart Window Blinds

The block diagram in Figure 1 shows all the modules in the Smart Window Blinds project. Each module will be described in detail based on specific inputs, outputs and functionalities.

Table 1: Smartphone Module

Module	Smartphone
Inputs	User interacts with mobile application to indicate desired position of window blinds. When ESP8266 receives encoded data, send success message back to mobile application to inform user.
Outputs	Transmit encoded data: using P2P Wifi Protocol, send user data to ESP 8266
Functionality	Exchange data wirelessly between Wifi module and user

Table 2: ESP 8266 Module

Module	ESP 8266
Inputs	Transmit encoded data: Using Wifi, decode received user input from smartphone
Outputs	When ESP8266 receives encoded data, send success message back to application to inform user. Send a pulse width modulation to DC motor determined by the decoded user data.
Functionality	Receive and decode user data to transmit appropriate pulse width modulation to DC motor.

Table 3: DC Motor

Module	NEMA 11
Inputs	Pulse width modulation from controller: moves the motor a specific angle based on the duty cycle of the signal. Power supply: 5V from L7805 voltage regulator
Outputs	Moves the window blinds as the user has requested
Functionality	Physically moves the window blinds based on user input

Table 4: Voltage Regulator

Module	Voltage Regulator
Inputs	Receives 9V from battery
Outputs	3V-3.6V for the ESP 8266 Supply appropriate voltage for DC motor
Functionality	Supplies voltage to components in the system within safe operating conditions.

Current and Alternative Solutions

One of the problems we may face is unsuccessful communication between the ESP 8266 and the mobile application on the smartphone. Luckily, the ESP 8266 has extensive documentation on setup, testing, and recommended usage. There are many suggestions and proven tips by those who have done similar projects. We can find multitude of solutions once we are in the debugging and testing stages of the project. Replacing the ESP 8266 is not an option for us because then we would have to re-code and learn how to use a different wifi module.

Another problem we have is with the mechanical system that physically moves the window blinds. This is probably the biggest obstacle for us since we are computer and electrical engineering students. We have no experience with building mechanical systems and how to make them work. We will have to find help within the mechanical engineering department to gain perspective and insight on how to begin constructing the mechanical aspect of the project. We also need to do more research on DC motors, torque, and other mechanical engineering concepts.

Another possible issue is with the software development of the mobile application. We may find that our chosen software development platform is not what we need or incompatible with what we're doing. There may be additional functionalities that we want that the integrated development environment (IDE) lacks. An easy solution for this would be to research other IDEs and finding one that is more fit for the project.

Table 5: Parts List

Item	Quantity	Cost	Total
ESP 8266-01	3	$6.95	$20.85
USB to TTL Adapter (CP2102)	2	$9.00	$18.00
9V Battery	1 (2-pack)	$7.80	$7.80
Voltage Regulator	4	$0 ECE shop	$0
Miscellaneous Circuit Components (resistors, capacitors, wires, breadboard, etc.)		$0 ECE shop	$0
L293D Motor Driver	4	$3.00	$12.00
Evothings Studio Software Development	1	Free	$0
DC Motor	5	~$7	$35

3-D Printed Casing	1	~$10	$10
Vertical Blinds Set	1	~$50	
			$153.65

Table 5 shows the parts, quantity, and approximate cost for the Smart Window Blinds project. Development costs are much higher due to ordering extra parts just in case. We also did not include a smart phone since this is already privately owned by both members and adds no additional expense.

Final Cost Analysis and Project Funding

The total cost for the Smart Window Blinds project is $73.65 which is well under-budget for the provided $250 funding by the ECE department. No outside funding sources will be needed. This cost estimate would be much lower if only the exact required quantities for each part are ordered. One of the goals for the project is to design the system in the most reliable functionality but as cost-effective as possible. With this cost estimate, the Smart Window Blinds can be priced competitively in the IoT market.

Design/Construction

Software

To develop the software application, we have decided to use the platform, Evothings Studio. With Evothings, we can easily create the front-end of the application using various web technologies like HTML, CSS and Javascript. HTML will be used to create a structured layout on the application; Javascript will handle all the functionality; and CSS will make the buttons

and layout of the page appealing. All we need to do is download Evothings Workbench to our laptop and the Evothings Viewer on our mobile devices.

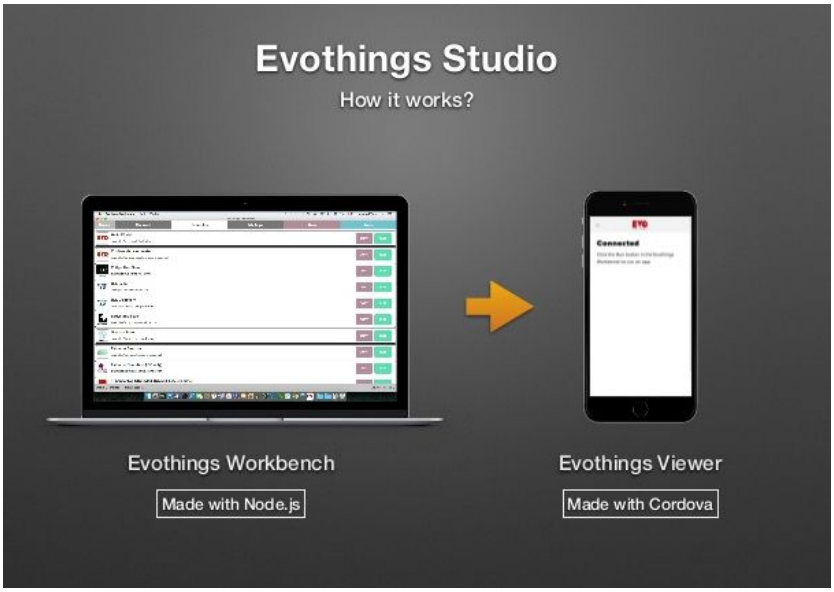
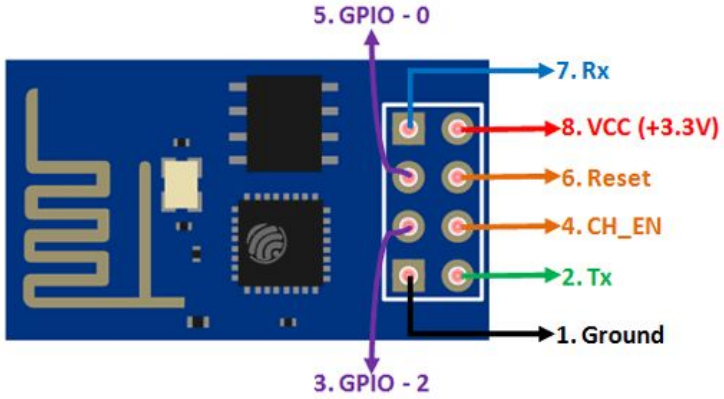


Figure 2. Evothings Workbench and Evothings Viewer.

Evothings Viewer is available for IOS and Android phones so we don't need to create two different codebases tailored to a specific operating system. The way it works is if a laptop and mobile device is connected to the same Wifi network, the laptop and mobile device can be connected via a specific "Connect Key". Once connected, any new saved changes in the code from the Evothings Workbench will auto-reload the Evothings Viewer on the mobile device and reflect the changes in real-time as shown in Figure 2. Multiple devices will be able to run and display our specific application as well. Currently, we plan on only supporting one range of motion for our vertical blinds so we will only need to create an "Open/Close" button for the app. Clicking this button on the app should open (if the blinds are closed) and close (if blinds are open), the window blinds upon the user's request. To test the connection between the mobile application and the ESP 8266, we can first use an LED connected to the GPIO2 pin on the

ESP8266, and program the ESP8266 to send a signal to the pin whenever a connection has been established with a mobile device. We can then move on from here to connect the output pins (shown on Figure 3) of the ESP 8266 to the motor driver.



Pin Number	Pin Name	Alternate Name	Normally used for	Alternate purpose
1	Ground	-	Connected to the ground of the circuit	-
2	TX	GPIO – 1	Connected to Rx pin of programmer/uC to upload program	Can act as a General purpose Input/output pin when not used as TX
3	GPIO-2	-	General purpose Input/output pin	-
4	CH_EN	-	Chip Enable – Active high	-
5	GPIO - 0	Flash	General purpose Input/output pin	Takes module into serial programming when held low during start up
6	Reset	-	Resets the module	-
7	RX	GPIO - 3	General purpose Input/output pin	Can act as a General purpose Input/output pin when not used as RX
8	Vcc	-	Connect to +3.3V only	

Figure 3. ESP8266 Pin Configurations.

Hardware

Once we obtain the ESP 8266 and FTDI Programmer, the first step is to program the chip. We will program the ESP 8266 with NodeMCU which uses LUA script. Programming with LUA using NodeMCU firmware is very similar to programming an Arduino. In addition, we need to use ESPlorer program to create and upload LUA files into the ESP 8266. The first goal is to establish communication between the Evothings application and ESP8266. We can confirm this communication by connecting an LED to one of the pinouts of the ESP 8266, and when the button is pressed on the Evothings mobile application, the LED should should light up. After this, we can connect the ESP 8266 to send PWM signals to the motor driver which will drive the DC motor. We can test the PWM signals first by connecting one of the pins to an oscilloscope and when a button is pressed on the mobile application, we should be able to detect the PWM signal on the oscilloscope.

Mechanical

Our mechanical design involves using the bipolar mini stepper motor, NEMA11. Its torque is 6 N-cm and it takes 200 steps/rev. Its voltage and current rating are 3.8 V and 0.67 A, respectively. We chose this stepper motor on the premise of examining the motion involved in opening and closing a vertical window blind. A manual window blind has a rotating wand which rotates a 1.8cm diameter knob at the top of the blind. To open and close, it takes 5 full turns of this knob. Since we did not have a tool to measure the torque required to turn the knob, we are estimating that a stepper motor with 6 N-cm torque is enough for our gear system. Connected to the stepper motor is a gear system which will be attached to a metal rod. This metal rod should be inserted in

the hole currently occupied by the rotating wand. With the gear system and rod, the motor should be able to rotate the knob automatically. Right now we are not so much concerned about the speed at which the window blind should open and close, but we need to make sure that the stepper motor is capable of moving the gear system which will move the blinds. To be safe, we may order a stepper motor which has a slightly greater torque, just in case 6 N-cm is not enough. Once we are able to get it to work, we can worry about increasing the voltage delivered to the motor to possibly increase its speed. Another concern is that a higher torque may require more power from our 9V power supply, and we need to be careful with keeping the system low-powered to prolong our battery life. Our motor driver for the stepper motor is the DRV8833C. We will connect this to 3 I/O pins of the ESP 8266. One of the I/O pins will control the sleep mode of the motor driver. The other two I/O pins are connected to output the PWM signal to the motor driver. The pin that transmits the PWM signal will determine the direction which the stepper motor will turn.

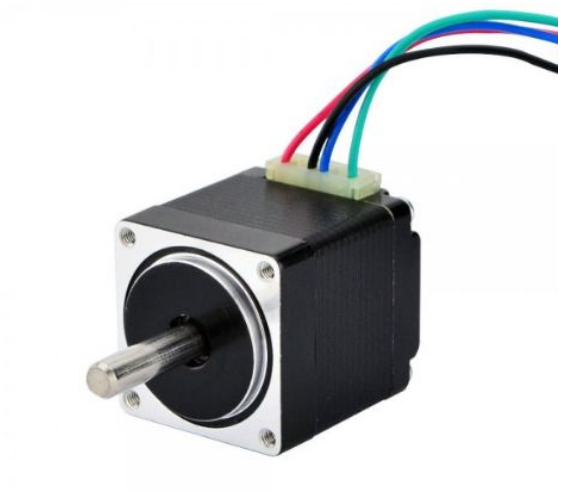
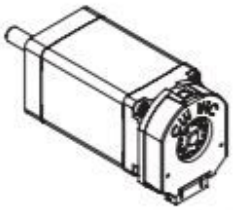
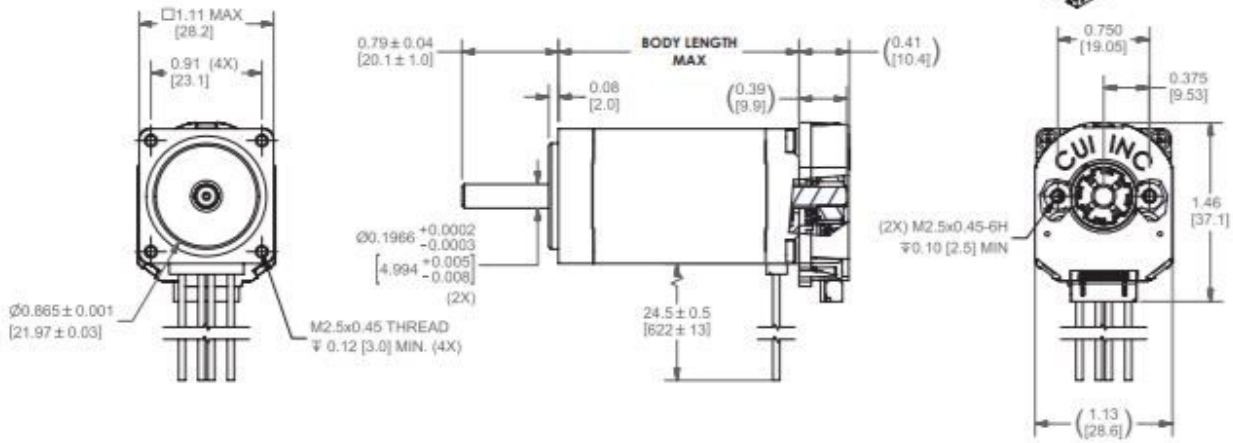


Figure 4. NEMA 11 Stepper Motor

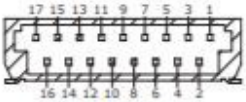
MECHANICAL DRAWING

units: inch [mm]
tolerance:
X.XX ±0.01 [±0.25]
X.XXX ±0.005 [±0.13]
X.XXXX ±0.0005 [±0.013]





MOTOR WIRE CONNECTIONS	
Color	Function
red	A
blue	$\overline{A}$
green	B
black	$\overline{B}$
26 AWG, PVC	



Mating Connector:
JAE FI-W17S

ENCODER CONNECTIONS	
#	Function
1	TX_ENC+
2	RX_ENC+
3	N/A
4	GND
5	N/A
6	+5 V
7	N/A
8	B+
9	N/A
10	A+
11	N/A
12	Z+
13	N/A
14	MCLRB
15	N/A
16	N/A
17	N/A

MODEL NO.	BODY LENGTH (inch)	WEIGHT (lb)
NEMA11-13-01D-AMT112S	1.26	0.24
NEMA11-18-01D-AMT112S	1.77	0.35
NEMA11-18-02D-AMT112S	1.77	0.35
NEMA11-20-01D-AMT112S	2.01	0.45
NEMA11-20-02D-AMT112S	2.01	0.45

Figure 5. NEMA 11 Motor Data Sheet

Project Timeline With Specific Deadlines

TASK	START	END	DAYS
Software			
Create Webpage Application	1/7/19	1/16/19	10
Test Application	1/16/19	1/21/19	6
Connect Application to ESP	2/16/19	2/28/19	13
Test Application with ESP	3/1/19	3/15/19	15
Hardware			
Built ESP circuit	1/21/19	1/27/19	7
Program ESP	1/28/19	2/3/19	
Test ESP	2/4/19	2/15/19	12
Connect ESP with Mobile Application	2/16/19	2/28/19	13
Machanical			
Order Parts	1/14/19	1/27/19	14
Produce a design with motor	3/1/19	3/10/19	10
Develop System for motion and Test	3/11/19	3/31/19	21
Connect motor with ESP and Test with Application	4/1/19	4/30/19	30
3D printing and casing project	4/22/19	4/30/19	

Schematic Diagram

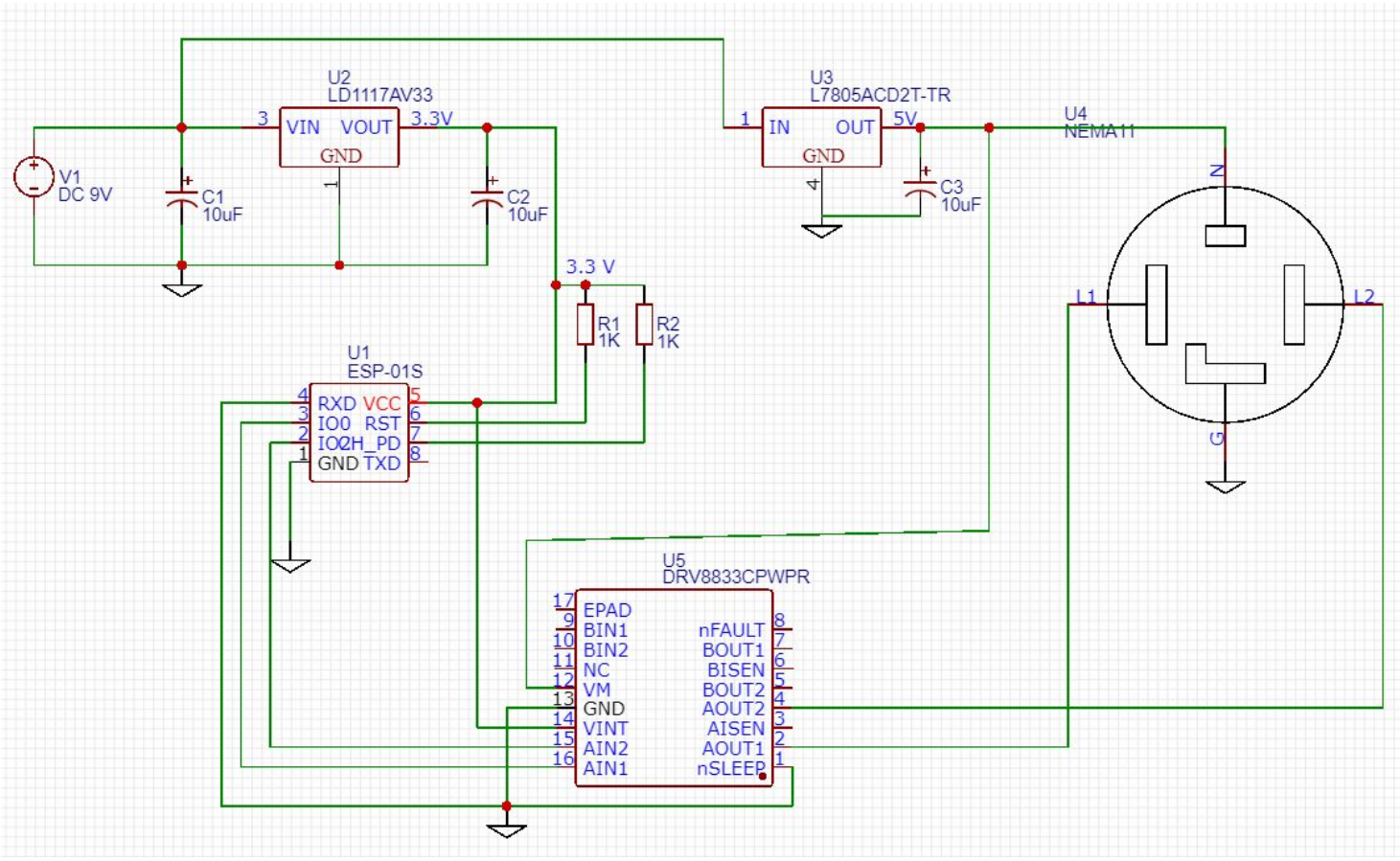


Figure 6. Smart Window Blinds Schematic Diagram.

For our schematic, we are using a 9V DC power supply. 9V is fed into LD1117 and steps down to 3.3 V to power our ESP 8266. It is also an input to L7805A which will output 5V for our stepper motor and motor driver. Our motor driver is the DRV8833C and our stepper motor is the NEMA 11. L1 and L2 are connected to AOUT1 and AOUT2 of our motor driver which will control when the stepper motor moves and in what direction. AIN1 and AIN2 will receive the PWM signal from the IO0 and IO2 pins of the ESP 8266 and this will control the direction in which the stepper motor moves. RXD of the ESP 8266 can be used as a general pin when not in programming mode, and this will be connected to nSleep of the motor driver. nSleep will put the

motor driver in sleep mode when it's set to 0, and enables the motor driver when set to 1. We have the 10uF capacitors at the input and output of the voltage regulators to keep the circuit stable. In addition, we have RST (Reset) and CH_PD (chip enable) set to HIGH to make sure the ESP 8266 does not reset when it’s not suppose to as well as to enable to the Wifi module.