

Cloud-Based LightSwitch

Edgar Lopez-Garcia

Professor Kastner

CSE 145 Spring 2016

Abstract

This paper discusses the research, implementation, and contributions achieved from the Cloud-Based LightSwitch project. The goal of the project was to enable cloud-based control of house lights in an economic manner. To promote this goal, cheap widely available hardware components like the ESP8266 WiFi module and Arduino Pro Mini 3.3v were utilized to make this goal into a reality. The research of this project led to a documented way to easily use the ESP8266 WiFi module for Internet-Of-Things (IOT) projects, which is talked about in this paper. With the ESP8266 providing WiFi connectivity, the Arduino Pro Mini is used in conjunction to power calculation heavy operations and provide control of components connected to it. The ESP8266 WiFi module paired with the Arduino Pro Mini, provides a robust platform for any IOT project.

Introduction

The goal of the Cloud-Based LightSwitch project was to research a way to control house lights from anywhere in the world in an economical manner. The main benefits of being allowing control of house lights worldwide include: reduced electricity bill charges, prolonged bulb lifespan to reduce waste, and reinforced home security by making people think someone is home. With economic-cost in mind, the components of the project were chosen carefully to allow for the most price-conscious components without sacrificing easy usability. For this reason, the ESP8266 Wifi module, along with the Arduino Pro Mini were chosen. At the time of writing, the ESP8266 can be purchased for as low as \$6.95 from a reputable seller^[1]. The Arduino Pro Mini 3.3v used in this project can be purchased for \$9.95^[2]. Together, the total cost of the project sums up to \$16.90, which is much more cost-effective than other products that provide a similar functionality. Note that these components can be purchased for a reduced price from 3rd-party resellers. However, quality and reliability are some things to consider when purchasing through these outlets.

There are a couple of reasons that a user would want to control their house lights from anywhere in the world. One reason is to lower the electricity bill by not wasting electricity on having house lights on when they are not in use. The table below shows a breakdown of the cost of having a fluorescent and incandescent bulb on 24 hours a day for a whole year. The data and rates used to calculate this cost was used from SDGE.^[3]



 Fluorescent Bulb (13-18 watts)	 Incandescent Bulb (100 watts)
1¢/hr	4¢/hr
\$87.60 Annually	\$350.40 Annually
24,000 - 36,000 hours	750 - 2,000 hours

Table1 Comparing Fluorescent and Incandescent bulb yearly cost

From the table we see that wastefully leaving house lights on can be costly in the long run. Considering that an average household has around 12 lights, we can see that the cost rises drastically. If a user were able to check if

they accidentally left a light on, and turn it off remotely, then the user could save a lot of money on their electricity bill.

Another reason a user would want to remotely control house lights is to increase the lifespan of said lights. In connection with the cost problem, this will also save the user money because they would have to buy less light bulbs to replace when they burn out. From table 1, we see that on average a fluorescent bulb will last up between 24,000 - 36,000 hours, while an incandescent bulb will last 750 - 2,000 hours. By allowing the light bulbs to not wastefully be on, the lifespan could be expanded further. Since users would have to buy less bulbs with the extended lifespan, less waste would be produced, which is something that benefits everyone.

Finally, controlling house lights remotely increases home security. Users can switch lights on and off remotely to give the effect that someone is home. This is a deter for burglars or home invasion attempts as they will less likely target a house in which the residents are present. This saves the user from valuable items being stolen from their home.

The outcomes of this project provided not only a working proof-of-concept but resources for usage in other IOT projects as listed as follows:

- Organized documentation and usage of the ESP8266.
- Interfacing instructions between ESP8266 and Arduino Pro Mini on both hardware and software level.
- Android application that sends HTTP requests to ThingSpeak server.
- Setting up a ThingSpeak server for your IOT project.
- Parsing ThingSpeak responses to control light based on light state in server.

Technical Material

Talk about all the components you had to use for prototyping the project. Talk about the process and how everything connects. Use A LOT of pictures to show all this stuff.

For this project, multiple technologies were integrated in order to deliver a working solution. Figure 1 shows the roadmap for how these technologies interact with each other.

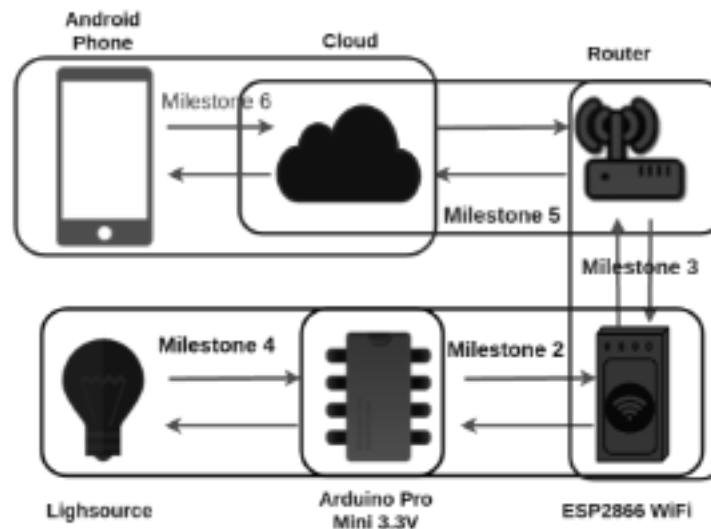


Figure 1

As you'll notice by figure 1, the light bulb, and ESP8266 interface with the Arduino Pro Mini. The Arduino acts as the medium for control. On the hardware side, both the ESP8266 and light interface with the Arduino, which acts as the "brain" of the project. The arduino uses the ESP8266 to connect to the ThingSpeak server and grab light status information that is stored on the server. The light source relies on signals from the Arduino that tell it to turn

on or off. The arduino knows when to power on or off the light source based on the type of light status it gets from the ThingSpeak server.

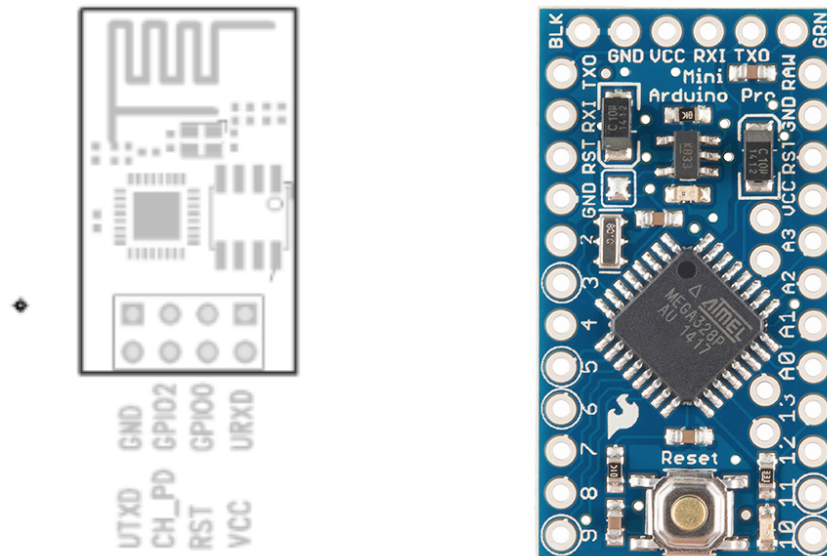


Figure 2 The ESP8226 (left) and Arduino Pro Mini 3.3V (right)

Figure 2 shows the ESP8266 and Arduino Pro Mini. Milestone 2 involved interfacing both components through hardware and software. For the hardware side, the Arduino and ESP8266 needed to communicate data with each other by connecting the TXD and RXD from the ESP8266 to pins in the Arduino that accepted serial communication (which is configured in software). Figure 3 shows the hardware connections between the ESP8266 and the Arduino.

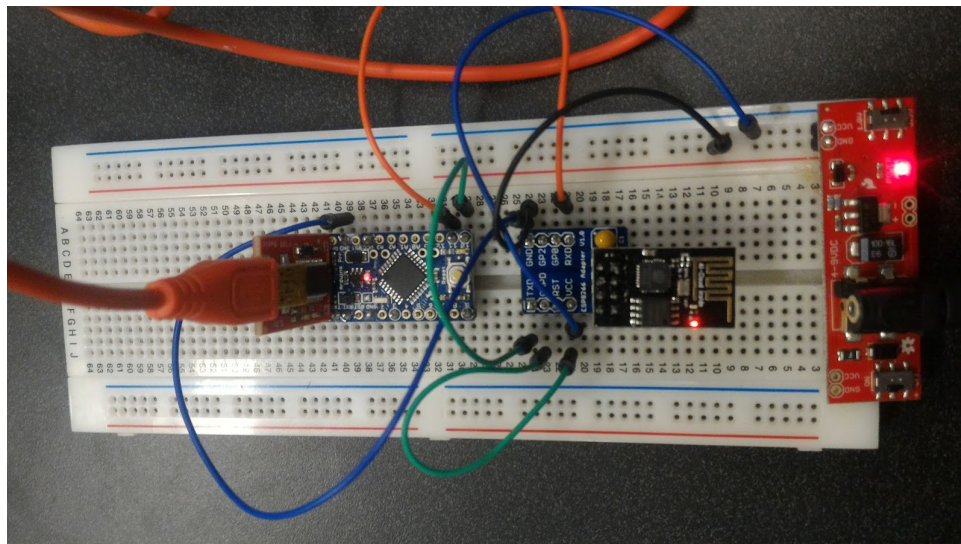


Figure 3 Hardware configuration of system

You'll notice from figure 3 that there are other components introduced to the system in order to make it function. In particular, the FTDI Basic Breakout, ESP8266 Prototyping Adaptor, and Breadboard Power Supply Stick 3.3V were used for prototyping the system. In a real-life deployment, only the Arduino and ESP8266 would exist. However, these extra components are crucial for prototyping and testing the system. Figure 4 shows these three

components separately. The FTDI Basic Breakout is used to communicate between a computer and Arduino. This component is absolutely essential because it is the method of programming the Arduino with the code that will control the light and communicate with the ESP8266. The FTDI Basic Breakout is seen in figure 3 to the far right connected to the Arduino and USB cable. The ESP8266 Prototyping Adaptor is not an essential component to the project; however, you will find that the stock ESP8266 is not prototyping-friendly. A custom adaptor can be made, in which I attempted to do, but the prototyping adaptor is cheap enough that it is worth getting one if it means avoiding the effort to create your own adaptor. Furthermore, it is more safe to use the adaptor as you make damage the ESP8266 when creating your own custom adaptor. The ESP8266 Prototyping Adaptor can be seen in figure 3 under the ESP8266. Both the ESP8266 and Arduino require a 3.3V power supply. The BreadBoard Power Supply 3.3V can plug directly into the ESP8266 and Arduino to power them and had the added benefit of being able to be used easily on a breadboard. It can be seen on the for right side in Figure 3.

ESP8266 (Through adaptor)	Arduino Pro Mini 3.3V	BreadBoard Power Supply
VCC	-	VCC
GND	GND	GND
UTXD	PIN 10	-
URXD	PIN 11	-
CH_PD → VCC	-	-

Table 2: Hardware connection between ESP8266, Arduino, and BreadBoard Power Supply

Arduino Pro Mini 3.3V	FTDI Basic Breakout
VCC (TOP)	3V3
GRN	GND
TX0	RX0
RX0	TX0

Table 3: Hardware connection between Arduino and FTDI

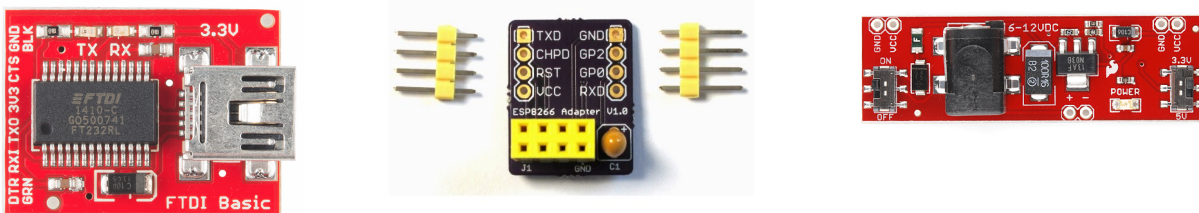


Figure 4: FTDI Basic Breakout 3.3V (Left), ESP8266 Prototyping Adaptor (Middle), BreadBoard Power Supply Stick 3.3V (Right)

Table 2 and 3 shows the hardware connection between the components used in this project. The ESP8266 are connected with each other to share power and data. The BreadBoard Power Supply connects to both components in order to power and ground them correctly. The FTDI Basic Breakout is connected to the Arduino Pro Mini through the headers. As mentioned before, the FTDI Basic is required to program the Arduino. I left it connected for the whole project as it powered the Arduino directly and also made it easier to debug the Arduino code because it allowed me to view the program output in real-time.

There are quite a few tutorials on using the ESP8266 and Arduino Pro Mini for IOT projects. However, most are unorganized and do not give you the full picture. The result is that it is difficult to integrate the same methodology from tutorials to your own projects. The research done in this project provides an organized documentation of the usage of the ESP8266 that will lend well to expandability to other IOT projects, as it is the core for any IOT project. Using a microcontroller such as the Arduino Pro Mini gives great flexibility and control of components within the project. Once the Arduino, ESP8266, and computer are connected together, communication to the ESP8266 takes place through AT+ commands^[4].

Table 4 shows the AT commands I used in this project in order to communicate with the ESP8266 and allow it to be controlled by the Arduino.

Command	Function
AT+RST	Reset Module
AT+GMR	Print firmware version
AT+CWLAP	Search available APs
AT+CWJAP	Connect to specified AP
AT+CIPSTART	Start a TCP connection to specified server
AT+CIPSTART	Start an HTTP request
GET	HTTP request sent after connection to server

Table 4: AT+ commands used to communicate with the ESP8266

To enable remote control, the Android platform is used in this project. The Android platform is widely used around the world. With its robust application-building platform, it is possible to rapidly develop powerful applications in any domain. It's SDK and tools that are provided for developers makes it easy to integrate the Android system to many projects. In particular for this project, it acts as a control interface for the light switch. Since mobile data can theoretically be used anywhere around the world (so long as there is mobile reception), the Android platform fits perfectly with this project to enable users to control their home lights from anywhere in the world. The Android application serves two purposes for this project:

- 1) Give real-time house light status to users.
- 2) Allow users to control house lights on-the-fly.

Thus, the Android application is a crucial component of the project, and its functionality and implementation is documented in this paper.

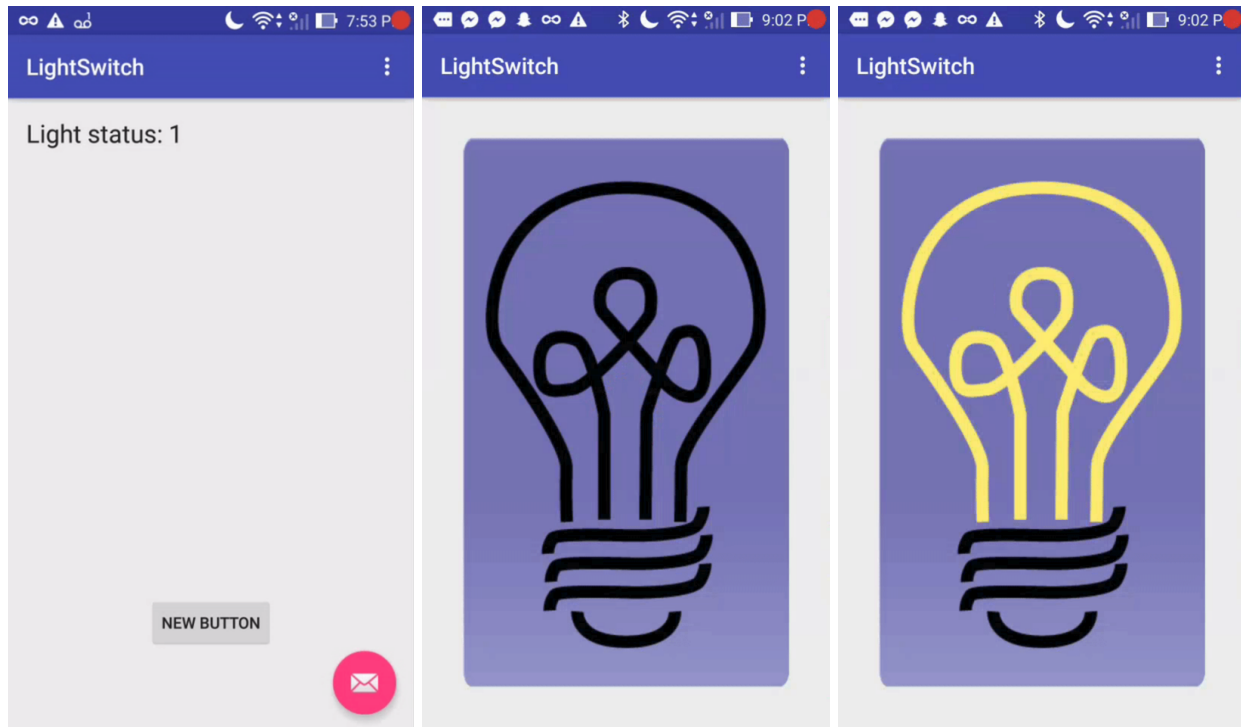


Figure 5: Arduino application user-interface. first iteration (left) and second iteration (middle, right)

Figure 5 shows the iterations of the user-interface of the Android application. The first iteration is the default UI that comes with an Android activity. I added the light status text to indicate feedback from the ThingSpeak server. Iteration one of the UI was essential used to test the correctness of the infrastructure for the Arduino application. Once I was confident the infrastructure was implemented correctly, I focused on adding graphics and streamlining the user-experience. I did this by making the whole screen a button because this emphasises that the app is to switch on/off similar to a lightswitch you would find at home.

Milestones

This section should describe the milestones that you stated at the beginning of the quarter, the revisions that you made in the middle of the quarter. It should provide details on the milestones that you did and did not achieve, with an explanation on why you did not achieve any milestones. It should also discuss the problems that you encountered over the quarter and how you managed them. This is not in a typical technical paper.

Initial milestones

- Milestone 1: Solder chip for easy prototyping
- Milestone 2: Interface Arduino and WiFi communication
 - Hardware connection
 - Software connection
 - Communication (Parsing commands)
- Milestone 3: Interface WiFi chip and router communication
 - Communication
- Milestone 4: Light-switching mechanism on Arduino board
 - Allow WiFi to control this light-switching mechanism
- Milestone 5: Interface cloud-based communication
 - Acquire server (maybe thingspeak)
 - Communication to server (Arduino → WiFi → Router → Server)
- Milestone 6: Android Application

- Communication with cloud
- Register light feature
- Display light status
- Turn on/off

Revised milestones

- Milestone 1: Solder ESP8266 to adaptor for easy breadboard prototyping
- Milestone 2: Interface Arduino and ESP8266 communication
 - Hardware connection
 - Software connection
- Milestone 3: Connect ESP8266 WiFi chip to Internet network
 - Setup server on ESP8266
- Milestone 4: Light-switching mechanism on Arduino board
 - Let commands received from ESP8266 turn on/off current on pins that light is connected to
- Milestone 5: TCP connection from ESP8266 to ThingSpeak
 - Acquire ThingSpeak server
 - Receive data from ThingSpeak server (Server → Router → ESP8266 → Arduino)
- Milestone 6: Android Application
 - Communication with ThingSpeak server
 - Change light flag in ThingSpeak channel
 - Display light status

These milestones were grouped up into tasks and mapped to a gantt chart to impose target deadlines for each task, allowing milestones to be completed within a 10 week period. The tasks created and allocated in the 10 week timeline are shown in figure 5. The Arduino & WiFi task includes milestones 1-3. This task is the core part of the project. It is required before any further development can be made because communication between the ESP8266 and Arduino is needed to connect to the ThingSpeak server. The task WiFi & Router is the interface between the ESP8266, and ultimately the Arduino, and the ThingSpeak server. This task is what connects the ESP8266 to the outside world via a home router. The Cloud task deals with any operation required to provide cloud functionality to the project. This includes setting up the ThingSpeak server and communication with it. This task is documented in milestone 5. The Switching task encompasses milestone 4. This task is to accomplish the switching functionality between the Arduino and light. The result from the request to the ThingSpeak server from the ESP8266 are parsed by the Arduino and the light is turned on or off depending on the state of the value of the light recorded on the ThingSpeak server. The Arduino task is delegated with milestone 6 in that it encompasses the development of the Android application. Finally, tasks Software and UI/UX are auxiliary tasks that are not required for the system to function. These are tasks that would be nice to finish if time permits.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Research										
Design										
Prototyping				Arduino & WiFi	WiFi & Router	Cloud				
Software				Arduino & WiFi	WiFi & Router	Cloud	Switching	Switching/Android	Android	
Quality Control									Software	UI/UX
Final Presenting										

Figure 5: Gantt chart of the project

The Gantt chart was designed so that there was a buffer in case some tasks took longer to complete (as they could bleed into the auxiliary tasks slot). For the most part, the tasks were on track with the proposed Gantt chart, however I did run into some problems that slowed down development for some tasks. Connecting the Arduino to the computer took some time to work because sometimes it would work and I could send commands to the ESP8266, but other times it would not work. At first, I thought the ESP8266 had been burned out since I was tinkering with the power supply to it. I ended up buying a second ESP8266 WiFi chip. It took a week to ship so the Arduino & WiFi task halted. Eventually I got it to work and continued development after a costly week. Another problem I faced that took an extra week to complete was the Switching task. Setting up the ThingSpeak server was

pretty straightforward. However, there was not much documentation on how to receive HTTP responses with the ESP8266; thus, it took a while for me to figure that out. In the meantime, I worked on the Android application while trying to figure out the Switching task. To my surprise, I was able to create the Android application within two days, rather than a week as I originally thought. This saved me a lot of valuable time in development and allowed me to catch up on task development that was falling behind.

Overall, I was able to complete all the core tasks for the project. This means that all milestones were completed. However, I was not able to fully fulfill the auxiliary tasks. In particular, the UI of the Android application could have been much better. The software on the Android side was not very organized, and is not in a state to allow expandability. I would have liked to clean up the code and apply architectural and design patterns to make it easier to manage and expand.

Conclusion

In today's modern world, we take light for granted. We waste it by leaving it on when it is not in use, and this causes many problems: it makes electricity bills costly, and damages the environment by producing more waste because light bulbs have to be replaced more often. The goal of the Cloud-Based LightSwitch project is to reduce these problems by allowing users to remotely control their house lights from anywhere around the world. To do this, the Arduino platform, along with a "smart" switch consisting of the Arduino Pro Mini and ESP8266, was used since it is one of the most accessible and widely used platforms. It was chosen with the idea that users always have their Android phone with them in an accessible place at all times. By allowing users to control their house lights with their Android phone, it introduces accessibility to control the house lights because shutting them off is only a tap away.

Much of the work done in this project builds on the groundwork needed for this functionality to be used by users. This project researched and implemented the core technology that is needed for this product. In particular, the hardware requirements were implemented with the software to control the light switching mechanism along with communication to the server. A ThingSpeak server was used in order to store the state of the light at home and to respond to requests to change the state (from on to off and vice-versa). Finally, user control was integrated with an Android application. The Android application communicates directly with the ThingSpeak server in order to change the state of the light.

Further work would include refining the software architecture so that it is easier to maintain and expand. With user-experience as a goal of this project, further research can be done on designing good user-experience for this application. In particular, the user-experience would be implemented on the Android application since that is the component that the user will most interact with. This would mean minimizing the effort the user asserts in order to turn the light on or off. There is some extended functionality, above the core functionality, that I would have liked to add, but did not have time to research. For example, it would be great to have an easy way to add multiple lights to the system so that the user could have full control of all these lights. This would be an essential requirement to have for a product that would actually reach users. Another feature that would be nice to add is dimming support for the light, rather than just having digital on or off.

In conclusion, this project required the integration of multiple technologies in order to achieve an end goal and produce non-static systems. All of the core technology required for this product was integrated, however there is a lot of room for improvement, namely code cleanup, better UI integration, and more functionality for users. A lot can be learned about IOT projects through the research of this project, and hopefully it is useful to someone who is interested in their own IOT project.

References

- [1] <https://www.sparkfun.com/products/13678>
- [2] <https://www.sparkfun.com/products/11114>
- [3] https://www.sdge.com/sites/default/files/documents/savingenergy_0.pdf
- [4] <http://www.pridopia.co.uk/pi-doc/ESP8266ATCommandsSet.pdf>