TWILIGHT

Reconfigurable Self Organizing Office Lighting

Rauhul Varma
Naren Sivagnanadasan

Undergraduate Design Project - Design Review

Dept. of Electrical and Computer Engineering

University of Illinois
Urbana-Champaign

2017

Contents

1 Introduction

1.1 Objective

With the development of low cost wireless chips and microcontroller the prevalence of internet connected devices is ever increasing. This provides a great deal of potential as we can start to exercise more nuanced control of previously simple tasks (e.g. lighting). This new level of control can allow new interesting interaction models however, current "smart" devices instead of enabling these interaction models in a natural way, force extra complexity onto the user [1] in order to access the functionality. For instance, almost all connected device platforms require an app on your phone to configure the actual "smart" functionality, which is a user interaction that is really unnatural to most people. More intelligent behaviors typically require even more apps on your phone. Installation is a long process of pairing with a hub of some sort or connecting to wifi networks. The system also does not know how to run itself either and requires the (typically novice) end-user to program it. And if there system needs to be moved or a bulb needs to be changed out, then the system has to be reprogrammed. It is clear that the interaction models currently implemented in smart lighting systems are not user friendly and do not enable functionality that would improve outcomes for users. We can see from examples like Nest that when the user experience is considered as a key component of the development of a product and if the system does not rely on a user to enable the intelligent behavior, the benefits of the product are more likely to be utilized.

We present a lighting platform called Twilight that seeks to demonstrate a similar level of consideration to the user experience, allowing users to enjoy the benefits of a lighting system that thinks for them, trying to optimized the environment for productivity and comfort. Lighting is one of the subtle factors that affect our mood and productivity and building this system will bring us closer to the grand vision of our environment modifying itself to maximize comfort for its inhabitants at any given time and so it makes sense to develop systems that optimize lighting conditions [3]. However, instead of the home, Twilight targets the workplace, a space where people will spend 50% of their lives but as of yet does not have the same level of intelligent systems developed for it due to differing requirements such as scale, reconfigurability,

and management requirements. By using ideas inspired by biology in self organizing systems, we seek to demonstrate a reconfigurable lighting system for workplaces that is easy to control in an intuitive fashion, easy to maintain both from a technical development standpoint and a repairs standpoint and also be a good platform for people to experiment with controlling distributed peer to peer systems. All of this can be done for significantly less than the current lighting systems being deployed in office buildings today.

1.2 Background

The environment people live and work in has a deep impact on many things including mood, psychological health and productivity [2]. Simple changes like having the correct color temperature at different times of the day or or having the environment handle simple background tasks to reduce cognitive load may help people live happier and healthier lives [3]. However, the main blockers to having these intelligent environments widely deployed include cost and rigidity of the system (i.e. the system cannot be torn down and rebuilt easily or parts are hard to replace). Twilight aims to rectify some of these problems.

Typically there are a couple issues that can cause systems to be rigid and hard to use. Dependence on user interface flows that are not intuitive or do not leverage the habits people are used to (e.g. using a light switch to control the lights) means that whatever baked in smart functionality doesn't get used [1]. These systems when not fully utilized as designed often fall back to some sort of default functionality that is not much better than the standard light fixture (e.g. turns on as a pure white light by default) which calls into question why someone would invest in a more advanced system.

These systems are rigid as well because they typically have some sort of centralized external reliance like a hub or cloud service in order to coordinate the system. The hub is a not so great solution as it is a bottleneck for the scalability of the system. Typically there is are limit to how many nodes a hub can support (usually far lower than the number of bulbs needed to light a house) and a maximum range at which lights can connect to the hub. So that means a user must build around hubs, buying multiple ones, placing them in the foreground in order to support the system.

Even with a hub, most systems also require use of a cloud service in order to get the full set of features. This means that in order to use your expensive smart bulbs you need to have an internet connection. And if the company running the cloud service ever goes out of business your homes lighting system will not work anymore.

A potential solution to the rigidity problem of traditional systems is to use ideas from biology, namely the idea of collective behavior and self organization. By developing a federated system to govern the control of the entire of lighting system, we remove the central controller bottleneck. Such an architecture also introduces flexibility into the system, as nodes can now enter and leave the system without the system collapsing and the system can partition itself and continue to operate as well. This sort of approach has been demonstrated before [4]. Farrow et. al. present a platform built upon this core idea and show that intelligent collective behavior can be displayed without a central controller. Their work shows a smart wall system where each brick communicates with other bricks to develop a distributed touch screen and to leverage the heterogeneous nature of the bricks to allow the entire system to utilize special components of particular blocks.

These are all useful properties of a potential lighting system to have. Because there is no reliance on a central controller the system scales cleanly from 1 to many nodes. The self organizing behavior means that as long as a node is connected to the system it can inherit the properties, hence removal of any one node does not collapse the system. The system can be partitioned easily so nodes in one room need not be connected to nodes in another room. The system also can act like a normal device on a network through the use of a border router, so control of the system is direct within the network and not through some sort of cloud system.

1.3 High-Level Requirements

1.3.1 Simple Usage

Almost all existing smart lighting solutions require an app on a user's phone to configure their *smart* functionality. This is an unexpected and unnatural user interaction model for most people; the expected interaction is through a light switch.

Even more intelligent behaviors typically require even configuration through the companion app. Installation is a long, disingenuous process of pairing with a hub or connecting to a WiFi network.

Twilight will not require any significant external control for its operations, will make programming the system should be easy for application developers and most importantly not required by the end user.

1.3.2 Easy Installation

Installation and maintenance of Twilight should be trivial to the point where a student can put a fixture up with no help. As a result, Twilight should be compatible with standard ceiling tiles and be robust to individual Twilight Blocks failing.

Twilight will detect and localize all nodes in the system and automatically reconfigure if connection to one is lost.

Twilight blocks will run off standard AC wall power and will daisy chain power to one another to reduce cabling.

1.3.3 Education Tool

Twilight is sponsored ACM@UIUC, an organization that exists to help students explore the world of computing, mainly through experimentation and project building. This is one of many platforms ACM@UIUC is bringing up to provide the members opportunities to work with complex technologies in a tangible way. Therefore Twilight must be a system that lets students easily express their creativity on this platform.

2 Design

2.1 Approach

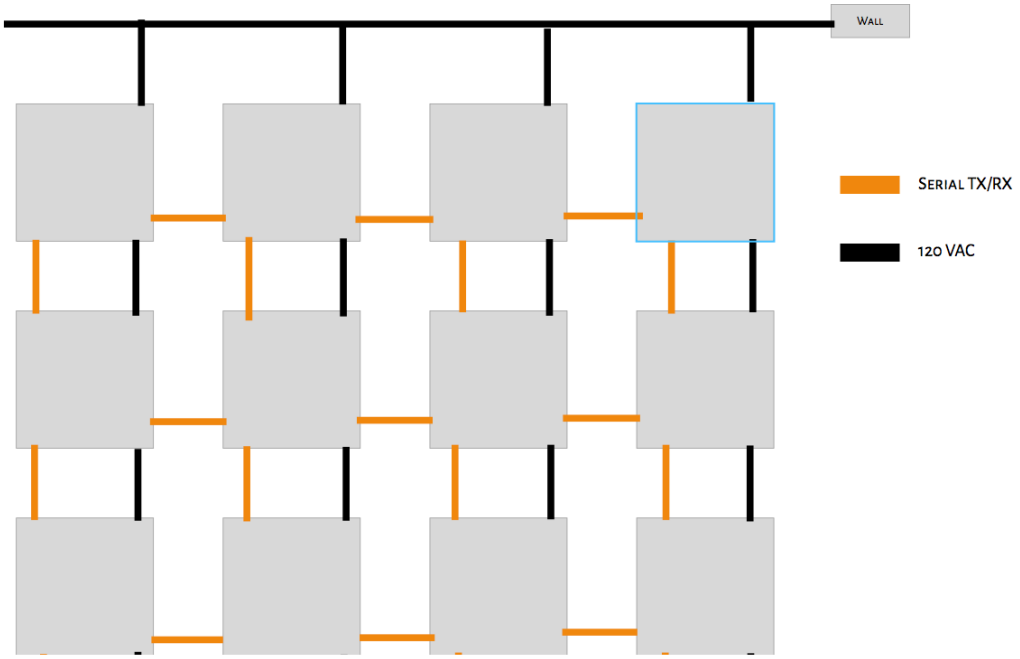

Figure 1: Proposed Default Topology

As entailed in Section 1, the use of a self organizing architecture alleviates many of the issues current intelligent lighting systems have today. From the outside Twilight looks like a collection of wooden boxes in a grid formation with cables connecting them together (see Figure 1). Each block is connected to its neighbors via a serial connection (the orange connections) and power (the black connections) is distributed via $120V_{AC}$ rails extending from the wall, into each block and out the other side, which powers the Pi and LEDs. While other configurations of the system will be supported this will be the default implementation as it most closely matches the space we are going to deploy the system in. This simple topology is easy to install because of the blocks compatibility with standard ceiling tiles, the magnetic mouing system

and the simple interconnection mechanism. However, the exterior hides the intelligent behavior below.

2.1.1 Self-Organization

When the system is powered on for the first time, it will go through a localization protocol where each block will begin to create a map of the system. This is done by the device with the lowest EEPROM ID propagating a location assignment based on itself as the origin. Other blocks will receive the localization assignment, assign itself and propagating the location assignment to its neighbors (e.g. if a block is assigned 1,2 it will assign its neighbors 0,2, 1,3, 1,1, and 2,2), signing the message with its EEPROM ID and the number of times the message has been propagated. The block will take assignment from the message with the lowest hops. If a block has been assigned and sees a location assignment message that has more hops than the assignment message that assigned its location, the block will ignore the message and not propagate it so eventually the messages will all die out finalizing the network topology.

Once the topology of system is established, the system will enter the default functionality initialized on each block. The program to run on the system is decided through consensus. Essentially, the program configuration most common in the network is run. Once the system reaches consensus, execution of the program will begin and depending on the application, messages can start to be transfered between blocks over the serial connection in the system in order to share relevant information (future commands, input data, etc.). On subsequent power cycles, the system will first begin by trying to recover the topology it discovered previously by confirming its neighbors are the same neighbors it remembers. If not, it will trigger attempt to begin the localization and consensus protocols for the system again. If a threshold is passed (33% of the network) then the entire system will re-localize. If the threshold is not passed then the unlocalized block will ask its neighbors to tell it its location. In the case of inconstant locations, the block will defer to the location closest to the origin location of the network.

2.1.2 Control

Now that the system is organized, it can start to do useful work. User control of the system is done through border routers (blocks that are both connected to the system and some sort of external interface e.g. the local network or even a light switch). This will be the gateway to control the system's high level functionality. There can more that one border router if redundancy or multiple methods of communication (e.g. a control box and a web interface) and there will be a tie breaker protocol (most recent instruction) if the system cannot reach consensus on which command to run. The system can also run without a border router, so loss of network connection will not hamper the functionality of the system. From the user's perspective, control of the system is a setting selection. It is the developers job to translate the high level setting into operations the network can carry out. For the example of a color temperature regulator, the code is as simple as a each block running a mapping from date/time to RGB value. However for more complex examples like displaying a pattern or music syncing, the developer can draw on the internal localization and consensus of the system in order to create collective behavior. New commands will be received through a border router and propagated through the system (e.g. new programs).

2.1.3 Reconfigurability

It may be the case that a user would want to take down the system and move it, or add a new block or remove a broken block. The self organizing property of Twilight makes it robust to these possibilities. If a block is removed, the system will continue onwards since there is no dependency between blocks in the system. If the block removed is the sole border router in the system, then the system will continue to run as previously configured until a border routers is re-added to the system. If the block removed is the sole block connecting two partitions of the network, each partition will continue to run as a full system after the loss of connection. If a block is added or the network is rearranged, the system will go though the localization steps described in 2.1.1 and resume operation as normal.

2.1.4 Requirements

1. System on start up, it will localize itself and begin execution of the current program
    - Tolerance: N/A
    - Verification: With 5 nodes with identical initial code connected in a cross topology, the system will localize itself and all nodes will display the same color. Then any node can be queried to get a map of the network
2. An instruction can be provided to a single node and all nodes eventually begin executing that instruction
    - Tolerance: N/A
    - Verification: With 5 nodes with identical initial code, connected in a cross topology, sending a command to an arbitrary node to change color will change the color of all nodes in the network.
3. A node can be removed and re-added and learn the program currently run by the network
    - Tolerance: N/A
    - Verification: With 5 nodes with identical code, connected in a cross topology, removing a node, changing the program (to display a different color) on the network and re-adding the removed node cause the re-added node to change to the same color as the network.
4. A node can be removed creating 2 partitions, both partitions are fully functional
    - Tolerance: N/A
    - Verification: With 5 nodes connected in a line, running identical code, removing the center node and changing the color on one node in a one partition causes the rest of the nodes in that partition to change their color to the same color.

2.2 Physical Design

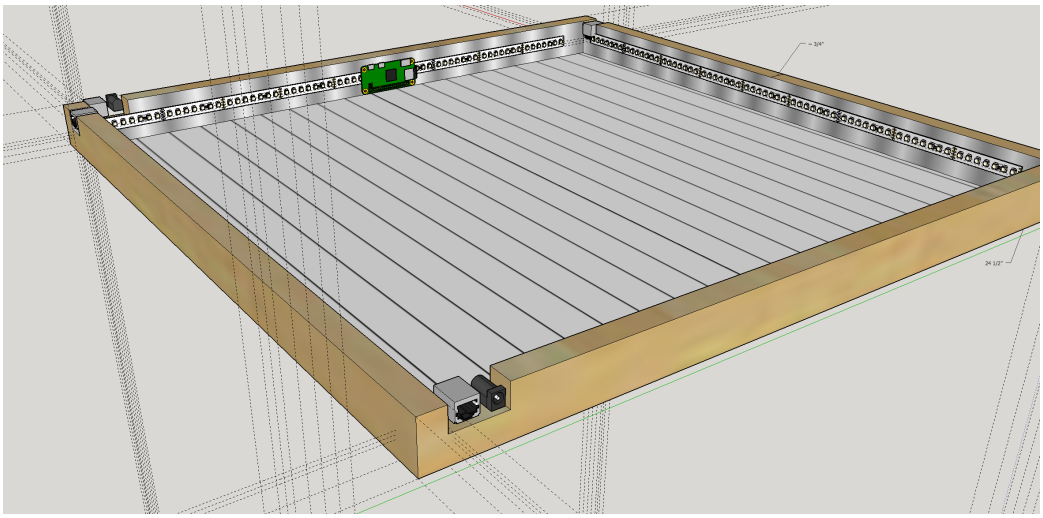

Figure 2: Top view of a block's frame

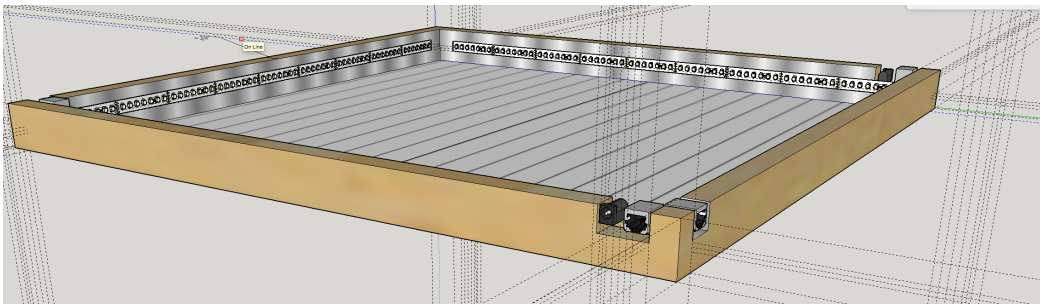

Figure 3: Barrel and RJ45 Jack inlaid into unit frame

The entire system consists of a collection of 12 blocks. Each block is a magnetically suspended wooden box with the same dimensions as a standard ceiling tile–a square with external width 24.5" and internal width 23". The internal sides are covered with with aluminum tape, improving the internal light reflection. A strip of 140 LEDs are wrapped along the these inside edges. The bottom face of the frame is covered by a canvas diffuser mounted to the inside of the frame. A Raspberry Pi Zero with the custom daughter

board is mounted to the inside of the block and controls the blocks' on-board communications, LEDs and power systems. Two barrel jacks to provide and distribute power are inlaid into the two adjacent sides. Lastly, an RJ45 jack is also inlaid into the frame for serial communication between blocks.

2.3 Twilight Block Architecture

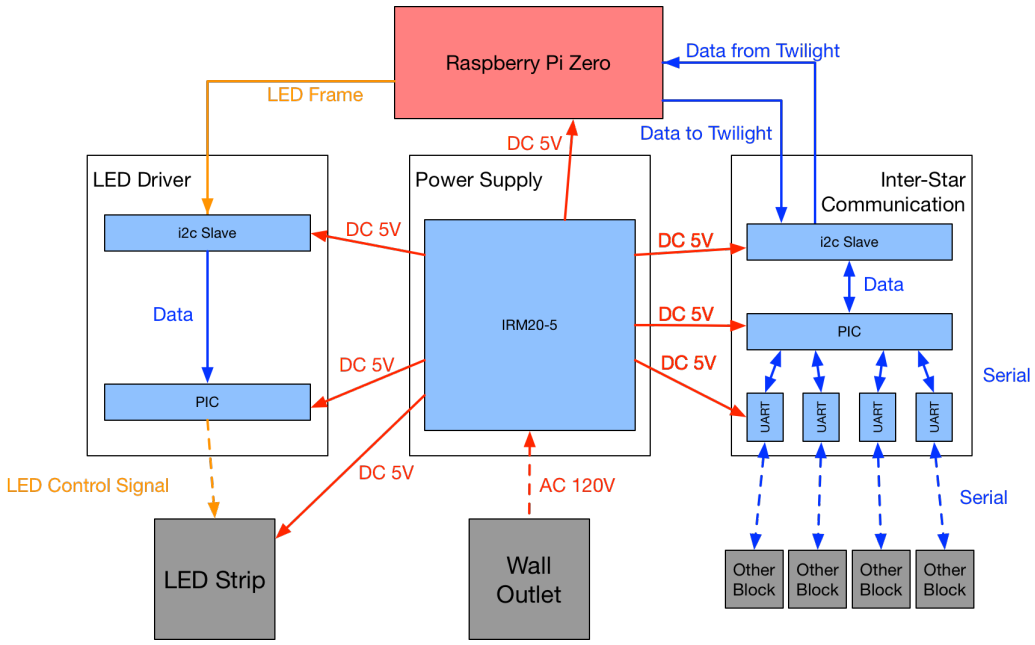

Figure 4: Electrical Block Diagram

The backbone of each block is a Raspberry Pi Zero responsible managing the software execution of the block as well as controlling the LED Driver and reacting to the messages from other blocks. Though the self organizing functionality of the project can be accomplished on a microcontroller, we choose the Pi because it provides a comfortable environment for developers (which is crucial as this is a learning platform).

We will be designing a custom daughter board for the Pi; it will include an LED driver to control an RGB LED strip within each block. We decided

to use an RGB strip because they allow for more complex and dynamic animations and smart effects than a purely warm LED strip (i.e. can change color temperature between a typical range of 2000K and 6300K). The daughter board will also include a power supply for the Pi and LEDs to run off of standard AC power and a I2C to 4-port UART breakout to allow for inter-block communication. We initially planned on using Ethernet between each block however the decided against it due to the technical complexity of creating/housing the corresponding required networking switch.

2.3.1 Risk Analysis

The Power Supply block appears to be the most difficult to implement and biggest risk to a successful completion of Twilight. The main reason for this is an overall lack of strong knowledge with power systems (beyond that in ECE 330) between the group members. Moreover, on top of the non-trivial electrical requirements, this block has significant weight restrictions that make the design more difficult. That being said, there is a substantial amount instructional material and guides on the subject online that restore confidence in completing this block.

2.4 LED Driver

LED Driver is responsible for taking frames (a description of the color of each LED in a block) from the Raspberry Pi and displaying them. This block was initially designed to work via a ping-pong buffer and a custom signal generator designed to meet the LED control signal requirements, however this was not achievable a single semester and the design was revised.

Instead, the LED Driver takes the form of a PIC on each block's daughter board. Where previously we used a hardware FSM and crystal to control signal generation and timing, we now do this process entirely in software on the PIC.

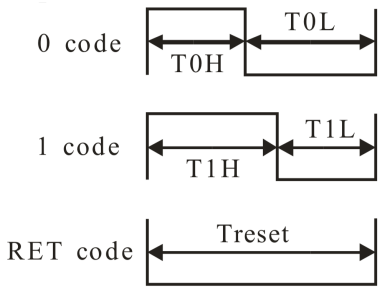

Figure 5: LED signal generation timing diagram

Operation	**High Time**	**Low Time**
Write 0	0.35 us	0.80 us
Write 1	0.70 us	0.60 us
Reset	-	>50 us

Table 1: LED signal generation timing diagram values

Figure 5 and Table 1 together specify the shape and duration of the various LED control signals. The PIC software meets these timing by setting an output GPIO pin high and performing an appropriate number of NOPs (No-operations) then setting the GPIO pin low and and again waiting an appropriate number of NOPs.

2.4.1 Requirements

1. Generate 0 LED control signal.
    - Tolerance: High time of 0.35 us ± 150 ns. Low time of 0.80 us ± 150 ns.
    - Verification: Record output signals on oscilloscope, ensure timing is met.
2. Generate 1 LED control signal.
    - Tolerance: High time of 0.7 us ± 150 ns. Low time of 0.60 us ± 150 ns.

- Verification: Record output signals on oscilloscope, ensure timing is met.

3. Generate reset LED control signal, be able to hold line low.
    - Tolerance: N/A
    - Verification: Record output signals on oscilloscope, ensure control line stays low for longer than 50 us.
4. Display 30 frames per second on a block.
    - Note: Frame rates lower than 30 fps appear disjoint and choppy, creating significant detriment to a user's experience of the system.
    - Tolerance: Variability between frames lengths of no more than ±3ms (the next frame appears between 30 to 36ms after the previous one).
    - Verification: Capture animations on a block with a high-speed camera, ensure that each frame takes between 30ms and 36ms to display.
5. The current frame is displayed until a new one is available from the Raspberry Pi.
    - Tolerance: N/A
    - Verification: Generate frames at 1 fps (starving the block) and check for flickering. No flickering should be observed.

2.5 Power Supply

Each Twilight block uses an integrated switching power supply that provides 4A at 5v. This approach was used due to a couple of reasons. Working with AC power is dangerous and using a integrate system mitigates that risk. Using a switching power supply allows for a much smaller package than a traditional transformer based power supply. This allows the power supply to achieve its weight and size requirements. However, a switching power supply is more complex than a standard transformer based one and would push the amount of work for this project out of the scope of this class. We therefore opted into using an integrated component, the Mean Well IRM-20-5.

2.5.1 Requirements

1. Convert $120V_{AC}$ from a standard AC wall outlet to $5V_{DC}$.
    - Tolerance: $5\pm$ $0.1V_{DC}$.
    - Verification: Measure output on oscilloscope.
2. Supply 4A at $5V_{DC}$ continuous current and up to 5A peak.
    - Tolerance: Continuous current must be at least 4A.
    - Verification: Load Power supply with 1.25Ω resistor and protective fuse, check that this fuse never trips.
3. Weigh no more than 1 pound.
    - Tolerance: N/A
    - Verification: Weigh component.
4. No larger than 55mm in any dimension.
    - Tolerance: N/A
    - Verification: Measure component.

2.6 Inter-block Communication

The I2C to UART block manages the inter-block communication. This is used to transfer new programs and configurations from the user and to transmit relevant data for the execution of the current program between blocks. Raspberry Pi Zero comes with a single dedicated UART handler exposed on GPIO. However each block may have to support up to 4 connections at 9600 baud. We therefore use I2C and PIC as a way to multiplex a single connection to a PI to the four serial connections it needs to support. The design is based off of an ATMega 2560, which has 4 hardware UART implements are protocol translating each incoming/outgoing UART connection into a I2C slave. We also developed a protocol to manage communication between 2 nodes such that blocks will not try to talk over each other.

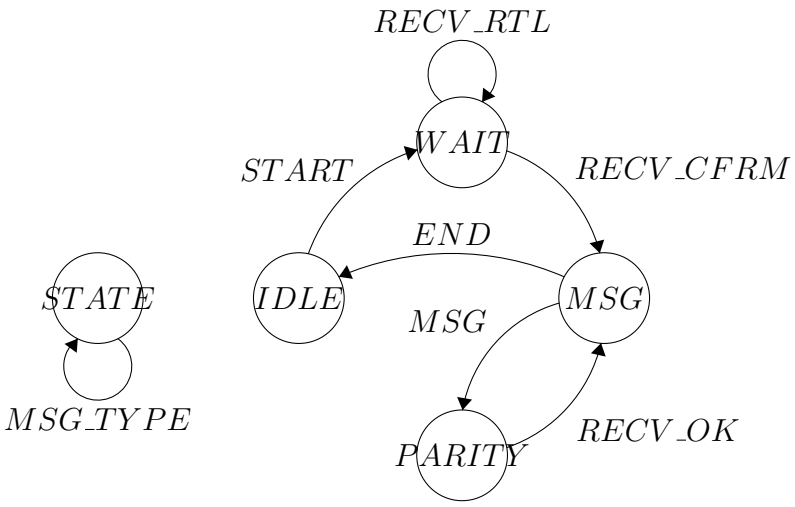

Figure 6: Initiator Protocol

The protocol starts in with the initiating block (referred to as the initiator) as shown in Figure 6. The initiator starts in the idle state, and after deciding to send a message to a node, sends a request to start a connection (termed: "Ready to Listen") and waits for a "Ready to Listen Confirmed Message" from the receiver. Once getting the confirmed message, the initiator begins transmitting the message, sending a packet, then a hash to confirm integrity. Once that is completed, the initiator returns to the idle state.

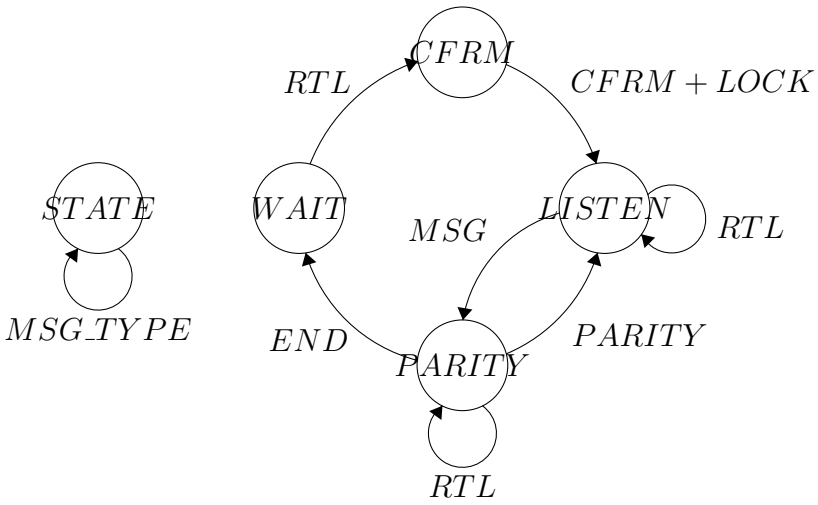

Figure 7: Receiver Protocol

For the receiver, once the "Ready to Listen?" message is received, then it either responds with confirm to start a connection or it ignores it (e.g. if there is another communication occurring). After a confirm is sent, the RX side of the connection on the responder is locked, so it will not listen to anyone else. Then it will wait for a packet, then for a hash to verify integrity. Once the initiator send the END message, the receiver returns to the wait state.

2.6.1 Message Types

Messages between blocks will have a couple main types.

1. Localization Message
    - The localization message type is used for the localization step of the system setup process. The messages contain the location assignment, the ID of the source of the assignment and its location, and the number of hops it took to get the message from the origin. Section 2.1 explains how this message type is used to create a network map.
2. Program Message

- The program message is a way to transmit new programs to blocked in the network not connected to the internet. Typically this will be the largest type of system message as its actual code to be run. Program messages initiate from the border router and are propagated through the network. If a block sees a message that it has already propagated it does not propagate it again, hence the message dies out after all blocks connected in the network receive a copy of the program message

3. Data Message

   - This message type is the primary message type used by an application developer for Twilight. Its away to share state and data between different blocks in the system. Rules about propagation are defined by the developer.

4. Ready To Listen and Confirm Messages

   - These messages are standard message primitives used to share communication state between two blocks (i.e. if a block is ready to listen to the messages from another node)

2.6.2 Requirements

1. Robust to multiple blocks attempting to initiate a message transfer. This will prevent race conditions in communications.

   - Tolerance: N/A
   - Verification: With 2 nodes trying to send messages to a 3rd node, only a single block's messages are received at any given time i.e. 2 or more nodes cannot be listened to at the same time.

2. Messages from any node looking to initiate a message transfer will eventually be transmitted, since we want to make sure that any message sent will be delivered even if it is not right away.

   - Tolerance: N/A
   - Verification: If 2 nodes try to send messages to a 3rd node, and each sends 5 messages, then all 10 messages are received in no

particular order i.e. 2 or more nodes messages will all be received eventually.

3. Communication will not dead lock due to loss of connection, so that no block will become unresponsive if communication is interrupted
    - Tolerance: N/A
    - Verification: A block can be unplugged mid transfer and the remaining block returns to listening.
4. Message data can be transfered uncorrupted. This can be done through the use of a hash of the message being sent after the message so the receiver can verify integrity.
    - Tolerance: 0 Bytes Lost
    - Verification: Transfer of a 1.5MB file occurs without data loss.
5. A localization map must be able to be developed from the connections between nodes. In order to have peer to peer communication you must be able to address nodes. This is done with the localization map.
    - Tolerance: N/A
    - Verification: Given an arbitrary network topology, each node in the network can generate a map of the network.
6. 4 Channel broadcast at 9600 Baud must be mainatable
    - Tolerance: +/- 1% of the baud rate [13]
    - Verification: A node can successfully concurrently broadcast to all perimeter nodes at 9600 baud and all messages are recieved without error

3 Power Analysis

Each block contains a Mean Well IRM-20-5 power supply with a total power budget of 4A @ 5V (20W). While the power supply can exceed the rated maximum (up to 25W) for short periods, we impose a hard limit 20W of power draw even in the worst conditions. This restriction provides a comfortable safety factor of, at worst, 1.25.

Table 2 contains the values for typical and maximum power draw for each component in a block's electrical system. The values for the "Raspberry Pi - Zero" and "ATMega2560" come from each component's respective datasheet. The values for the "Supporting Components" come from a reference design for an ATMega based daughter board. The values for the "LED Strip" were found empirically from 4 prototype blocks and represent the maximum value observed for both the typical and maximum power draw. The "Estimated Losses" come from possible DC losses through the system and represent a worst case scenario.

Component	**Voltage**	**Typ/Max Current**	**Typ/Max Power**
Raspberry Pi - Zero	5 V	80 / 120 mA	0.4 / 0.7 W
ATMega2560	5 V	10 / 14 mA	0.05 / 0.07 W
Supporting Components	5 V	30 / 40 mA	0.15 / 0.20 W
LED Strip	5 V	2.2 / 3.1 A	11.1 / 15.3 W
Estimated Losses	5 V	20 / 40 mA	0.1 / 0.2 W
Total Power			**11.8 / 16.47 W**

Table 2: Per-component and total power consumption

As seen in Table 2, in the worst conditions where ever component is drawing the maximum possible power a block will never reach the limit of 20W. The maximum draw of 16.47W ensures a safety factor of 1.52 above the industry standard safety factor of 1.25 [12].

3.1 Thermal Considerations

The wood frame of each block creates additional thermal constraints to the power system. The calculated max power draw, 16.47W, is non-trivial and

presents safety concerns regarding potential autoignition of the frame. In this section we explore the worst case scenario: the electrical system is 0% efficient, all energy is lost as heat into the wood frame, heat is only dissipated into the air external to the system.

Quantity	**Symbol**	**Value**
Wood-Air Thermal Conductivity	k_{wa}	$0.149\frac{W}{mK}$
Room Temp (at Ceiling)	T_{room}	$300K$
Power Injected	P_{in}	$16.47W$
Frame Length	l	$0.620m$
Frame Height	h	$0.038m$
Frame Depth	d	$0.019m$
Wood Autoignition Temperature	T_{auto}	$573K$
Wood Equilibrium Temperature	T_{eq}	

Table 3: Physical and thermal constants

Given this scenario we must ensure the wood equilibrium temperature falls below its autoignition temperature. Table 3 details the various physical thermal properties of the system used to determine the wood equilibrium temperature.

This system can be modeled as heat transfer through conduction (1) where the power transfered is equal to the power injected.

$$\frac{Q}{t} = \frac{kA(T_2 - T_1)}{d} \tag{1}$$

$$P_{in} = \frac{k_{wa}[4l(h+d)](T_{eq} - T_{room})}{d} \tag{2}$$

$$\begin{aligned} T_{eq} &= \frac{P_{in}d}{k_{wa}[4l(h+d)]} + T_{room} \\ &= \frac{16.47 * 0.019}{0.149 * [4 * 0.62(0.038 + 0.019)]} + 300 \\ &= 314.86\ K \end{aligned} \tag{3}$$

$$\begin{aligned} T_{auto} &\gg T_{eq} \\ 573\ K &\gg 314.86\ K \end{aligned} \tag{4}$$

As seen in equation (4) the wood equilibrium temperature in the absolute worse case scenario is far below wood's autoignition temperature. Realistically, the system will have an efficiency around 80% and additionally a large percentage of the heat will be dissipated through the diffuser. Given these calculations there is no concern that a block's frame will auto-ignite.

4 Cost Analysis

Each Twilight block contains every component listed in Table 4. While there is no set requirement for the per unit cost, keeping these costs down was an important consideration when choosing parts. When discussing various possible criterion and requirements keeping the per unit cost below $50 was mentioned, however never finalized. Regardless, as seen in the table, each block costs $35.44 in materials below this threshold. It is important to note that labor and maintenance costs are explicitly not included here as Twilight will be a student managed and maintained project.

Component	**Distributor**	**Quantity**	**Unit Cost**	**Total Cost**
Mechanical				
Wood 25 x 1.5 x 0.75	Home Depot	4	$0.99	$3.96
Diffuser	Home Depot	1	$1.40	$1.40
D42 Magnet 1/4 x 1/8	Home Depot	6	$0.34	$2.04
Electrical				
Mean Well IRM-20-5	Mouser	1	$8.55	$8.55
PSU PCB	Seeed Studio	1	$0.99	$0.99
Raspberry Pi - Zero	Adafruit	1	$5.00	$5.00
ATMega2560	Digikey	1	$8.32	$8.32
RJ-45 Connector (Female)	Amazon	4	$0.79	$3.19
IEC-C14 Connector (Male)	Amazon	2	$0.50	$1.00
Mega PCB	Seeed Studio	1	$0.99	$0.99
Total				**$35.44**

Table 4: Cost Per Light

5 Schedule

Week	Tasks
10/3/17	Initial Inter-Block network MVP Parts ordered: canvas, connectors, other hardware
10/10/17	Prototype LED controller on breadboard Partial implementation of Power Supply
10/17/17	Finish power supply Finish block network hardware
10/24/17	Send daughter board out for fabrication Finish inter-block network stack
10/31/17	Populate PCBs + verify Start application layer development
11/7/17	Address potential PCB bugs + send out new revision Begin construction of final Twilight Blocks with canvas diffuser
11/14/17	Populate revised PCB
11/21/17	Finish API Create front-end + demo app
11/28/17	Install array Prep demo
12/4/17	Demo

Table 5: Semester schedule

6 Risks to Project Completion

The major linchpin of this project is the daughter board. Communication between nodes and control of the LEDs both depend on the board working correctly. The iteration time and cost on PCBs is also significantly higher than for software, so mistakes are costly. Extra care must be taken in design verification.

The next major risk is in the communication software. Much of the unique functionality of the platform is enabled in the communication software. Due to the architecture being purely peer to peer, the complexity is much higher. The code also must be homogenious (each block runs the exact same codebase). That means that the software needs to be designed in a general way. Due to space limitations of the PIC and Pi the software also cannot be a monolithic codebase, considerations must be made for code reuse.

7 Ethics and Safety

The biggest safety issue we face is working with wall power since it is $120V_{AC}$. Since every block in the system will be working off wall AC power, verification of the power supply for the Pi is crucial. Additionally there are concerns around issues like epilepsy when dealing with fast animations. This can be addressed by limiting the frequency of the LEDs and by reviewing software deployed on the system.

There are potential environmental concerns that may arise when sourcing LEDs as some have been found to contain lead, arsenic and other dangerous substances. Proper care in sourcing RoHS compliant components must be taken.

Moreover, while this platform is designed to be used to improve peoples' mood (through the use of color temperature modulation), one could the system to decrease peoples' quality of life. For instance, many people have complained about short wave heavy white LEDs used in streetlights preventing people from sleeping.

Extended viewing of LEDs directly could also cause retina damage. The system also could potentially contribute to the growing issue of light pollution (though this is mostly an issue with outdoor lighting).

8 Appendix

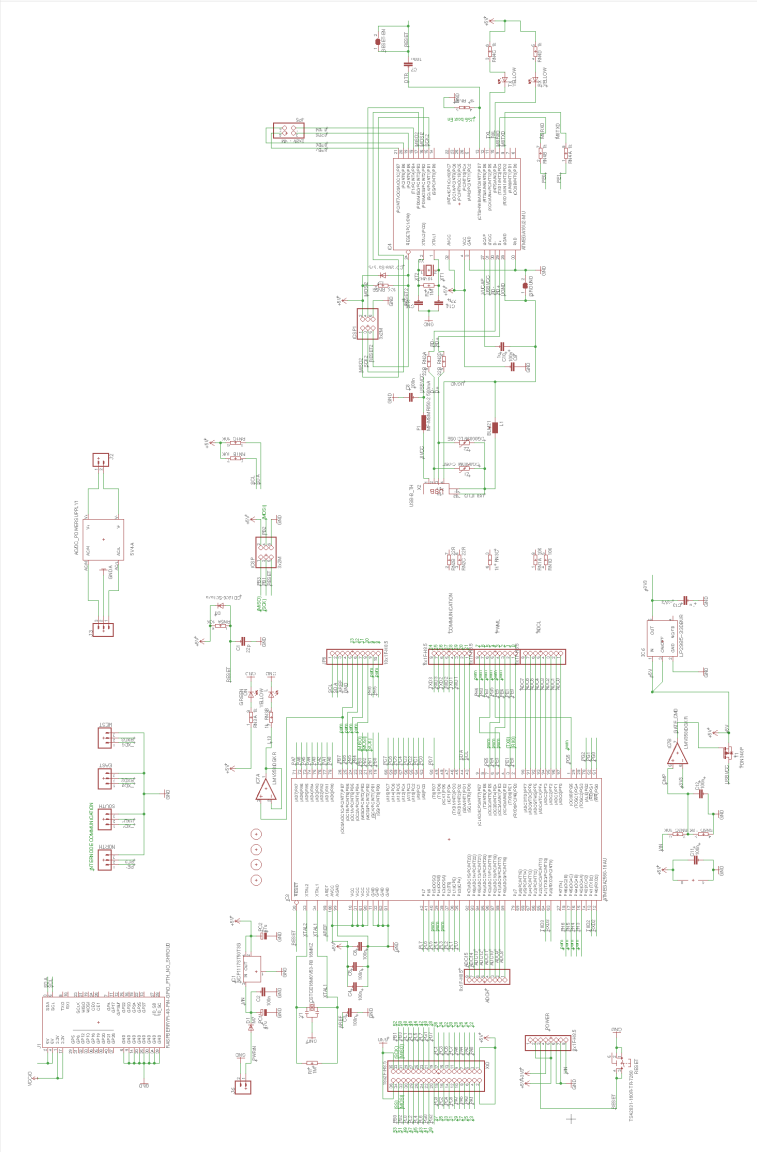

Figure 8: Twilight Block daughter board PCB schematic

References

[1] C. Rowland, "What's different about user experience design for the Internet of Things?", O'Reilly Media, 2017. [Online]. Available: https://www.oreilly.com/learning/whats-different-about-user-experience-design-for-the-internet-of-things. [Accessed: 03- Oct- 2017]

[2] W. van Bommel and G. van den Beld, "Lighting for work: a review of visual and biological effects", Lighting Research & Technology, vol. 36, no. 4, pp. 255-266, 2004.

[3] I. Knez, "Effects of indoor lighting on mood and cognition", Journal of Environmental Psychology, vol. 15, no. 1, pp. 39-51, 1995.

[4] N. Farrow, N. Sivagnanadasan and N. Correll, "Gesture based distributed user interaction system for a reconfigurable self-organizing smart wall", Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction - TEI '14, 2013.

[5] H. Ishii, H. Kanagawa, Y. Shimamura, K. Uchiyama, K. Miyagi, F. Obayashi and H. Shimoda, "Intellectual productivity under task ambient lighting", Lighting Research and Technology, 2016.

[6] R. Küller, S. Ballal, T. Laike, B. Mikellides and G. Tonello, "The impact of light and colour on psychological mood: a cross-cultural study of indoor work environments", Ergonomics, vol. 49, no. 14, pp. 1496-1507, 2006.

[7] J. Kim, J. Ko and M. Cho, "A Study of Integrated Evaluation of System Lighting and User Centered Guideline Development - Focused on the Lighting Design Method for Office Space -", Korean Institute of Interior Design Journal, vol. 23, no. 6, pp. 78-86, 2014.

[8] D. Park, Y. Lee, M. Yun, S. Song, I. Rhiu, S. Kwon and Y. An, "User centered gesture development for smart lighting", HCI Korea 2016, 2016.

[9] F. Tan, "User-in-the-loop smart lighting control system", Masters, Delft University of Technology, 2016.

[10] D. Burmeister, A. Schrader and B. Altakrouri, "Reflective Interaction Capabilities by Use of Ambient Manuals for an Ambient Light-Control",

HCI International 2016 – Posters' Extended Abstracts, pp. 409-415, 2016.

[11] A. Lucero, J. Mason, A. Wiethoff, B. Meerbeek, H. Pihlajaniemi and D. Aliakseyeu, "Rethinking our interactions with light", interactions, vol. 23, no. 6, pp. 54-59, 2016.

[12] National Electrical Code. Quincy, MA: National Fire Protection Association, 2007.

[13] "Serial Baud Rates, Bit Timing and Error Tolerance", 2017. [Online]. Available: http://www.picaxe.com/docs/baudratetolerance.pdf. [Accessed: 20- Oct- 2017].