

# Project Z

Rui-hao Oscar Qiu, Wong Guan (Diego), Alvin Yan, Pedro Coutin

## Abstract

Less exposure to sunlight and getting less sleep leads to many ailments. Research has shown that people who experience reduced exposure to sunlight to be correlated with mental ailments such as Seasonal Affective Disorder (Lam and Rosenthal). Lack of sleep or sleep quality has many proven consequences which range from increased risk of cardiac disease to decreased cognitive ability (Wilkinson). These two factors also contribute to other totems of health which are commonly known such as vitamin D generation, and should be carefully taken care of by everybody. This project aims to provide people with a better and cheaper way of taking care of both by using 3D printing, the Arduino microcontroller, wireless communication, and the smartphone platform.

## Introduction

This project has some personal roots. The founder had experienced bouts of insomnia in which he daily consumed half the recommended dose of sleep and experienced one of the worst years of his life. In layman math, it was better to be functioning cognitively at 100% for 16 hours of the day than be scooting along life at 50% for 20 hours awake. This prompted him to follow sleep journals and research ways to better improve hygiene. Of all the things that were learned, the only thing that could not be controlled well was sunlight.

This is important because sunlight is the most significant zeitgeber. Zeitgebers, partly influencing the project name, is defined as an external or environmental stimulus that entrains or synchronizes one's biological clock. Other zeitgebers include social activity, food, temperature, exercise, and drug intake. With light being the most significant zeitgeber, we sought a solution to manipulate natural sunlight, as this spectrum of light is the most familiar and trained type of light in wake-sleep regulation for the average person.

So far there have only been projects that are designed to mimic sunlight such as the Philips \$120 Wake Up Light and the \$60 Nature Bright Sun Bliss. They both have the similar "feature" of mimicking natural sunlight.

Our project manifests itself as an implementation of a solution that saves energy by using actual sunlight, is more effective, and can function as more than just a wake up tool. The synopsis of our project is that the user will use our smartphone app, to manually control the blinds or preferably set

a day, time, and position their blinds should move to automatically. It will use wireless communication in the form of bluetooth low energy as the preferred method of communication. The high level hierarchy of components that we finalized for this project is as follows:

- Physical structure:
  - Device support
  - Battery
  - Microcontroller
  - Motor system
  
- Code base:
  - Arduino
  - iOS app

Now that there is a general idea of how it is going to work, let us elaborate on the the high level design choices of our project with respect to sleep science. The effectiveness of our project in helping one getting better quality sleep comes from the strengthening of their circadian rhythm, otherwise known as the wake-sleep cycle, with the most effective zeitgeber, sunlight. The wake-sleep cycle, when strengthened with strong enough stimuli consistently will allow one to fall asleep and wake up faster. This is key in increasing one's quality of sleep. Say a person spends 8 hours in bed, but only sleeps 6 hours. This results in a 75% sleep efficiency, in which one of the goals of this project is to increase this efficiency. With a strong wake-sleep cycle one will be able to fall asleep faster, as well as get up at the right time. This is the second health factor that we better with our product. There are 5 stages of sleep, stages 1 to 4 and REM (Rapid Eye Movement), in which stage 1 and REM are the objectively the best times to wake up as the brain is in a state that most resembles wakefulness. When one wakes up groggily, this is due to waking up in a stage 2, 3, or 4, in which the body needs to adjust a larger amount to get accustomed to consciousness. A well trained wake-sleep rhythm will allow users to wake up at the last stage 1 or REM sleep they encounter.

The advantage of an automated natural light manipulation device is that the user will now be able to control the amount of sunlight, the time of exposure, as well as the length of exposure using our smartphone app. This ensures that they will be able to **consistently** awake to sunlight at the same time everyday as well as prepare to go to sleep in as little light as possible. In other words, the user will now have the choice to go to sleep in darkness **as well** as waking up with sunlight. Before one would could only chose one, meaning that they could either go to sleep well due to the blocking out of outside ambient light or wake up well due to the letting in of sunlight. No more choosing, now people will be able to have both! Automation also guarantees that the user no longer has to worry and remember to turning the blinds on in the morning to increase their exposure to sunlight or shutting the blinds to limit outside light at night. This of course has more applications than just sleep hygiene and overall health. With this implementation people will be able to do as they wish, whether it be a store's closing routine, a restaurant's mood lighting, or someone's personal netflix lighting.

# Technical material

An in-depth hierarchy is described below:

1. Physical structure
  - a. Device support
    - i. 3D printed casing
    - ii. 3D printed adjustable attachment limb
    - iii. 3D printed blinds twister adapter
  - b. Battery
  - c. Microcontroller
    - i. Arduino
    - ii. Bluetooth shield
  - d. Motor system
    - i. Stepper motor
    - ii. 3D printed bevel gears
2. Code base
  - a. Arduino
    - i. Bluetooth protocol
    - ii. Instruction sender/parser
    - iii. Time-tracker
  - b. iOS
    - i. Swift
    - ii. C bridge
    - iii. Bluetooth protocol

Starting from top to bottom we will go over the specifications of each component and subcomponent that work together to create Project Z.

## Physical structure

### Device support

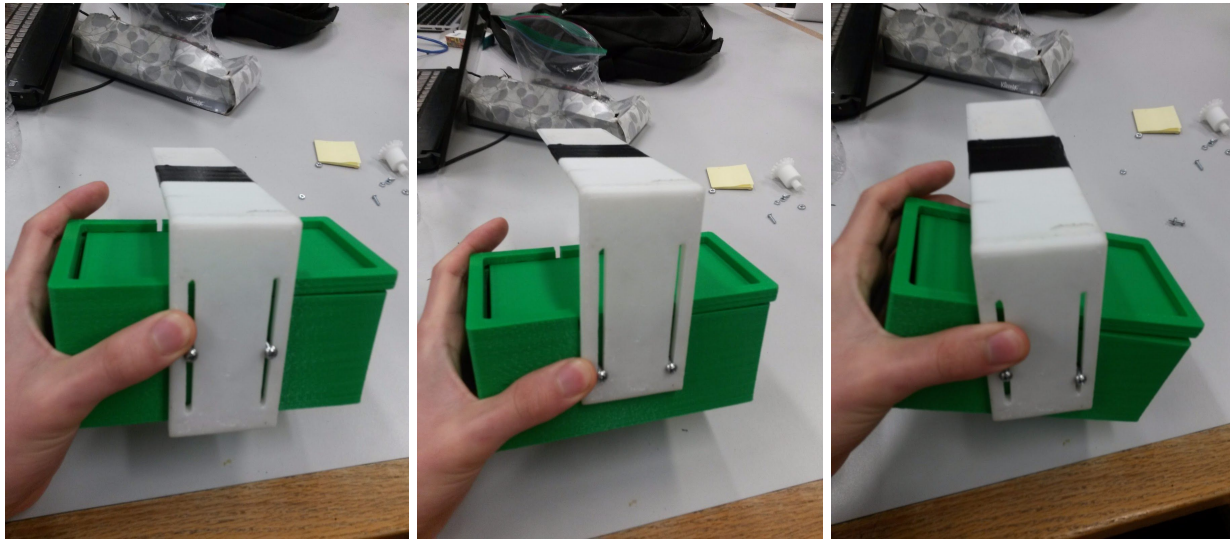
#### *3D printed casing*

Physically we need a support system for our machinery and logic that is housed. We thought of many types of systems but went with one that would allow us some leeway if we decided on using more than one arduino shield or a bigger motor. The print came out to be over 10 hours on high quality settings. Since most places don't allow overnight printing because if something goes wrong then it would be wasting a lot of fiber as well as making a mutated monstrosity that's a pain to peel off

the base plate below is a picture of the print which my friend was able to do for me in Envision studio makerspace on campus. The top of the casing makes for a very nice fit for the battery.

### *3D printed adjustable attachment limb*

The next physical component, the attachment limb is made to attach our device to a user's blinds. It takes advantage of the fact that most blinds are not fitted flush with the top ceiling. This allows our device to snugly fit in between a blind unit which aligns our blinds horizontally with the twisty gear of the blind, while giving the user a range of stable adjustment before the device slips and falls. Now that we have that correct horizontally, we need to position it correctly vertically. To solve this problem, we made two vertical, parallel ovals the size of with a nut and bolt system of tightening as pictured below. This makes it possible for users to position their device over the blind system in analog, allowing extremely flexible positioning. When adjusted to the perfect position, the user can then tighten to fix the configuration semi-permanently.



### *3D printed blinds twister adapter*

Then we have the adapter that we use attach to the blinds to make the twister compatible with our machine. It's a small piece as shown below that is friction-fitting using a screw that you tighten that presses into the blind twisty. Once tight enough, it becomes an extension of the twisty, forming a twisty with a gear, or formally known as a gear twisty twisty. (just kidding)



## Battery

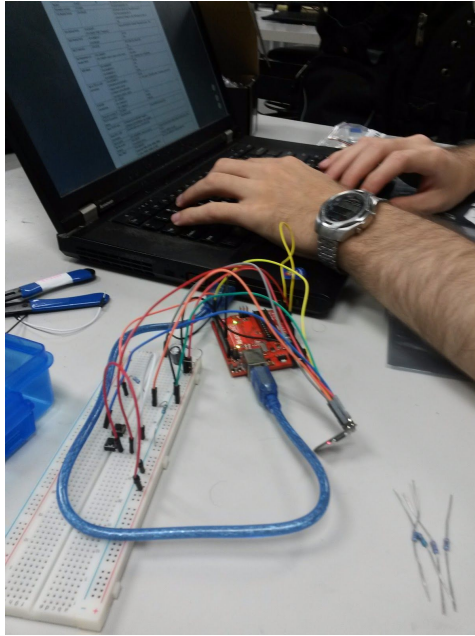
For power source, we decided to use a USB rechargeable battery in our final implementation which is enough to satisfy our use cases. It's a 5 volt, 2 amp battery that is slim enough to be invisible if you look at the device from the bottom, and it matches our Arduino energy restrictions.

## Microcontroller

### *Arduino*

Our microcontroller of choice is the Arduino Uno due to its variety of shields and abundance of online support, as well as being just smaller than our battery so it wouldn't add to the size of our casing. Our first microcontroller that we theorized was the Onion, but we ditched that in favor of the much more mature Arduino as well as having a member in our group that has Arduino experience. The first Arduino that we bought worked well, but we got another microcontroller to test our wifi implementation separately. So we got a CuHead Pro WiFi/Ethernet Shield, in which we first tried a wifi implementation. However, this didn't work as every time as when we send some data to the wifi shield, we always read some garbage data. After returning this and getting a new one, we thought it was unlikely that we would get two messed up shields in a row. So after the second one still came up with garbage data, no matter what we sent, we thought it was Arduino's problem. Therefore we used the IEIK UNO R3 Board ATmega328P, another microcontroller, to test our wifi shield. Since we didn't expect the second board to be faulty and spent almost 20 hours trying to get this setup to work. However, we gave up on this so we returned the second iteration of the wifi shield and bought a wifi

chip (ESP8622), not a shield this time. We still ran into corrupt data errors, so the problem lay with the microcontroller or the wifi chip. So then we got another Arduino, this time another Arduino Uno and realized that it probably is the wifi chips problem as two official Arduinos is very unlikely to be wrong, especially since both work with the Bluetooth shield. In the very end, wifi didn't make it in since we didn't incorporate solar power which would make keeping Bluetooth on and wifi on more sustainable. Also we were tired of working with potentially faulty hardware. Below are pictures, and [here](#) is a link to one of our first working prototypes (speed based servo with Arduino + Bluetooth shield).



*Bluetooth shield*

Our wireless communication of choice that came out of our trial period of trying different hardware was the Bluetooth shield. We chose this over an individual Bluetooth chip as this neatly packages everything together, makes this configuration not-permanent since nothing is soldered, and opens up more pins for testing. The wifi chip, the ESP8622, was implemented via breadboard and resistors which would not fit in the box, but the wifi shield would, so we assumed the same for Bluetooth. We also chose a Bluetooth low energy shield since we wanted to minimize power consumption.

## Motor system

### *Stepper motor*

The motor system was probably by far the most tricky to get right given our inexperience with motors. Our group first started out using a Servo speed based motor (FS5106R Continuous Rotation 360), which operates in speed states, with a number being stationary and above that to be clockwise and below that number to be counter clockwise. It went quite well, with us getting a small prototype out quite fast composing of just the motor, bluetooth, and quick and ugly iphone app. We thought this

motor was going to suite our needs as we can calculate time with a the Arduino delay function, and then we can multiply speed by time to get the distance rotated. Unfortunately, about a week and a half in, we found that the number for stationary change slightly by one. We found that the week after week, the state number for stationary kept changing! Some days it would move very very very slightly. It would completely unreliable, and could not be allowed in our project since a very slight movement over 24 hours would actually be a significant movement. Miscalculation could potentially lead to our device breaking or the user's blinds breaking or making some horrid noise and throwing off future movements.

Another motor that we tried was a position based motor Servo 180 position motor. This allowed us to calculate position very accurately with the tradeoff of not being able to adjust speed. We also thought this motor was going to be the right fit until we found that there's no way physically to extend the range of the motor past the max range. Our plan going into this motor was something along the lines of:

```
if ( position == maxPosition ){
    position = minPosition;
    counter += 1;
}
else if ( position == minPosition ){
    position = maxPosition;
    counter += 1;
}
```

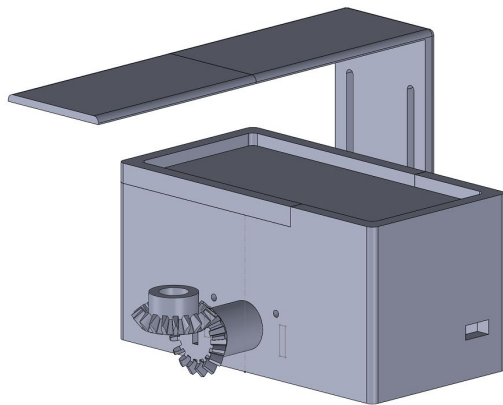
As a group of computer scientists we thought this would work, but alas if this were the case then there we be no stepper motors. Turns out that the position motor is limited in its range because of physical and hardware constraints. We couldn't just bypass this with a handful lines of code. This led us to our final motor, the stepper motor.

The stepper motor, has unlimited range of motion since the arguments it takes are +x or -x for x steps clockwise or x steps counter-clockwise. This is the middle ground compromise motor of the above. If I go +1000 and then -1000, it would be likely that we end up in our initial position, but this motor isn't as accurate as the position motor. There's a chance that the hardware doesn't behave exactly like we plan, which is unavoidable with all hardware. This has served us well ever since we adopted it and thus made it into the final version of the project. It takes up more space data wise in the Arduino as we have to store a position variable, but since we got the Uno we had plenty of space to spare.

### *3D printed bevel gears*

The reason for bevel gears, as shown below, is that we can't have a direct access of the twisty twisty as the would require the inside of the casing to be directly below the twisty twisty. This isn't feasible as the casing is a bit thick and there is very little room to spare in that area. This means that we need to have access the twisty twisty from another angle. This is why we have a horizontal gear extending from the box, to the vertical gear extending from the adapter, forming a bevel gear system. Here's what everything together looks like if everything was printed perfectly.





## Code base

### Arduino

#### *Bluetooth protocol*

Our Bluetooth Low Energy implementation is pretty standard. Our Arduino emits a signal with a unique identifier for our mobile app to pick up on. Then once connected with a device, all data sent through Bluetooth is sent to a pin, in which there is a listener there. The listener listens for any sent data, and this is how we can communicate wirelessly with our mobile app.

#### *Time-tracker*

To keep track of time on the Arduino, we track by milliseconds since the time we start the Arduino. Since the real world's time doesn't match, we get the current time in milliseconds and add the difference to the initial time to get the real world time in the Arduino. We decided to implement this ourselves, even though there is another library to make our Bluetooth side implementation as lightweight as possible. This functionality will come in useful when we read input time data from swift.



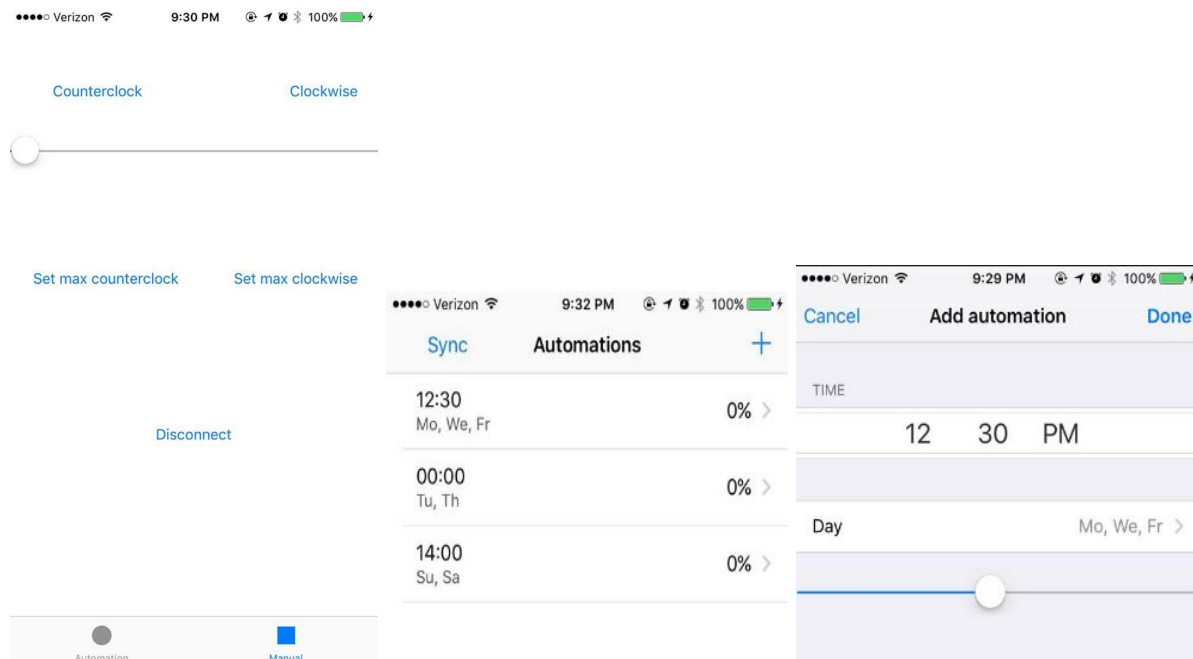
## Instruction parser

On the Arduino side of wireless communication, we receive the data sent from iOS to either move the motor immediately, or to move when the Arduino time that we describe above matches the one passed in from iOS, depending on the prefix of the data sent into the Arduino. We prefix automated actions with 'SCHED' which schedules an action to be done later and repeated until the user deletes all SCHED commands. Although not binarily maximized, e.g. 4 bits of data represent 16 unique states of data, we two types of data from our parser. If SCHED, we get a day, time, and %position. If the user isn't going to store data in the Arduino, i.e. if the user is asking the Arduino to execute a manual action immediately, then we have set maxCounter, set maxClockwise, rotate counterclockwise, rotate clockwise, and how many steps per button press. We will go into more in detail in iOS code.

## iOS

### Swift

There are two ways to approach swift on iOS. There is the storyboard approach which involves data values from GUI, as well as a lot of clicking and dragging as, and then there is the programmatic way which gives the programmer maximum control but isn't as practical for developing mobile apps since that way is best for way more complicated apps or command-line focused apps (which there are basically none in appStore. The main functionalities that we implement in swift is the user interface, which the user can use to navigate the functionalities we offer, input data for manual execution, and input SCHED commands easily. Since software is easy to add things on, we included a home screen in which you can select the method of connection, wifi or Bluetooth.



### *C bridge*

Our wifi, although not fully functional is still included with our code since we had done some work on it. Since swift is relatively and doesn't have a core wifi library, we used the DarwinC c bridge to allow wifi connectivity on our project. We demoed the ability to talk to a Servo (nonstepper motor) through wifi through a TCP client. However, this was demoed on the IEIK UNO, via resistors and breadboards as shown in this [video](#), and we didn't chose to implement wifi fully on the hardware, as it would cost extra electricity and soldering since all our wifi shields were faulty and we only had that chip working on the non-final microcontroller.

### *Bluetooth protocol*

There are two types of information that we send via Bluetooth to our Arduino from swift: immediate manual execution commands and automated commands. The immediate commands that are sent to the Bluetooth are immediately processed and sent to the corresponding pins which connect to the motor driver. The automatic commands are not as immediate and the signals to be sent to the motor driver remain latent until the time on Arduino matches that on the automatic commands. The manual control data is simple and how we have the User Interface, we send step size adjustment that is negative if counterclockwise and positive if clockwise. The automatic command is a bit more complicated. As mentioned before we prefix an automatic command with SCHED followed by 3 fields: time, days, and position, of which are delimited by fowardslash. We allow the user to set however many days they want for time and position. Position is a percentage of maxCounterCockwise and maxClockwise values, which are there to prevent the user from breaking the machine or the curtains when the system reachers of of the extremes. So if there are 100 steps between maxClockwise and maxCounterClockwise, then a 50% position would set the actual position to be 50 steps clockwise from maxCounterClockwise, and 50 steps counterclockwise from maxClockwise.

## Milestones

We had many milestones and we're proud to say that we attempted each one and learned a lot from each, independent of the success of the milestone. [Gantt Chart](#) here displays all the milestones with respect to time for each milestone. I'll go over each planned milestone here as they are described in good detail above. Since we before described from low level to high level, we'll reverse it for this description.

The first was to make a general user interface which we started off with three proposed platforms, Flutter, Android, and iOS. We had Diego on iOS, Oscar on Flutter, and Alvin on Android, however the Flutter development was abandoned since the technology was in Alpha stage and there is literally zero support for it. Oscar then went on to program with Diego as he was the only one with a Mac and Oscar had learned a bit of iOS dev before.

Next, we needed to use the user interface to send signals to the microcontroller. We were conflicted between many different choices of which signals to send, but in the end as we know, the stepper motor was implemented and thus we chose to send x amount of steps, which was either positive or negative for the manual control and a SCHED command with time, days, and position as a percentage. In the implementation of SCHED we saw saw a need for the Arduino to store the automation data instead of the phone, so that was a change we made in the middle of the quarter.

The group also had conflicting opinions on whether or not we had to implement a QR security scanner, in which we saw it being started on Android but not iOS. Another thing that fell through was the implementation of wifi wireless communication. Although not fully implemented, we decided to not use the wifi chip in our final build as it was a permanent decision due to soldering, and due to the fact that it would use more power.

On the hardware side of things, we didn't get to see a custom PCB get printed as the expert I consulted at the UCSD Prototype club meeting said that it would cost a lot of money and time to have one printed out, especially if you use the proprietary software used to design them.

And on the design side of things and on the positive side, we actually got a logo made for our project done by Amber Tang, a close friend of one of our group members. It made the iOS app look really good and we implemented it into our final video as well. We also made two videos to promote our video, one 'behind the scenes' video, and one promotional video that is apt to show to investors or recruiters. That's basically a medium level overviews of the things we overshot, undershot, and got on point; high level descriptions of these goals can be found in the Technical section.

## Conclusion

This project fit the goal we wanted to achieve, something cheap that could help people automatically strengthen their circadian rhythm. This is definitely something everybody should be interested in as sleep is one of those things that can't be observed well but actually has a huge impact on our wakeful state. We want this to help everybody and with us pursuing our degrees, we think that we'll release the code and 3d models after further improvements over the summer so everybody can benefit as soon as possible from our project.

If we were to further improve the project, we would definitely like to have a system where the user never has to charge the device because of solar energy implementations. We'd also like to have some encryption or security work done on it as blinds contribute a big amount to physical privacy and comfortableness in one's home. We also think it's a good idea to group all devices together and change their settings at once with wifi. Maybe we could catch the eye of some home automation companies that are soon rising up.

There's so many things that we learned on this project. We came in with unique sets of skills and weaknesses and faced big pressure to complete tasks. For example, this was the first mobile mobile apps we worked on from the ground up, as well as the first time Oscar was a project leader. Also most of us learned English as a second language and this was our first time since high school presenting, and we're glad we got to practice in the final presentation. Diego and Alvin were probably tied for the most time stuck on a bug, and Pedro became more comfortable speaking after seeing so many presentations. Oscar thought this class was one of the most stressful, as well as one of the most enjoyable classes that this school has to offer. Not only do you learn to be self guided, well disciplined, and team oriented in your own project, but

also you get to see other people's approaches to their projects as well. We hope others had a good time experiencing our project and will remember the laughs (and struggles) we shared in this class.

\*Sorry for going over the page limit, We added a lot of pictures in when we found out that other groups were doing it.



# References

- Lam, R. W.; Levitan, R. D. (2000). "Pathophysiology of seasonal affective disorder: A review". *Journal of Psychiatry & Neuroscience*, 25 (5), 469–480.
- Rosenthal, Norman E., et al. "Seasonal affective disorder: a description of the syndrome and preliminary findings with light therapy." *Archives of general psychiatry* 41.1 (1984): 72-80.
- Wilkinson, Robert T. "Interaction of lack of sleep with knowledge of results, repeated testing, and individual differences." *Journal of Experimental Psychology* 62.3 (1961): 263.

## **Project Z**

Oscar (Qiu) Wang, Diego (Guan) Wong, Pedro L Coutin, Alvin Yan  
CSE 145, Embedded Computing Systems, Spring 2016  
University of California, San Diego, La Jolla, California

Other paper (Oscar and Deigo):

[https://docs.google.com/document/d/1wJVAmxKY\\_3drvWKHcpiOqoxfAQ3BTSo5WPKZaJMU2FY/edit#](https://docs.google.com/document/d/1wJVAmxKY_3drvWKHcpiOqoxfAQ3BTSo5WPKZaJMU2FY/edit#)

## **Abstract**

Sleep deprivation is a common issue upon any individuals and can bring harmful and marring results during the day. Putting it simply, sleep deprivation is resulted from an individual's inefficient way of sleeping and unable to follow the circadian rhythm. Project Z strives to improve sleep deprivation by simply adjusting the scheduled lighting of the room. As a result, the user's body mechanism (specifically zeitgeber) are affected due to the change in the lighting environment. The project is used to simulate the user's sleeping pattern with lighting to aid efficient sleeping.

## **I. Introduction**

Sleeping is a natural activity for the body to recuperate after being deprived of energy both physically and mentally. During this natural process, the body follows its internal timer of a 'sleeping schedule', otherwise known as the circadian clock.

Nowadays, sleep deprivation has become one of the biggest obstacles in most individuals' daily routine. With the lack of sleep, many troubling issues have occurred in the individual's performances throughout the day. This includes the numerous amount of car accidents due to the lack of focus, and cognitive and physiological deterioration. As such, sleep deprivation can cause mental, physical, and emotional fatigue overall.

The solution to counter such daily issue is by inventing a mobile application, otherwise known as Project Z. Project Z strives to aid the user's circadian rhythm by simply simulating it with the room lighting. To boil it down by complexity means, Project Z is intended to stimulate a body's stimuli known as zeitgeber. Zeitgeber is a body mechanism that synchronizes the biological rhythm and clock throughout the body. Environmental variables, such as light, temperature and metabolism, affects the zeitgeber, which then affects the body mechanisms (specifically the circadian rhythm). Project Z's objective is to manipulate the light variable in the zeitgeber, thus manipulating the circadian rhythm.

## **II. Technical and Development**

### **2.1 Technical Concept of Project Z**

The technical concept of Project Z was for the smartphone, Android and iOS, to communicate with the Arduino (specifically the HM-10 model). From there, the schedule the smartphone has sent to the Arduino would follow the alarm time, controlling the window curtains or blinds. This way, the sunlighting of the room is controlled by the mobile application, easily adjusting the user's circadian rhythm.

## Communication Physical Chart

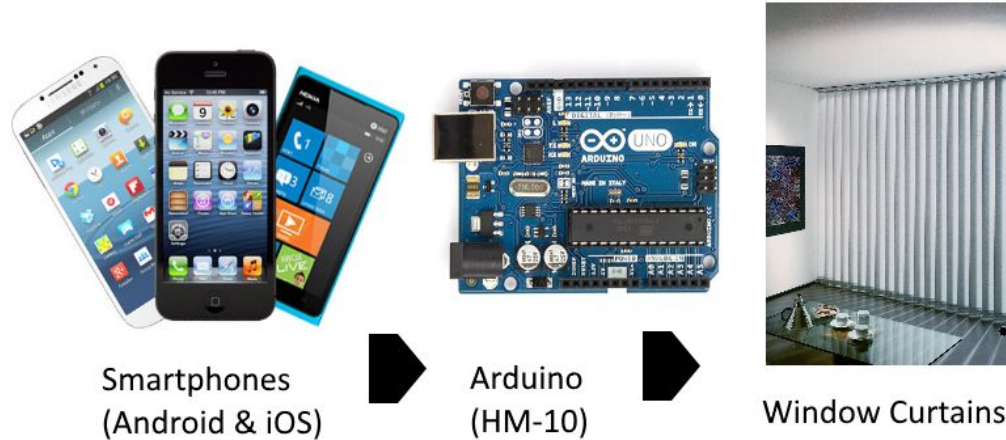
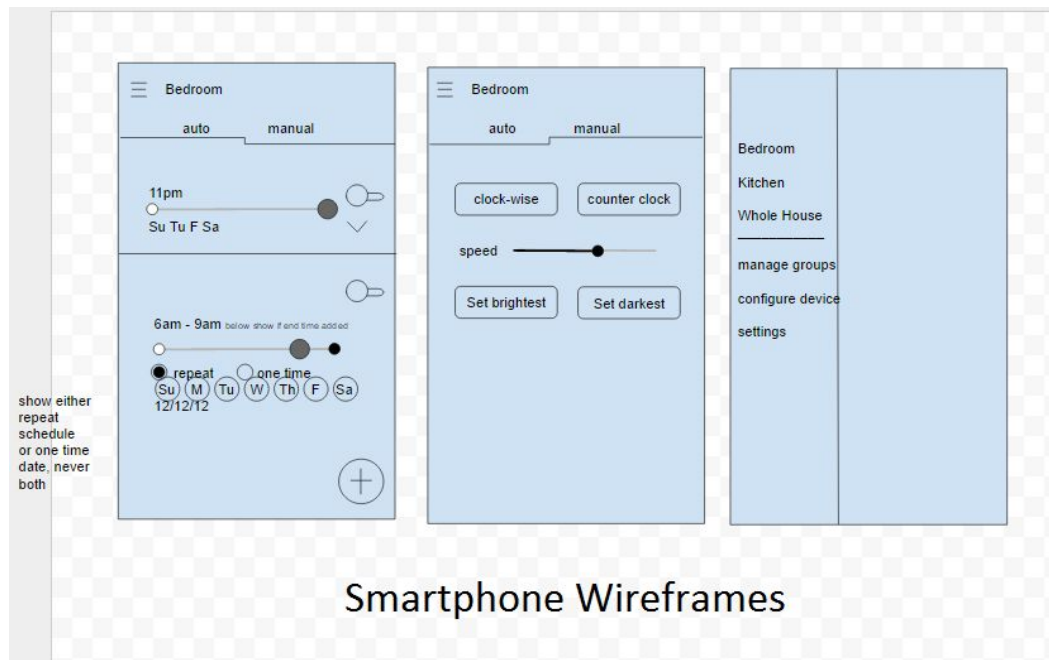


Figure 2.1.0 (Communication Physical Chart): The basic and physical concept of smartphones (Android and iOS) are to interact with the Arduino. The Arduino later on manipulates the window curtains (must need special gears to functionalize) accordingly to the set schedule by the smartphone.

On the smartphone aspects, the mobile application holds the potential to perform many functions and features. The concept of the mobile application was to be user friendly and interactive by using a tab swipe and navigation bar format.



## Smartphone Wireframes

Figure 2.1.1 (Smartphone Wireframes): The intended development of the mobile applications was to have a sliding tab manner. Each tab holds various buttons and settings to control the window curtains or blinds in various different manners.

In magnification, the Arduino controls the window curtains or blinds through certain gears. The main gear is the one controlled (and interlocked) by the Arduino, which revolves accordingly by the implemented schedule



from the smartphone. The other gears (which are meant to control the curtains) are adjustable, depending on the curtain knobs.

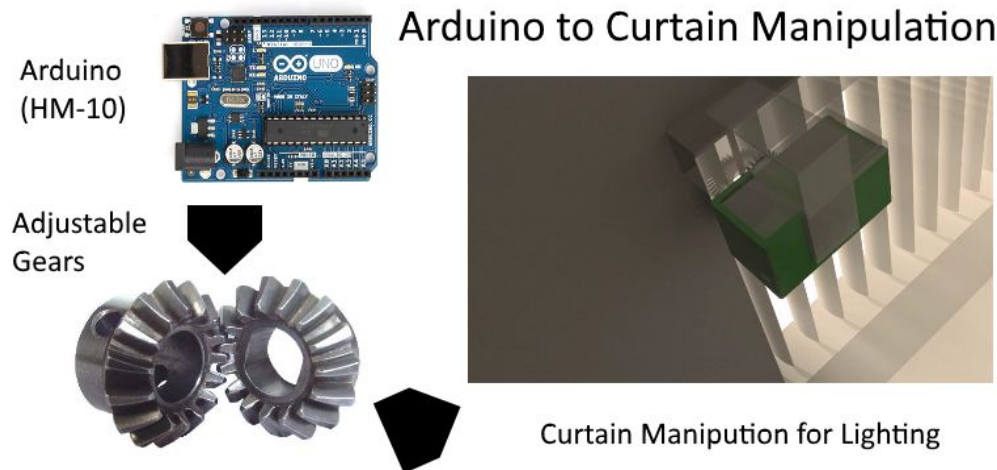


Figure 2.1.2 (Arduino to Curtain Manipulation): The Arduino follows the schedule and controls the gears between the Arduino and the curtains, which allows the gear interlocking to control the light.

Likewise on the architecture layers and schemes, there was the mentioned mobile application, but also an intended user recognition technology.

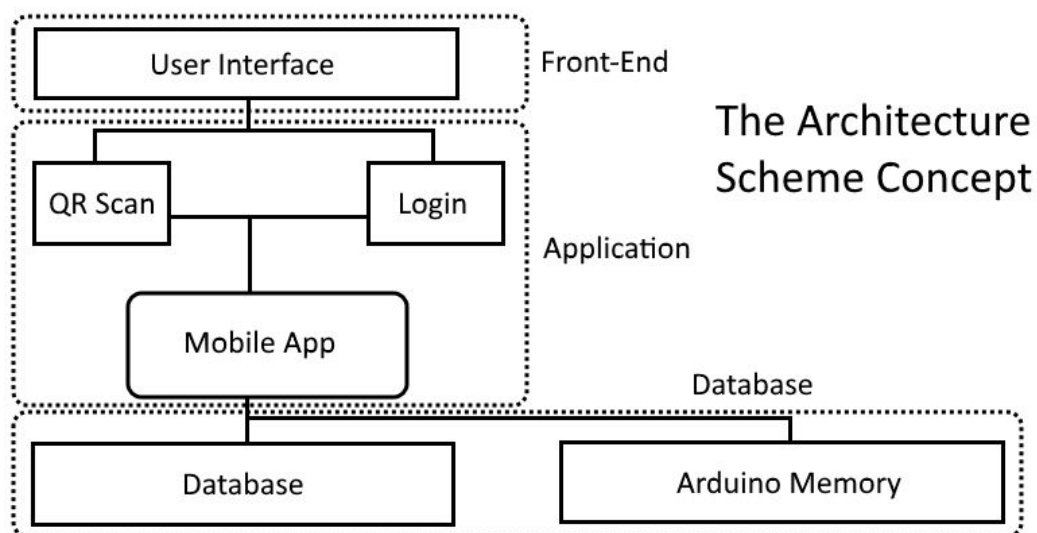


Figure 2.1.3 (The Architecture Scheme Concept):

## 2.2 iOS Application Development

The iOS application is the main pivotal mobile version of Project Z and serves as the controller of the Arduino and hardware. The technical objective of the iOS mobile application is to control the hardware components of Project Z.

The iOS mobile application was developed using the Xcode 7.3.1 integrated development environment, developed and distributed by the Apple Inc.. The programming language used for both front-end and back-end was

Swift version 2.2, a programming language developed and distributed by the Apple Inc. as well. The design layout and the Bluetooth library was provided by the Xcode development environment.

### 2.3 Android Application Development

Like the iOS application, Project Z was developed onto the Android operating system as a mobile application. The same technical concepts and objectives for the Android development can be applied to the iOS development. Like the iOS version of the mobile application, the Android mobile application was intended to serve as a controller for the Arduino and hardware as well. The Android application of Project Z currently runs on Android Platform Version 4.4, with minimum API of 19, otherwise known as KitKat.

The Android application was developed using the Android Studio integrated development environment, developed and distributed by the Google Inc.. The programming languages used for the application development are Java, used for both front-end and back-end development, and XML, used for front-end and screen layout. Furthermore, native and outside libraries were used in the development. The outside library used was from the Google Design library, used to aid developing the user interface of the application. Many of the native Android libraries, provided by Android Studio, were used to develop the Bluetooth connectivity and the other necessary functionalities, such as activities and widgets.

On a side note of the Android development, the Bluetooth connectivity library had various updates, thus many of the functions we deprecated at the process. The library used for the Bluetooth connectivity was the Bluetooth Low Energy (BLE) connection, an efficient way to connect to the Arduino. Even so, the Bluetooth Low Energy is still provided by the native Android library.

### 2.4 Arduino HM-10 Development

Without jailbreaking, iOS devices only support registered Bluetooth devices, unlike the HC-05 we were using earlier. However, they support any Bluetooth LE devices. On Android, the API for Bluetooth LE is completely different from that for regular Bluetooth [2]. All the Arduino sees, however, is a serial port, essentially a simple character stream [1]. This is in contrast to how the ESP8266 must have multiplexing enabled in order to do TCP listening, and prepends a connection number to each packet sent.

### 2.5 Arduino ESP8266 Development

At first, I thought I had to remove the CPU from the Arduino Uno board, so I could use it as a USB serial board to communicate directly with the ESP8266, because the default baud rate of 115200 baud would be too high for software serial... but it actually worked fine. I still changed the baud rate to 9600 baud for better reliability.

Relevant AT commands [3]:

- AT+UART\_DEF=9600,8,1,0,0
  - Set baud rate to 9600, 8 bits data, 1 stop bit, no parity or flow control.
- AT+RESTORE
  - Had to blindly send this when I enabled flow control with a command like the above, but I was unable to get non-corrupted output with any serial communication program. But sending it somehow worked.
- AT+CIPMUX=1
- AT+CIPSERVER=1,1001
  - The firmware of the ESP8266 requires enabling TCP connection multiplexing to serial in order to listen on TCP. Then I was able to send TCP data with netcat or a similar program, and got output like the following:
  - +IPD,0,6:hello
  - Where the first number is the incoming connection ID to the listener, and the second is the length of the message [4].

- The prototype Arduino code simply detected this in each line of the input, and ignored it if necessary. Thus the HM-10 and ESP8266 could be used interchangeably.

## 2.6 Servo and Stepper

Problems with continuous servo... values fluctuated, it was impossible to consistently determine which value to send to it to make it stop.

Both the servo and stepper motors we tested made the wifi chip bug out and reset. They drew too much current from USB, so a proper power supply was necessary. They made too much noise which messed with the wireless chip, so a better capacitor, which we didn't have, would be necessary.

## 2.7 Ability to cope with current draw of peripherals

What voltage did each component need? How many amps did they draw?

The motors needed 5V, the wifi chip 3.3V. The motors needed more current than the 45mA or so that the GPIO pins of the Arduino could safely provide, but GPIO would be fine for the control pins of the motors, not power.

## 2.8 Household Appliances Interaction

The household appliances that are being interacted with Project Z are primarily light controlling compartments, specifically window curtains. For example, the hardware component of Project Z is attached to the axis of the curtains, giving the position to control the drapes.

## III. Milestones

### 3.1 The Project's Ambition

Before the project was initiated, there were many objectives and ambitions that were idealized throughout the quarter. The project objectives were enlisted on a weekly basis, specifically with 'due' dates and the percentage of completion.

Project Z			Project Z			Start Date:	March	Project Z			Start Date:	March
Task	Start Date	End Date	Task	Start Date	End Date	Duration (days)		Task	Start Date	End Date	Duration (days)	
1.0 Planning	2016-03-28	2016-04-10	3.2 Hardware - integrate bluetooth card into arduino					4.2 Hardware - Convert app logic described above into logic for the MCU	2016-05-02	2016-05-13	12	
1.1 Project Proposal	2016-03-28	2016-04-03						5.0 Adding features	2016-05-16	2016-05-27	12	
1.2 Seek Team-mates	2016-04-04	2016-04-10	demonstrate bluetooth communication					5.1 Software - security				
2.0 Analysis	2016-04-12	2016-04-15						Make sure that only one person can modify the settings of a particular arduino	2016-05-16	2016-05-27	12	
2.1 Presentation	2016-04-12	2016-04-12	Move the motor at set times	2016-04-18	2016-04-29	12		5.2 Hardware - Incorporate all hardware aspects together in a device. Have it be able to hang from the blinds and operate with a rechargeable battery pack	2016-05-16	2016-05-27	12	
2.2 Specification / github	2016-04-13	2016-04-15	4.0 Logic	2016-05-02	2016-05-13	12		6.0 Testing	2016-05-30	2016-06-10	12	
3.0 Design	2016-04-18	2016-04-29	4.1 Software - back end logic					6.1 Software testing	2016-05-30	2016-06-10	12	
3.1 Software - Front end UI			Send settings to arduino which will be saved until next sync with the device					6.2 Hardware testing	2016-05-30	2016-06-10	12	
iOS app using official design guidelines from Apple												
Android app using official design guidelines from Google	2016-04-18	2016-04-29	move the motor set times, independent of connection to smartphone									
3.2 Hardware - integrate bluetooth card into arduino												
demonstrate bluetooth communication			group multiple arduinos together to be manipulated with the same settings									

Figure 3.1.0 (Milestone Accomplishments): The overall accomplishments enlisted throughout the quarter by weeks.

### 3.2 Academic Quarter of Week 1 - 3

For the Android development, the basic outline of the layout was implemented as well as the user interface functionalities. Users can swipe the tabs around and pull out the navigation bar to navigate around the application. Overall, majority of the user interface components have now been implemented, however, the functionality needs to be implemented.

### 3.3 Academic Quarter of Week 4 - 6

Continuing with the Android development, buttons and widgets were laid out accordingly to the planned wireframes. Furthermore, the activities passing were working fine and there is a Bluetooth device scanning feature upon connection. The screen lists the Bluetooth devices nearby and the users can have the ability to select whether upon connection.

### 3.4 Academic Quarter of Week 7 - 9

As for the Android development, a Bluetooth connection bug occurred during production, causing copious amount of time spent on debugging and testing the software. To be precise, the bug's behavior causes the application to crash when a detected Bluetooth device is detected. Such technical issue was due to a null pointer error in one of the variables of the Bluetooth service connectivity. As a result, this causes the activity to crash due to not racing the Bluetooth device properly.

### 3.6 Unfinished Milestones

With the QR scanning as one of the lower priorities, the milestone for QR scanning wasn't delved into much. Even though there is a button to prompt QR scanning on the Android mobile application, it only show the tag of being in progress.

Another milestone that has not been tapped on was the ability to control multiple curtains around the house. Mentioned from the mobile application wireframes, users can control multiple hardware devices of Project Z around the entire house at once.

## **IV. Conclusion**

### 4.1 Project Summary

Whilst, having the basic necessities done for Project Z, it was a challenging, yet a satisfying project anyone could have asked for. We got the main function of Project Z to work, which was manipulating the window blinds and curtains by mobile commands.

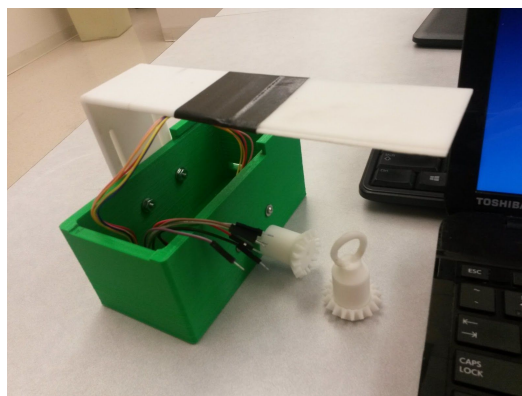


Figure 4.1.0 (Quarter Product): The Arduino of Project Z sits within the green 3D printed container with its main gear knob attached to it. Furthermore, the curtain attachment compartment sits above the container for curtain connection.

### 4.2 Accomplished Objectives

Throughout the quarter, many objectives on our priority list has been met. To say the least, the basic necessities were done to meet the goals that Project Z was striving for. The iOS mobile application can implement a schedule (with restricted functions) into the Arduino. Also, the Arduino turns the knob of the curtains accordingly to the sent schedule. This is the basic and prime key function of Project Z.

Furthermore, the Arduino and hardware has a supporting shell given by the 3D printing technology, providing a plastic cover and mobility.

#### 4.3 Group Contributions

It is fair to say that every member behind the development of Project Z did roughly equal amount of work throughout the quarter. The group dynamic was pretty efficient, considering that the weekly routine was to meet once or twice to catch up on each other's progress.

**Oscar Qiu:** The head leader of Project Z, including the technical concept of the project. He also aided the development and testing of the hardware devices and Arduino.

**Diego Wong:** Developed the main iOS mobile application for Project Z, including the debugging and testing of the application. He has also helped developed the translation language of the between the mobile applications and the Arduino.

**Pedro Coutin:** Developed the Arduino hardware and software, also wired the ESP8266 chip. He has also debugged Wi-Fi connectivity port for the mobile applications to communicate. Developed the project's main website on the GitHub hosting site.

**Alvin Yan:** Developed the main Android mobile application for Project Z, including the debugging and testing of the application. Worked on the Bluetooth library and connectivity between the Android mobile application and the Arduino.

#### 4.4 Future Outlook

If Project Z were to given more time for development for future course, the list of many implementations is countless, otherwise limitless. Recalling from the unfinished milestones earlier, many technical concepts, such as the QR scanning, would be implemented for future updates of the mobile applications.

QR scanning can provide many potentials for Project Z, specifically under the topic of hardware and software security. As mentioned, the QR scanning feature was intended to be developed, but due to time constraints and priority issues, the ambition did not come to fruition.



Figure 4.4.0 (QR Code Concept): The flow visual diagram intended earlier of development for the mobile application.

Aside from the QR scanning, the Wi-Fi functionality would be one of the prime interest to develop further in the future. Even though the QR scanning concept requires Wi-Fi connectivity, the Wi-Fi connectivity can bring other possible potentials onto the project. Mentioned before, Wi-Fi connectivity can promote security purposes between the application and the user. Thus this makes an incentive to further the project into such potential field of cyber security.

## V. References

- [1] - <https://github.com/danasf/hm10-android-arduino/blob/master/Arduino/BT4LEDTTest/BT4LEDTTestSoft.ino>
- [2] - <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>
- [3] - [https://cdn.sparkfun.com/assets/learn\\_tutorials/4/0/3/4A-ESP8266\\_\\_AT\\_Instruction\\_Set\\_\\_EN\\_v0.30.pdf](https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf)
- [4] - [http://wiki.iteadstudio.com/ESP8266\\_Serial\\_WIFI\\_Module](http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module)

<https://github.com/ocawa/pz>