Remote Light Switch Controller

Personal Project



**Andy Zheng**

September 2020 – January 2021

Table of Contents

[**I.** **Introduction** 3](#_Toc62151404)

[**II.** **Planning** 3](#_Toc62151405)

[A. Establishing Product Requirements 3](#_Toc62151406)

[B. Choosing the Processor 3](#_Toc62151407)

[C. Choosing Control Options 4](#_Toc62151408)

[1) Push Buttons 5](#_Toc62151409)

[2) IR Remote 5](#_Toc62151410)

[3) Wi-Fi 5](#_Toc62151411)

[D. Choosing a Power Source 5](#_Toc62151412)

[E. Actuating Mechanism 5](#_Toc62151413)

[F. Parts List 6](#_Toc62151414)

[**III.** **Testing** 6](#_Toc62151415)

[A. Component Testing 6](#_Toc62151416)

[1) Batteries 6](#_Toc62151417)

[2) Servo Motor 6](#_Toc62151418)

[3) IR Receiver 6](#_Toc62151419)

[4) Push Buttons 6](#_Toc62151420)

[5) Servo Arm and Base 6](#_Toc62151421)

[6) ESP-12E 7](#_Toc62151422)

[B. Code Testing 7](#_Toc62151423)

[1) IR Control 7](#_Toc62151424)

[2) Button Control 7](#_Toc62151425)

[3) Wi-Fi Control 7](#_Toc62151426)

[C. Functional Test 7](#_Toc62151427)

[1) Building on a Breadboard 7](#_Toc62151428)

[2) Improving Wi-Fi Connectivity 8](#_Toc62151429)

[3) Fixing the Servo Motor Noise 8](#_Toc62151430)

[4) Reducing Power Consumption 8](#_Toc62151431)

[**IV.** **Building** 8](#_Toc62151432)

[A. Power Delivery Interface 9](#_Toc62151433)

[B. Designing the PCB 9](#_Toc62151434)

[C. Testing the PCB 9](#_Toc62151435)

[D. Soldering the Components 9](#_Toc62151436)

[E. Finishing the System 10](#_Toc62151437)

[**V.** **Reflection** 10](#_Toc62151438)

[A. Evaluation of Results 10](#_Toc62151439)

[B. What I Enjoyed 10](#_Toc62151440)

[C. What I Found Difficult 10](#_Toc62151441)

[D. Possible Improvements 11](#_Toc62151442)

[1) Power Supply 11](#_Toc62151443)

[2) Internet 11](#_Toc62151444)

[E. Lessons Learned 11](#_Toc62151445)

[**VI.** **Additional Information** 11](#_Toc62151446)

[A. Video Demonstration 11](#_Toc62151447)

[B. Project GitHub 11](#_Toc62151448)

List of Figures

Fig. 1. ATtiny85 Microprocessor 4

Fig. 2. ATtiny85 Pinout 4

Fig. 3. ESP-12E Microprocessor 4

Fig. 4. Adafruit HUZZAH ESP8266 Breakout Board 4

Fig. 5. Circuit to Program Standalone ESP-12E 5

Fig. 6. Circuit for Normal ESP-12E Operation 5

Fig. 7. Omron 3BF Tactile Switch 5

Fig. 8. Keyes IR Remote 5

Fig. 9. TSOP38238 IR Receiver 5

Fig. 10. Case for Three AA Batteries 6

Fig. 11. 3D Printed Light Servo Mount 6

Fig. 12. Adafruit 3.1kg/cm Torque Servo Motor 6

Fig. 13. Code for IR Remote Control 7

Fig. 14. Code for Button Control 7

Fig. 15. Code for Wi-Fi Control 8

Fig. 16. Wi-Fi Control User Interface 8

Fig. 17. Fritzing Sketch of Functional Test 8

Fig. 18. EasyEDA PCB Citcuit 9

Fig. 19A. Custom PCB Front (In Mounting Orientation) 9

Fig. 19B. Custom PCB Back 9

Fig. 20. Components Soldered Onto PCB 10

Fig. 21. Final Product 10

Fig. 22. System Mounted on Light Switch 10

List of Tables

Table 1. Parts List 6

Remote Light Switch Controller

Andy Zheng

***Abstract*—This report documents my process for making the remote light switch controller from concept to final product. The completed device uses an ESP8266 Wi-Fi capable microcontroller to process user commands and control a servo motor arm that rotates to flip a light switch on or off. The device is mounted by using two screws that hold the faceplate for a standard toggle switch in place. Controls include physical push button to toggle the light switch up or down, an IR remote to control the switch remotely when it is in line of sight, and Wi-Fi that can be done from any device that is connected to the same network as the ESP.**

# **Introduction**

At the Rochester Institute of Technology, I have taken classes on electronics, circuits, Arduino microcontrollers, and many other engineering-related courses. I have also gained some engineering experience working as an intern at L3Harris where I helped test, troubleshoot, and fix complex circuit boards. I wanted to test the knowledge I have acquired throughout the years by starting a project. Ideally, it would also allow me to learn a new skill. I wanted to make something that is practical. I decided to make a remote light switch controller. Now that my parents are getting old, and this could eliminate the need for them to walk across the room to turn off the lights. Although the concept of a remote-controlled light switch flipper is not unique, the practicality of it makes it worth making. I decided to add some features to my design to make it more unique. The entire system must be compact and be easily portable. It must be compatible with as many toggle switches as possible. This means that the system must have the capability of being mounted upside down or right side up and operate correctly in the case that there is no room to install on one side of the light switch. Finally, the device must be as convenient to use as possible and have backup options for when a certain method of control is less convenient.

# **Planning**

This section outlines my decision-making process for selecting components and establishing design criteria for the final product.

## Establishing Product Requirements

The following are a list of design criteria for the remote light switch controller that will make it more unique as well as make it more useful.

* The device must be controllable via IR remote, Wi-Fi, and physical push buttons. The user might not be in line of sight of the device, so Wi-Fi control from a mobile device can serve as a secondary option. If Wi-Fi does not work, then the IR remote can be used instead. Push buttons that control the up and down movement can be used if the user is already close to the light switch and it would be easier to flip it manually than to have to grab the remote or their phone.
* The system must be mountable upside down as well as right side up in case the light switch is obstructed on one side.
* The processor must be easily removeable from the system to allow for reprogramming to add features, bug fixes, or Wi-Fi networks.
* The system must be battery powered with easily replaceable or rechargeable batteries.
* Components must be soldered onto a PCB.
* The system must be mounted using the two screws that hold the light switch faceplate in place.

## Choosing the Processor

In class, I was introduced the Arduino Uno and its ATmega328P microprocessor. I wanted to use something similar for my project since it was familiar. However, an entire microcontroller such as an Arduino would be too large and wasteful. Using a standalone microprocessor would be more efficient. I did some research a found a good candidate: the ATtiny85 (Fig. 1), which is 9.27mm x 6.35mm in size. Using a standalone microprocessor is also beneficial for its lower power consumption. This would be ideal, as the system would be battery powered. Programming would be easy, since the Arduino Uno can be used as the programmer with Arduino IDE.

As I did some planning around using ATtiny85, I realized that there were going to be some problems. For one, there would not be enough GPIO pins on the chip. The ATtiny85 has 6 GPIO pins (Fig. 2), whereas my design would require



Fig. 1. ATtiny85 Microprocessor

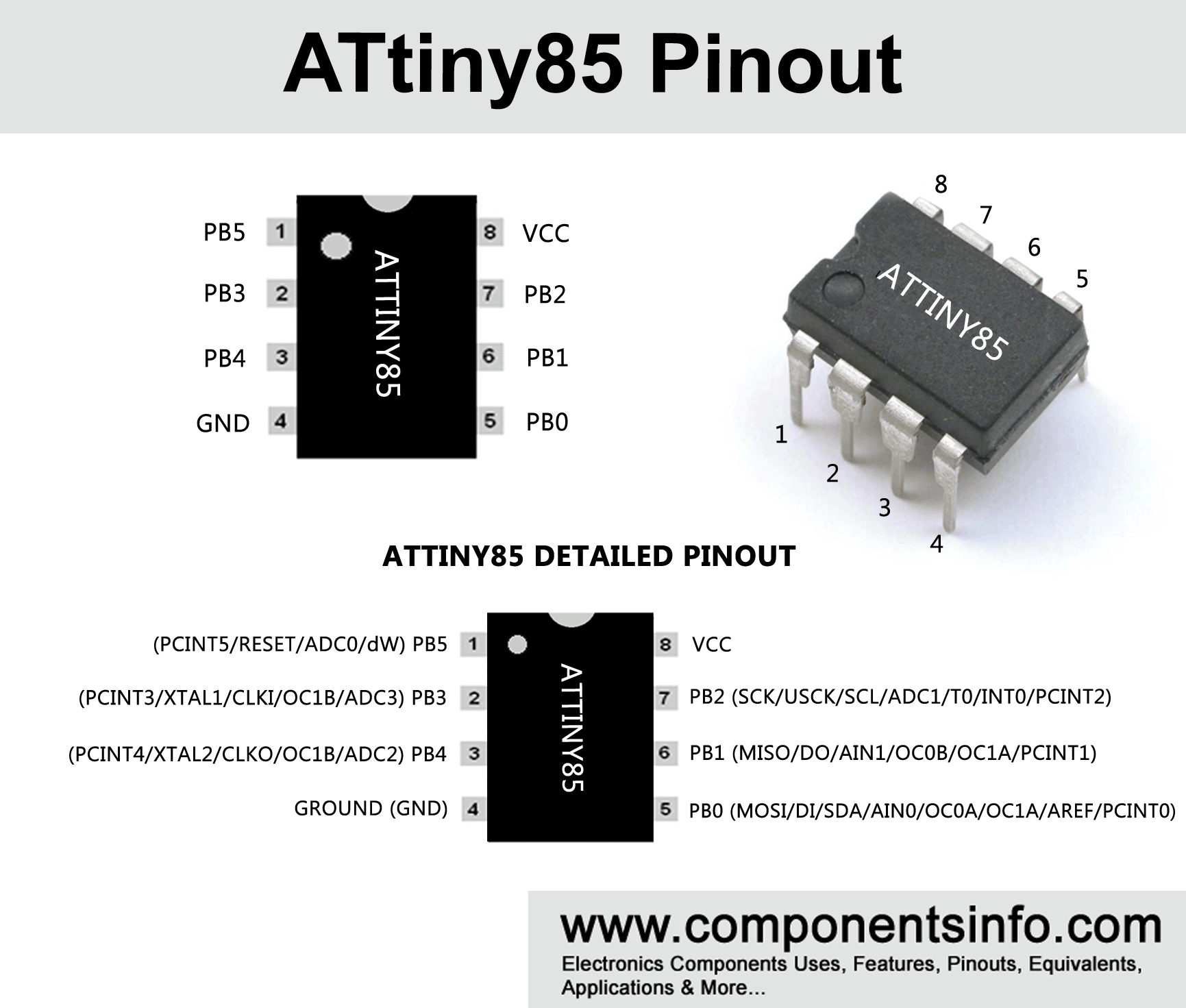


Fig. 2. ATtiny85 Pinout

7 for 1 slide switch, 2 push buttons, 1 IR receiver, 1 signal pin for the motor, and a Wi-Fi module that would communicate with the ATtiny85 using 2 pins. While searching for a suitable Wi-Fi module, I discovered the ESP8266 series of microprocessors. The variations of these processors came in different form factors, number of GPIO pins, and features. I found the ESP-12E (Fig. 3) has the most suitable form factor of 24mm x 16mm. I found a guide online that explains the ESP and then realized that it alone can function as my processor. However, these standalone chips have the drawbacks of being more difficult to program and do not come with breadboard pins. While searching for a solution, I found the HUZZAH ESP8266 breakout board from Adafruit (Fig. 4), which comes with a reset button, a button to put the ESP in programming mode, and pins to solder on. Despite being slightly on the larger side, it checked all my boxes, which included being easy to remove.

It took me some testing before realizing that the Adafruit ESP8266 breakout board would be my best option. I had tested an ATtiny85s and successfully programmed it before I realized that I wouldn’t have enough GPIO pins. I had also tested a standalone ESP-12E chip before I realized the problems that would entail. I was unable to permanent solutions for the pins it would require. The circuit configuration required to program the ESP (Fig. 5) is vastly different from the configuration for normal operation (Fig. 6). Most importantly, the ESP requires a 3.3V supply, and using 5V would damage it. Since the servo motor uses a 5V supply, I would have to use a 5V power source and drop it



Fig. 3. ESP-12E Microprocessor

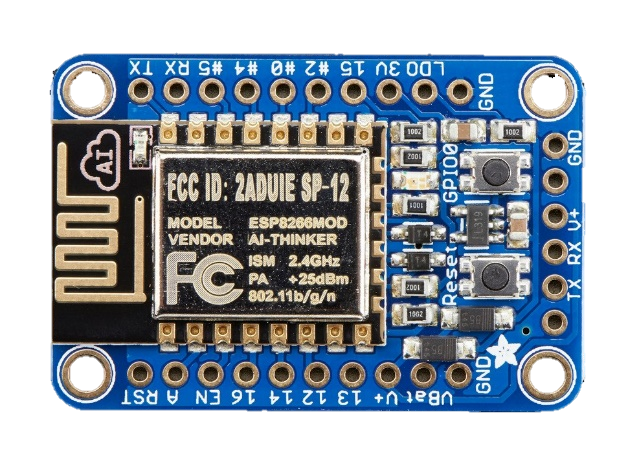


Fig. 4. Adafruit HUZZAH ESP8266 Breakout Board

down to 3.3V for the ESP. I tested two different methods for dropping the voltage: a voltage divider and a voltage regulator. However, I was not satisfied by either of these solutions. I felt like the circuit would become too complex and I would accidently send 5V to the wrong component and fry the ESP. I decided that the easiest way around all these problems would be to get an ESP8266 on a breakout board that has the programming/operating circuitry, with standard breadboard friendly pins, and a built-in voltage regulator. This was how I decided on the Adafuit HUZZAH ESP8266 breakout board.

## Choosing Control Options

When choosing ways to control the light switch mechanism, I wanted to avoid using only one method since there will be times when controlling remotely would be inconvenient. For example, when the user is near the light switch when they want to flip it but the remote is sitting on their nightstand across the room. I decided to add up and down push buttons so the user can still easily control the light while standing by the switch. Another one of my concerns was that depending on the layout of the room, the light switch may be out of line of sight from where the user would want to use the remote. To get around this, I could use Bluetooth or Wi-Fi so the user can control the switch from anywhere in the room. I chose Wi-Fi because it has the benefit of being usable anywhere in the house on any device connected to the same network. If the house does not have Wi-Fi, the remote can serve as a backup. If there is no line of sight for IR transmission, then Wi-Fi can serve as backup. Having these three options add versatility and convenience for the user.

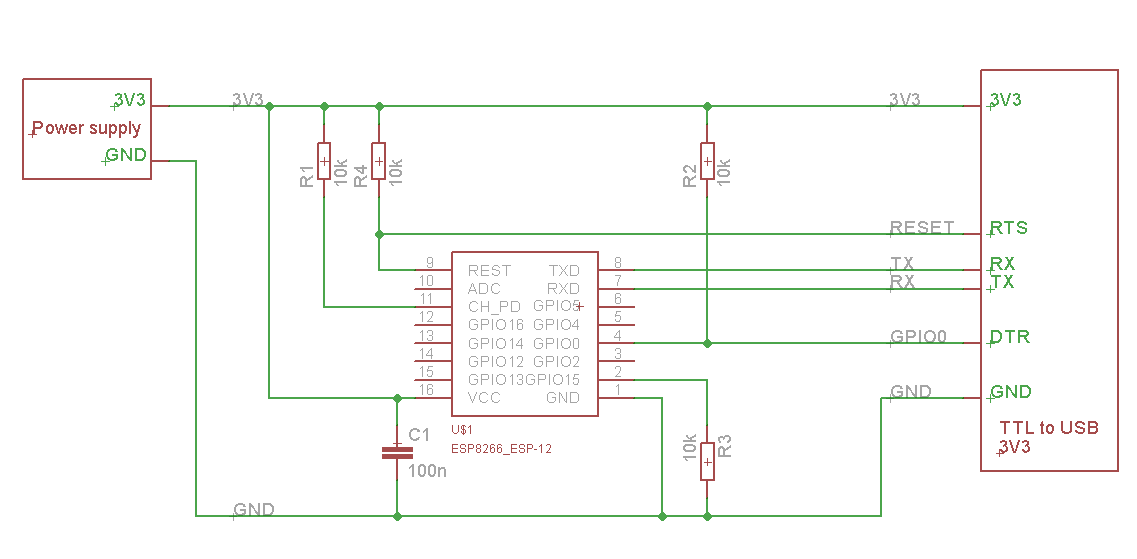


Fig. 5. Circuit to Program Standalone ESP-12E

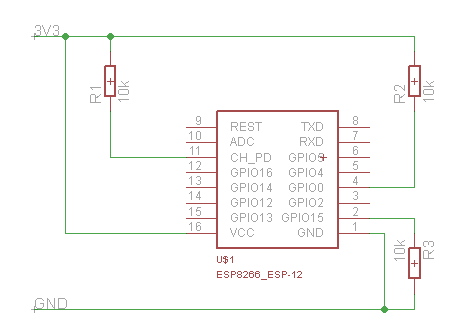


Fig. 6. Circuit for Normal ESP-12E Operation

### Push Buttons

For the up and down buttons, I chose 12mm x 12mm push buttons as shown in Fig. 7. I wanted the size to be large enough to be easy to see and push.

### IR Remote

For the IR remote, I chose the Keyes remote shown in Fig. 8. This was the remote I used in class. Only the up and down buttons will be used, and the remaining buttons will have no function. For the IR receiver, I chose the TSOP38238 IR receiver module (Fig. 9) for its advertised improved immunity against ambient light. It also has an operating voltage between 2.5V and 5.5V which allows it to be powered by the ESP-12E’s on board 3.3V output. Using a 3.3V supply for the IR receiver ensures that the output signal is at most 3.3V and will not damage the ESP.

### Wi-Fi

For Wi-Fi control, the options I considered were creating an Android application or creating a website with the ESP8266 as the server. I chose the website option as it can be used on any device with an internet connection and a browser.

## Choosing a Power Source

I wanted my system to be portable and compact, so using batteries would be the best option. While choosing which batteries to use, I had to keep my voltage, capacity, and size constraints in mind. Using a small lithium-ion polymer (LiPo) battery was a compelling option, but there are safety

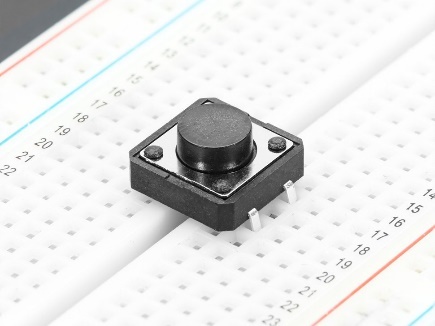


Fig. 7. Omron 3BF Tactile Switch



Fig. 8. Keyes IR Remote



Fig. 9. TSOP38238 IR Receiver

risks involved, especially when I wil be transporting the project often. It was also difficult to find a LiPo battery with the target voltage output. My next idea was using a 9 volt battery. It was the smallest battery that supplied at least the amount of voltage I needed. All I would need to do is step down the voltage using a regulator. However, a 9 volt battery has a very low capacity, of only 500 mAh. This would not be sufficient for something that would be continuously running. After some research, I found that using three AA batteries in series would be a good option. At 1.5V per battery, three in series would supply 4.5V, which is within operating range for a micro servo. The capacity of these batteries are 2400 mAh, which is significantly better. I would also not have to use an additional component to lower the voltage as I would with the 9 volt. I would need to use a holder for three AA batteries as shown in Fig. 10.

## Actuating Mechanism

The actuation mechanism is the motor that powers the arm’s sweeping motion that pushes the light switch up or down. The actuating system is made from two 3D printed parts, the arm and the base (Fig. 11). The design and 3D print files were created by a user on Thingiverse. The base is mounted by using the two screws on a light switch faceplate and then a servo motor is mounted on top. The sweeper arm is connected to the output shaft of the motor. I found that different light switches require different amounts of force to flip, so I had to find the motor with the maximum possible amount of torque that has the correct form factor and a 5V operating voltage. I tested several different servo motors and found that some are noisier than

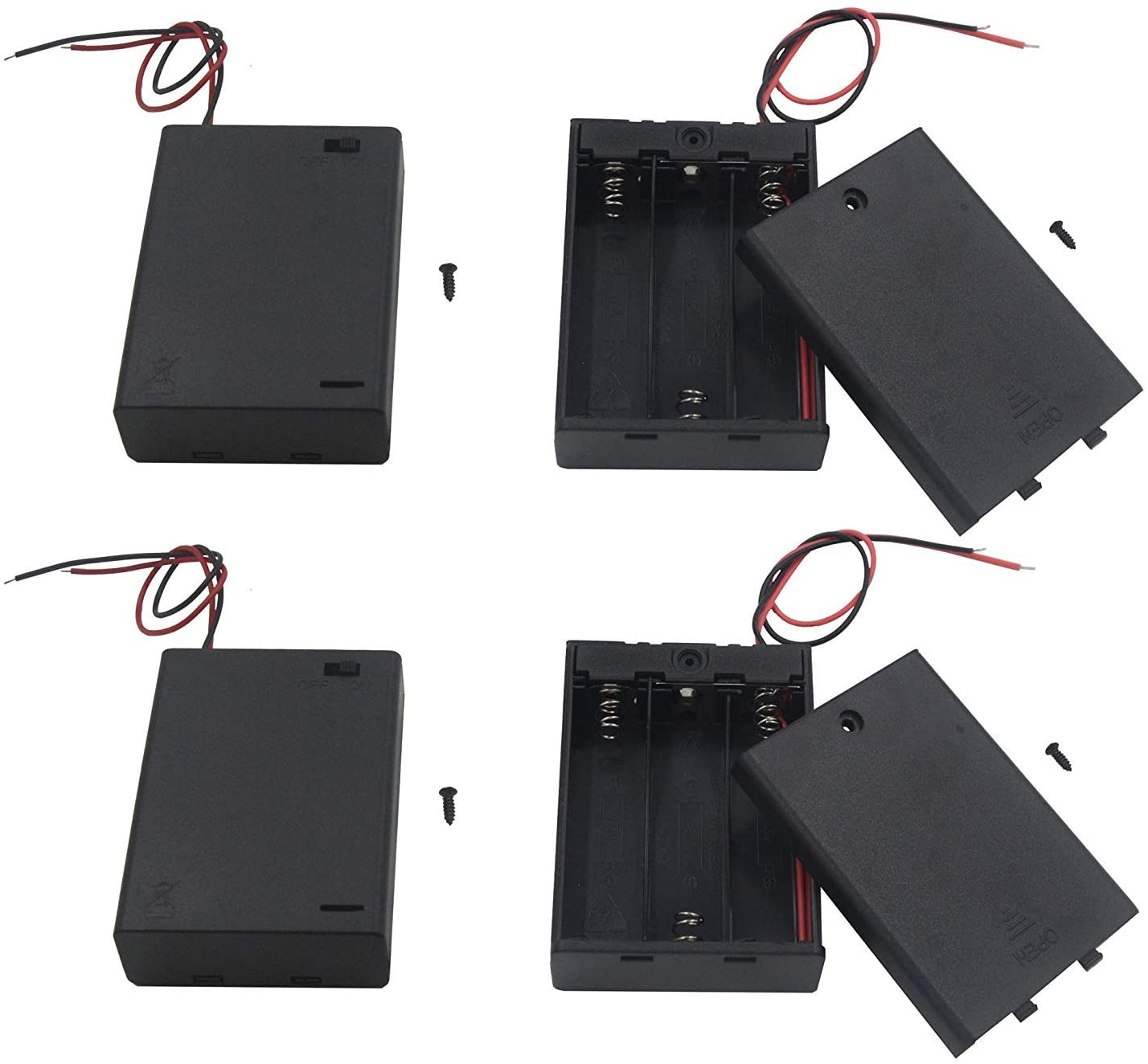


Fig. 10. Case for Three AA Batteries



Fig. 11. 3D Printed Light Servo Mount

others. I chose a micro servo motor from Adafruit with a torque of 3.1kg/cm, as shown in Fig. 12. It is also the most silent one I’ve tested.

## Parts List

The following table lists the parts I have used for the project.

Table 1. Parts List

|  |  |  |
| --- | --- | --- |
| **Part Function** | **Part Name** | **Image** |
| Processor | [Adafruit HUZZAH ESP8266 Breakout Board](https://www.adafruit.com/product/2471) | Fig. 4 |
| Up and Down Buttons | [12mm x 12mm Push Buttons](https://www.adafruit.com/product/1119) | Fig. 7 |
| Remote | Keyes IR Remote | Fig. 8 |
| IR Receiver | [TSOP38238 IR Receiver Module](https://www.amazon.com/gp/product/B08594ZSJQ/ref=ppx_yo_dt_b_search_asin_image?ie=UTF8&psc=1) | Fig. 9 |
| Power Source | Generic 3x AA Batteries | N/A |
| Power Source Interface | [3 AA Battery Holder with On/Off Switch](https://www.amazon.com/gp/product/B07C6XC3MP/ref=ppx_yo_dt_b_search_asin_image?ie=UTF8&psc=1) | Fig. 10 |
| Light Switch Actuating Mechanism | [3D Printed Servo Arm and Mounting Base](https://www.thingiverse.com/thing:1156995) | Fig. 11 |
| Motor | [Adafruit 3.1kg/cm Torque Micro Servo](https://www.adafruit.com/product/2307) | Fig. 12 |
| Orientation Switch | Generic Single Pole Double Throw Slide Switch | N/A |

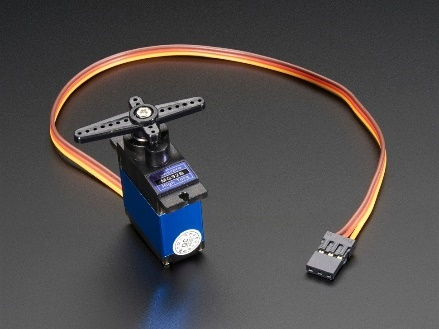


Fig. 12. Adafruit 3.1kg/cm Torque Servo Motor

# **Testing**

This section outlines my testing process to ensure the hardware and my code were working properly before finalizing the design.

## Component Testing

When I receive a component, I first perform a visual inspection and then test its functionality. Before adding any components to my system, I make sure that I fully understand its operation. This allows me to identify hardware defects early and makes troubleshooting unexpected problems more easily. The following describes my methods for testing my components.

### Batteries

I used a digital multimeter (DMM) to measure the voltages of each of the AA batteries to make sure they were within specification of 1.5V each. Then I measured the voltage of all three batteries in the battery case to verify that the total is 4.5V and that the battery case on/off switch works properly.

### Servo Motor

I tested the servo motor by connecting its VCC to the output of the battery pack and its signal to an Arduino pin. I wrote a simple sketch to sweep the motor arm. As a result of performing this test, I learned that the battery pack, the servo motor, and the Arduino must share a common ground for the motor to work. I verified that the servo motor can be powered by an external source.

### IR Receiver

I tested the IR receiver by using an Arduino and a sketch that prints out the code that is transmitted by the remote. I recorded the codes for the up and down buttons for later use.

### Push Buttons

I used a multimeter’s continuity test to verify that the switch was working properly. This also showed me what orientation the buttons will need to be in.

### Servo Arm and Base

I installed my servo motor into the 3D printed base with the servo’s included screws to make sure it will fit. I used the included servo arm to attach the 3D printed wiper arm,

|  |
| --- |
| **...**  #define ir\_up 0xFF629D  #define ir\_down 0xFFA857  **...**  **myservo.write(85);**  **if** **(**irrecv**.**decode**(&**results**))** **{**  **switch** **(**results**.**value**)**  **{**  **case** ir\_up**:**  myservo**.**write**(**120**);**  delay**(**500**);**  **break;**  **case** ir\_down**:**  myservo**.**write**(**45**);**  delay**(**500**);**  **break;**  **default:** myservo**.**write**(**85**);**  **}**  **}**  ... |

Fig. 13. Code for IR Remote Control

also using an included screw. Then, I wrote a sketch for my Arduino to rotate the arm up and down. Using the physical constraints of the base, I determined which angles would become the up, middle, and down positions.

### ESP-12E

While working with the ESP, I made sure to be cautious when supplying it with power. When using a USB to serial programmer that supplies 5V, I first verified the voltage using a multimeter before connecting it to the ESP. To test ESP, I uploaded a simple sketch that blinks the on-board LED by following a guide provided by Adafruit. Next, I tested the Wi-Fi capabilities of the ESP by using a provided sketch that connects it to my Wi-Fi network. This lets me know that it connected successfully. Finally, I tested the web server feature by using a sketch that sends a message to the website tied to my IP address. This confirms that the ESP and the website will send and receive data.

## Code Testing

When coding for a large project, I start by writing several small portions of code that each have a specific function and combine them after I have verified that each portion works. This section details my process for coding the project.

### IR Control

After acquiring the relevant IR codes to my remote, I wrote the code that controls the servo motor when the up or down buttons on the remote are pressed. The code snippet in Fig. 13 checks the received IR code. If the up button is pressed, the servo arm moves up, and similarly for down. When any other button is pressed, the motor does not move. Upon exiting the conditional statements, the motor is set to its default position.

|  |
| --- |
| #define button1 12  #define button2 13  void setup**()** **{**  **...**  pinMode**(**button1**,** INPUT\_PULLUP**);**  pinMode**(**button2**,** INPUT\_PULLUP**);**  **...**  **}**  void loop**()** **{**  myservo**.**write**(**85**);**    **if** **(**digitalRead**(**button1**)** **==** 0**)**  **{**  myservo**.**write**(**120**);**  delay**(**500**);**  **}**  **if** **(**digitalRead**(**button2**)** **==** 0**)**  **{**  myservo**.**write**(**45**);**  delay**(**500**);**  **}**  **}** |

Fig. 14. Code for Button Control

### Button Control

The condition for the servo arm to move up or down is a physical push of a button. In the code snippet from Fig. 14, the buttons connected to the GPIO pins on the ESP-12E are pulled high by default. When a button is pressed, the pin reads low, and the conditional statement instructs the servo arm to move.

### Wi-Fi Control

After confirming that the ESP-12E will connect to my network and can operate as a web server, I wrote the code that creates the website’s user interface that allows the user to send commands (Fig. 15). When “/up” is entered to the end of the root URL, the servo arm will rotate up. The opposite occurs when “/down” is entered. The code in Fig. 15 contains functions, handleUp and handleDown to handle those actions. Next, I wrote the HTML and CSS to make the website more user friendly. The result is shown in Fig. 16. When the up button is pressed, it performs the action equivalent to typing “/up” into the address bar. The HTML is written such that the website will scale to fit the aspect ratio of any device. On a smartphone, it will look like an app. This control method works anywhere if the controlling device is connected to the same Wi-Fi network as the ESP.

## Functional Test

The functional test involves making building the system on a solderless breadboard and making sure that everything works as expected before finalizing.

### Building on a Breadboard

I connected my components on a bread board as shown in Fig. 17. The slide switch would be used to identify which orientation the system would be mounted and adjust the

|  |
| --- |
| void setup**()**  **{**  **...**  server**.**on**(**"/"**,** handleRoot**);**  server**.**on**(**"/up"**,** handleUp**);**  server**.**on**(**"/down"**,** handleDown**);**  server**.**begin**();**  **}**  void loop**()**  **{**  server**.**handleClient**();**  myservo**.**write**(**85**);**  **}**  void handleRoot**()**  **{**  server**.**send\_P**(**200**,** "text/html"**,** index\_html**);**  **}**  void handleUp**()**  **{**  digitalWrite**(**pin\_led**,** LOW**);**  myservo**.**write**(**120**);**  server**.**send\_P**(**200**,** "text/html"**,** index\_html**);**  delay**(**500**);**  **}**  void handleDown**()**  **{**  digitalWrite**(**pin\_led**,** HIGH**);**  myservo**.**write**(**45**);**  server**.**send\_P**(**200**,** "text/html"**,** index\_html**);**  delay**(**500**);**  **}** |

Fig. 15. Code for Wi-Fi Control

wireless up and down directions accordingly. The slide switch should always be in the up position with respect to the user. Starting with the Wi-Fi control code I wrote before, I copied in the IR remote control code. I cleaned is up and made sure it worked correctly. Then I did the same with the push button control code. Finally, I implemented the feature where remote control directions for IR and Wi-Fi will reverse when the system is mounted upside down.

### Improving Wi-Fi Connectivity

I want my system to be operational even when there is no Wi-Fi. Other features will work, but Wi-Fi control will simply be unavailable. I had to rewrite the setup portion of the code to stop trying to connect to the specified network after a predetermined amount of time has elapsed. Now the ESP will attempt to connect to a network for 10 seconds after startup. If it is unsuccessful, Wi-Fi features will be unvailable but the other control options will still work. An on-board LED will blink to indicate that the ESP is attempting to connect to a network. Then the LED will

remain on to indicate it successfully connected or turn off to indicate a failure to connect.

### Fixing the Servo Motor Noise

During testing, I discovered that the servo motor will make a buzzing noise while it is checking its programmed angle to its current angle. This occurred every one to two

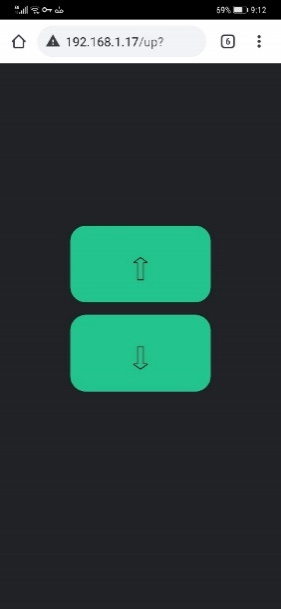


Fig. 16. Wi-Fi Control User Interface

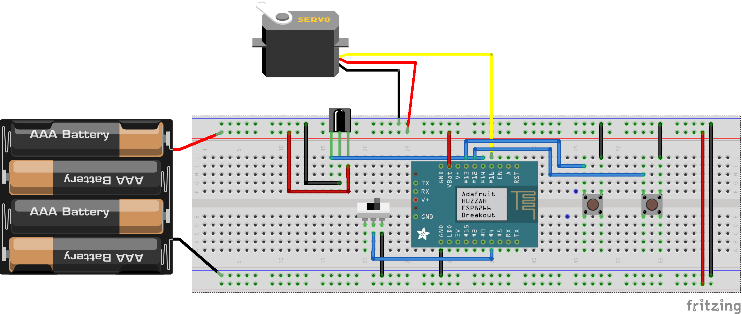


Fig. 17. Fritzing Sketch of Functional Test

seconds because the code to set the arm in its default position was always being run when idle. It would be too noisy to use in the household. I did some reasearch and found that the best way around this is to detach the servo in software when it does not need to move. Then I can reattach it immediately before it needs to move. By default, the motor will be detached.

### Reducing Power Consumption

While doing my testing, I also measured the amount of current the system was drawing. I found that after making the change to detach the motor when idle, the amount of current draw decreased significantly. While the motor is moving, the system drew ~250mA. When idle with the servo motor attached, the system drew ~150mA. With the servo detached, the current draw went down to ~78mA. Given that a typical AA battery has 2400mAh of capacity, the duration of continuous ~150mA draw would last approximately 16 hours. With a continuous draw of ~78mA, the duration would nearly double to 31 hours. However, neither of these times are acceptable for a system that must remain on at all times. This was the result of a lack of foresight during the planning stage. Possible solutions will be discussed in the reflection section.

# **Building**

This section details how I planned and designed the PCB.

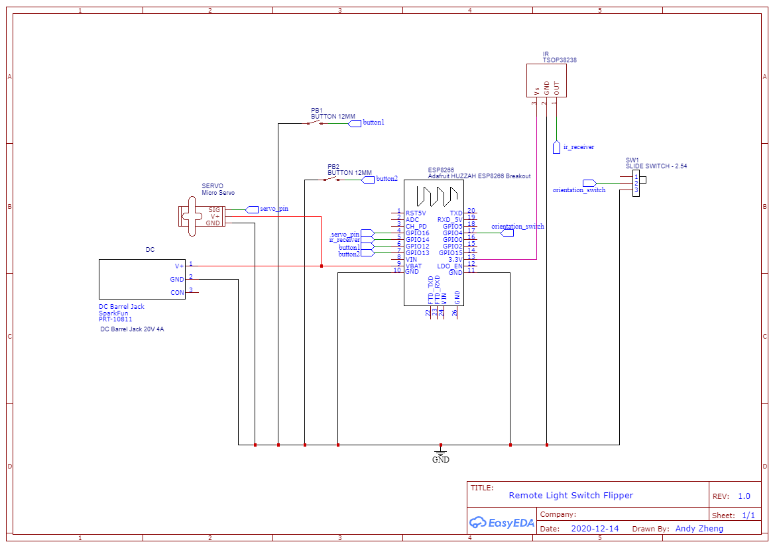


Fig. 18. EasyEDA PCB Citcuit

## Power Delivery Interface

On the breadboard, I was able to insert a power and a ground wire into the rails to supply power. However, that would not work on a PCB. I would need a more reliable way to form a connection. A very common interface for delivering power is through a 2.1mm barrel jack. I used a standard aftermarket barrel plug and soldered on the power and ground wires from my battery pack. Then I verified the correct voltage output using a multimeter. To pair with the barrel plug, I got an aftermarket barrel jack that can be soldered onto a board. I used a multimeter to check which contacts were power and ground so I would know how to route the PCB traces.

## Designing the PCB

To design the PCB, I used EasyEDA. With their design tools, I can build a circuit schematic and then convert it into a PCB that has the appropriate component holes. I selected and placed components that most closely represented my hardware and wired them together using the design tool. The circuit schematic I created is shown in Fig. 18. The price tag-shaped icons, called a net port show how the components are connected without having to use a visible wire. The net ports sharing the same labels are connected to one another. After designing the circuit, I converted it into a PCB so I can begin placing the component holes and route them. I dragged and rotated the component holes so that the system will take up minimal space but still be intuitive to the user. The up button labelled as PB1 in Fig.19A is on top with respect to the user. Then I routed the traces, making turns at 45-degree angles. Since each component required a connection to a common ground, I routed those traces on the bottom layer so they would not intersect the other traces. The remaining traces were routed on the top layer. Text identifying the component locations were placed on the top layer where they could be seen. The width of the traces was 0.6mm. The final PCB dimensions were 48mm x 65mm x 1.6mm. The top and bottom of the PCB are shown in Fig. 19A and Fig. 19B, respectively.

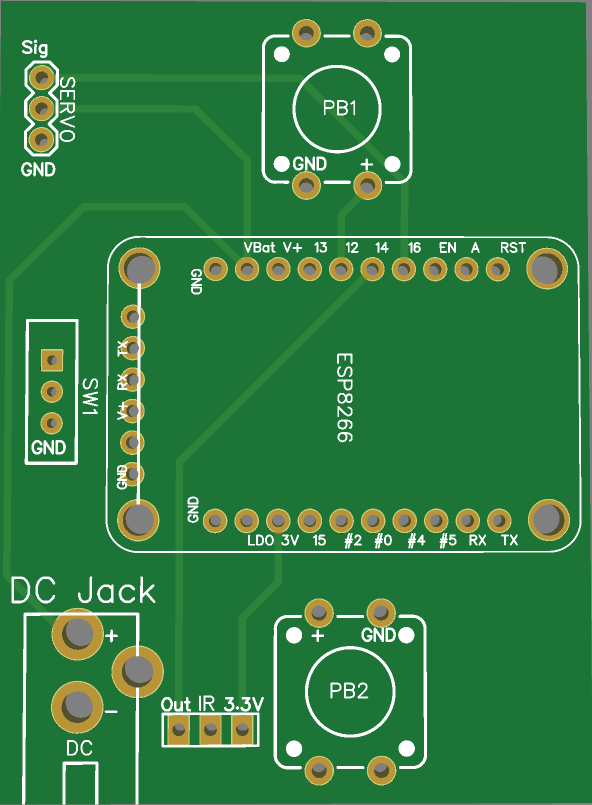


Fig. 19A. Custom PCB Front (In Mounting Orientation)

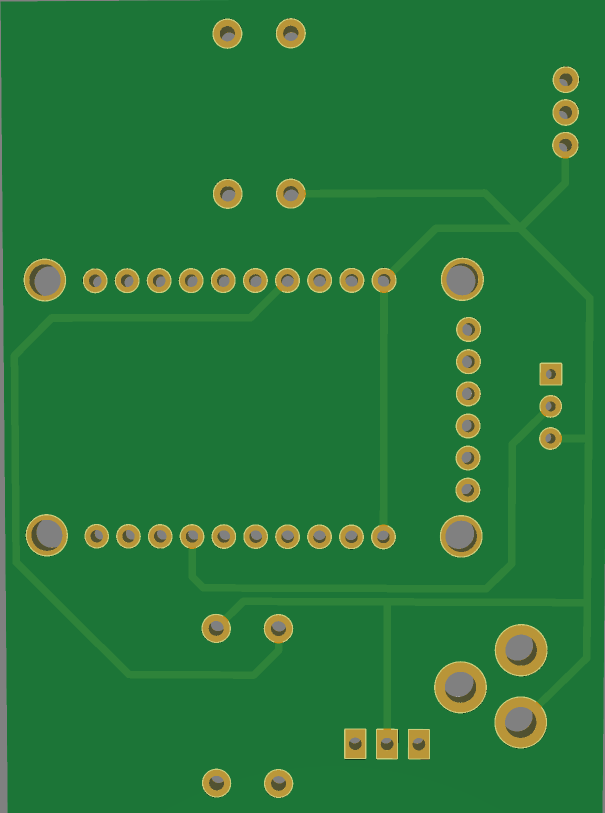


Fig. 19B. Custom PCB Back

## Testing the PCB

Before submitting the order for the PCB, I wanted to make sure that I had placed the correct component holes. I printed out the PCB to scale on a piece of paper and lined my components up to the holes to check that they would fit. After verifying, I submitted the order to JLCPCB for manufacture. I received the PCBs two weeks later. Before soldering on my components, I wanted to first make sure that the traces were correct. I visually inspected the board and then using a multimeter, I tested the continuity of the traces between the holes. There were no issues to be found.

## Soldering the Components

I used female to male pin headers for the ESP, so it can be plugged into the board instead of being permanently soldered on. This will make it removeable. For the IR sensor, I soldered it in between the DC jack and PB2 (Fig. 19A) and bent it 90 degrees up so that it would be facing outwards when the system is mounted parallel to the wall. For the servo motor connections, I soldered on a 3-pin male header so it can be plugged in. Component placements are outlined on the PCB and they were soldered accordingly. After everything was soldered, I verified that the system works as expected. The PCB with its populated parts is shown in Fig. 20.

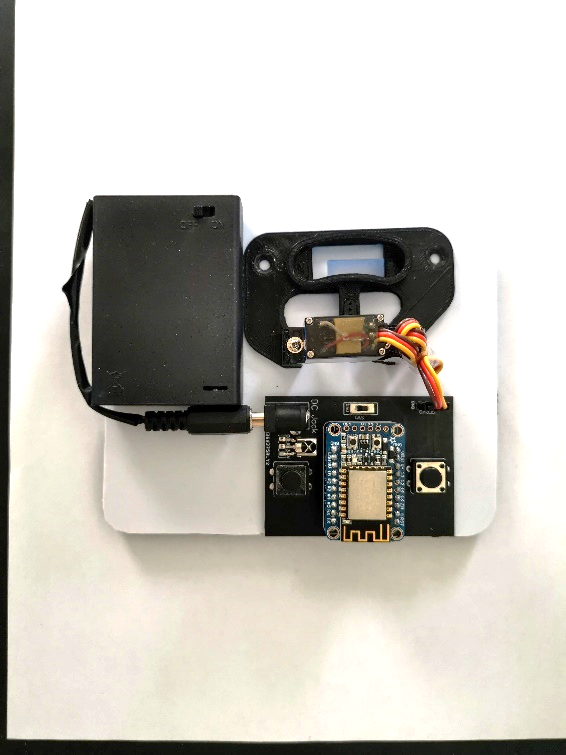


Fig. 21. Final Product

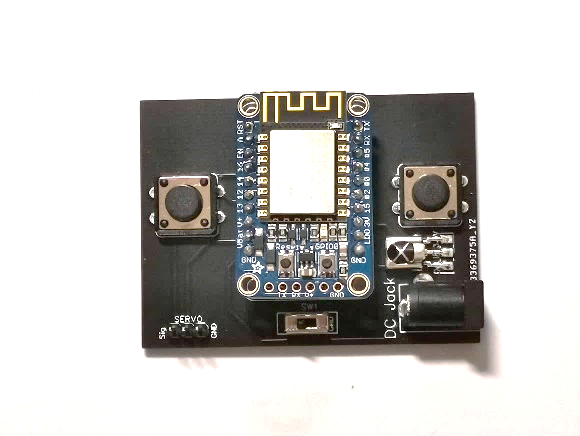


Fig. 20. Components Soldered Onto PCB

## Finishing the System

The final step is to make the system mountable. I cut a 0.5mm-thick sheet of plastic to a size large enough for the 3D printed servo base, battery pack, and circuit board. Then I drilled two mounting holes in the plastic that will allow me to mount it using the two screws from the light switch faceplate. Finally, I adhered the battery and the circuit board to the plastic sheet close enough for them to be connected. Both were placed close to the servo mount. The result is shown in Fig. 21. Fig. 22 shows how the system is mounted.

# **Reflection**

## Evaluation of Results

The finished product has some limitations that would unfortunately make it impractical to use. For one, it is limited by the battery life. When using three AA, the system would only last about two days. Secondly, the device would not work on light switches that require a lot of force. Thirdly, the system cannot connect to a public Wi-Fi. This makes it less useful in a college dorm. Finally, there is a bug that occasionally causes the arm to suddenly rotate in an unexpected direction after it returns to its default position. I have closely inspected the code but was unable to find any



Fig. 22. System Mounted on Light Switch

problems there. I suspect that this is caused by a noisy PWM signal that controls the servo motor’s rotation. A capacitor may be able to resolve this issue, but more research would be required. After this bug occurs, the system will continue to work as normal after it is reset to its default position. Unfortunately, I was unable to build a system that has a practical use due to its many drawbacks. However, it can be used as a concept and I was able to learn from the experience. The problems found in this design can be fixed in a second version and improved upon.

## What I Enjoyed

The part I enjoyed most about the project was designing the PCB. At my job with L3Harris, I worked with a lot of complex circuit boards. It was a revelation to learn about PCB design and to design one of my own. See my design go through the manufacturing process, arrive with quality that exceeded my expectations, and work as planned was exciting. Having this knowledge is extremely useful and I plan to use it on more projects in the future.

## What I Found Difficult

The most difficult part of the project were the consequences of having to order most of my parts online. It would take several days for ordered parts to arrive. Failing to do enough research and ordering the wrong part wasted a lot of time. One example of this was with the servo motors. When I ordered the first servo motor, I had only considered the size and not the amount of torque it could provide. I found that it was not strong enough to move the switches in my dorm room. The next servo I purchased was better, but still did not have enough torque for the light switched in my room. I ordered a third servo, but it was only as effective as the second one I ordered despite having a higher advertised torque. That was the strongest motor I was able to find with the correct size. I had to accept that there were going to be light switched that my device would not work on. The time it took to order the first servo and eventually getting the third took about 2 weeks. There were many other instances of waiting for parts to arrive. Most of the time throughout the project’s duration was spent waiting.

## Possible Improvements

### Power Supply

The biggest issue with the system is that the batteries are insufficient for continuous use. Therefore, the greatest improvement to the system would be with a longer lasting power source. One method is to use a 5V power adapter plugged directly into a wall outlet. This will allow the system to operate indefinitely but would require that an outlet be nearby. The other possible solution is to use a portable charger commonly used to charge smartphones. These are typically high in capacity and are rechargeable. Using those, they would only need to be recharged every few days. The drawback of this would be storing the charger. They are bulky and would look unsightly mounted on a wall. It is possible to adjust the system to use either solution but may restrict its usage to only rooms with an outlet nearby or a place put the portable charger.

### Internet

Another possible improvement is adding the ability to control over the internet from any place in the world. This is the concept of the internet of things (IoT), where the technology is connected to the internet to exchange data. The hardware on the ESP-12E is capable of this, however, there would be a lot of setup involved and I would have to pay for website hosting.

## Lessons Learned

I chose several features for my project that would require I learn about in order to implement. One was to solder my project on a PCB. I wanted to learn about the PCB manufacturing process, so decided to create a situation where I would have to learn it by making it a feature requirement. Another example was adding Wi-Fi features, specifically designing the user interface for remote control. I knew that I would have to learn HTML and CSS. In the end, I came away with an understanding of PCB design, HTML and CSS languages, and a better understanding of microprocessors.

# **Additional Information**

## Video Demonstration

<https://www.youtube.com/watch?v=Jz_OZMiVXgk&feature=youtu.be>

## Project GitHub