

The Beer Grid

EEL 4914



Edgar Alastre

Colton Myers

Jonathan Chang

Ashish Naik

Table of Contents

1.0 Executive Summary	1
2.0 Project Description	3
2.1 Motivation	3
2.2 Beer Pong History	4
2.3 Beer Pong Game Description.....	4
2.3.1 Setup	5
2.3.2 Gameplay.....	6
2.4 Objectives.....	8
2.4.1 Central Controller Objectives	8
2.4.2 RGB LED Array.....	9
2.4.3 Impact Sensors	9
2.4.4 Cup Display System.....	9
2.4.5 Ball Cleaner	10
2.4.6 Power Supply.....	10
2.4.7 Smartphone/Desktop App	11
2.5 Requirement Specifications.....	11
3.0 Research Related to Project Definition	13
3.1 Existing Similar Projects	13
3.1.1 LED Dome Array	13
3.1.2 The DCPPTSS.....	14
3.1.3 Interactive Touch LED RGB Table.....	15
3.1.4 RGB LED Project With Arduino & TLC5940	16
3.1.5 Interactive LED Beer Pong Table 2.0 (BPT X5)	16
3.1.6 3D RGB LED Cube	17
3.1.7 Game Qube	17
3.2 Relevant Technologies	19
3.2.1 Pulse Width Modulation (PWM).....	19
3.2.2 Serial Peripheral Interface Bus (SPI)	19
3.2.3 Persistence of Vision Display	20
3.2.4 Universal asynchronous receiver/transmitter (UART).....	20
3.3 Component Research	20
3.3.1 MCU	20
3.3.2 LED	31
3.3.3 LED Controller	32
3.3.4 Impact Sensors	36
3.3.5 Cup Display System.....	40

3.3.6 Shift Registers.....	42
3.3.7 Ball Cleaner	43
3.3.8 Power Supply.....	45
3.3.9 Smartphone /Desktop App.....	54
3.3.9.1 Smartphone App.....	54
3.3.9.3 Database	59
3.3.10 Table Building Material	60
3.3.11 Ball Design	62
4.0 Standards	63
4.1 Safety Standards.....	63
4.1.1 ASTM D2633-13a	63
4.1.2 ASTM D2395-14	64
4.1.3 ASTM D4442-07	65
4.1.4 ASTM E564-06(2012)	66
5.0 Realistic Time Constraints.....	67
5.1 Economic and Time Constraints.....	67
5.2 Health and Safety Constraints.....	67
5.3 Manufacturing and Sustainability Constraints.....	67
5.4 User Constraints.....	67
6.0 Hardware and Software Design Details	68
6.1 Final Design Architecture, Related Schematics, and Printed Circuit Board (PCB)	68
6.2 MCU	78
6.2.1 Hardware	78
6.2.2 Software.....	80
6.3 LED Array.....	81
6.3.1 Hardware	81
6.3.2 Software.....	82
6.4 Impact Sensor Array.....	86
6.4.1 Hardware	86
6.4.2 Software.....	89
6.5 Cup Display System	90
6.6 Ball Cleaner.....	90
6.7 Power Supply	95
6.8 Smartphone App	95
6.9 Desktop App.....	97
6.10 Table Construction	99

7.0 Prototype Testing	100
7.1 Test Environment	100
7.2 Component Testing	101
7.2.1 MCU	101
7.2.2 LED Array	101
7.2.3 Impact Sensor Array	102
7.2.4 Cup Display System	102
7.2.5 Ball Cleaner	102
7.2.6 Power Supply	102
7.2.7 Smartphone App	102
7.2.8 Desktop App	103
7.2.9 Database	104
8.0 Administrative Content	105
8.1 Budget	105
8.2 Labor Division	106
8.3 Milestones	106
8.4 Group Member Information	106
9.0 Conclusion	108
Appendices	108
A. Bibliography	108
B. Copyright Permissions	113
C. Software	117

List of Figures

Figure 2.3.1	pg. 5
Figure 2.3.2	pg. 6
Figure 3.3.1	pg. 30
Figure 3.3.2	pg. 34
Figure 3.3.3	pg. 35
Figure 3.3.4	pg. 37
Figure 3.3.5	pg. 38
Figure 3.3.6	pg. 39
Figure 3.3.7	pg. 40
Figure 3.3.8	pg. 43

Figure 3.3.9	pg. 48
Figure 3.3.10	pg. 48
Figure 3.3.11	pg. 49
Figure 3.3.12	pg. 49
Figure 3.3.13	pg. 50
Figure 3.3.14	pg. 51
Figure 3.3.15	pg. 52
Figure 3.3.16	pg. 52
Figure 6.1.1	pg. 68
Figure 6.1.2	pg. 69
Figure 6.1.3	pg. 70
Figure 6.1.4	pg. 71
Figure 6.1.5	pg. 72
Figure 6.1.6	pg. 73
Figure 6.1.7	pg. 74
Figure 6.1.8	pg. 75
Figure 6.1.9	pg. 76
Figure 6.1.10	pg. 76
Figure 6.1.11	pg. 77
Figure 6.2.1	pg. 78
Figure 6.2.2	pg. 79
Figure 6.2.3	pg. 79
Figure 6.2.4	pg. 80
Figure 6.3.1	pg. 82
Figure 6.3.2	pg. 83
Figure 6.4.1	pg. 87
Figure 6.4.2	pg. 88

Figure 6.6.1	pg. 91
Figure 6.6.2	pg. 93
Figure 6.6.3	pg. 94
Figure 6.6.4	pg. 95
Figure 6.8.1	pg. 96
Figure 6.9.1	pg. 97
Figure 6.10.1	pg. 99

List of Tables

Table 3.3.1	pg. 25
Table 3.3.2	pg. 26
Table 3.3.3	pg. 27
Table 3.3.4	pg. 28
Table 3.3.5	pg. 29
Table 3.3.6	pg. 54
Table 6.9.2a	pg. 98
Table 6.9.2b	pg. 98
Table 8.1.1	pg. 105

1.0 Executive Summary

This document covers the project that group 13 (from now on referred as “the group”) is creating for their Senior Design Project. The name of the device that the group is creating is an “**interactive beer pong table**” that will be referred to hereafter as **The Beer Grid**. The group chose this project to make a team-based game which involves precision skill by throwing a ball to a target more interactive and overall more enjoyable, enthusiast and beginners alike.

The game we are deciding to enhance is called water pong, popularly known as “beer pong.” The game consists of two teams, usually composed of two persons. Each team stands on the far end of a long table. The teams compete by throwing a ping pong ball into cups that are on the other end of the table aligned in the shape of a triangle. These cups are filled with a liquid in order to prevent them from falling once the ball gets inside of cup. The ball-throwing phase is turned based and once the opposing team “scores” by landing a ping pong ball inside of the cup the other team must remove that cup and drink the liquid inside of that cup. Once the two ping pong balls are thrown by one team, then is the other’s team turn to attempt to land the ping pong balls and once one team is out of cups on their side the opposing team is declared “winner.”

The purpose of **The Beer Grid** is to make such a popular game with basic rules more interactive by adding more features to the most important part of the game which is the table. By making the table more interactive the players will be allowed to further enjoy the game.

The Beer Grid will house many features. As a basic feature the table will be able detect when the ball bounces on the table and hence, detect when the ball has landed on a cup using a basic algorithm. **The Beer Grid** will also be able to keep the score of the teams which could become a shore since these games are often played in crowded social environments.

The table of **The Beer Grid** will be regulation size which measurements 8 feet by 2 feet by 27.5 inches (L x W x H) which will allow the standard number of cups which are six on each side set in a triangular shape measuring 3.5 inches in diameter for the rim and 2.5 inches for the base and 3.75 inches high. The table will have sensors that will detect the presence of the cups along with a base to securely lock each individual cup. The aforementioned sensors will have RGB LED’s to create a ring around each cup which color will be user programmable and its primary function will be to notify the user whether the cup is placed. Optimally, the color of the aforementioned LED’s will change color if a cup is removed.

Furthermore, the entire table field of **The Beer Grid** table will have a sensor field and a RGB LED array. Both of these will serve for an aesthetic appeal and a functional aspect. The purpose of the aforementioned sensor array is to detect when a ball bounces any place on the field. This situation is called when player

performs a “bounce shot” to land their targeted cup. Another possible situation could be when the player misses their shot and the ball happens to hit the field. The “sensor field will work in conjunction with the RGB LED array. The function of the aforementioned RGB LED array would be light up on the specific area where the specific sensor in the sensor field is activated. For instance, if the player performs a “bounce shot” a sensor will be activated and thus, would light up a RGB LED in that same location or area. The behavior of the RGB LED array would all be decided by the user. As of the writing of this document, the RGB LED array will have preset behaviors such as a ripple effect that would resonate through the entire table once the ball makes contact with a sensor. The ripple will be in different shapes. Additionally, another effect would be added where the RGB LED array would light up in patterns that simulate effects such as a plasma globe. Finally, another preset effect will be added where the RGB LEDs will react to the music being played.

The group will utilize several microcontroller units (MCU) for the completion of this project. A main ARM processor with enough power to control several slave MCUs. Most likely, a MCU will be required for the RGB LED array as well as one for the sensor field. The main MCU will most likely contain a wireless module in order to communicate with a smartphone for the players to configure **The Beer Grid** to their desire. Furthermore, the main MCU will control the scoring aspect of the game as well as other key features such as team management, scoring, etc. All information will be sent directly to the smartphone.

Additionally, the table will have a feature that has always been neglected which is hygiene. Often, when the game is played, the ping pong falls to the ground and often collecting foreign particles such as dust and germs. The team solves this problem by adding a built-in ball cleaner that will not only dry the ball using pressurized air, but also free ball from all types of particles and a UV light that will potentially kill any germs thus making the game more hygienic while adding a better gaming experience since the ball will be perfectly dry during each turn.

Finally, **The Beer Grid** will include a smartphone app that as mentioned in previous paragraphs, will control the behavior of the table. The smartphone app will be a critical feature as it will provide the players control of **The Beer Grid**. The smartphone app will allow players to set up a profile that will track progress of the players and provide a rank based on their performance. It will also allow the players to configure the behavior of the RGB LED array, set game mode and keep track of the game’s score.

2.0 Project Description

2.1 Motivation

The members of the group have in occasion gone to social gatherings such as house parties or tailgate as responsible, legal-aged adults. These gatherings involve the enjoyment and company of others as well the consumption of alcohol. While at parties the most common activities would include socializing, dancing, and the enjoyment of music there are also games involved which sometimes include card or board games, but in most cases the game is Beer Pong.

Beer pong is one of the most popular games in college parties and tailgating gatherings. It is part of the college culture and has been practiced for many years. As there are official rules, there are also very different play styles in different regions as it is a “cultural” activity.

Given the nature of Beer Pong, being a house-party activity. The equipment is often forgotten about. The team realized that usually the table is not looked upon, often hampering the experience of the game. Additionally, since house parties are not exactly events where organization is had. Often, when Beer Pong is played since there are so many distractions, teams sometimes lose track of which team has their turn, also there are queues problems as to which set of teams are next to play. Overall, Beer Pong can often become an inefficient experience given by the nature of a house-party organization.

The team also seeks to modernize the game by making it more interactive by integrating a system that keeps tracks of teams, players and their score. In these social gatherings, none of this information is collected as there is no technology available to store it. The creation of the app will address all of these issues by not only giving control of the players of how they want to customize the table, but by also giving the power of the player to show their performance to other team and players and to provide a more fair matchmaking during these social gatherings.

As aforementioned, the appeal of the table has been something forgotten about by many of the biggest table manufacturers. Beer Pong tables have been standardized for many years and barely any aesthetical changes have been made, but rather, portability has been the only improvement addressed. The team seeks to address the aesthetical aspect of the table by making **The Beer Grid** more visually appealing and interactive with the RGB LED array and the feedback sensors which will provide another layer of gameplay.

Additionally, the team noticed that often when the game is played a player may miss a shot that would result in the ball falling on the floor and ending up in places that accumulate foreign particles and bacteria such as under a table, furniture and so forth. As a result, the ball would accumulate some of these particles. The most popular solution is keeping a cup of clean water to rinse off the ball. Although

clever, the team has considered this an inadequate solution and seeks to solve it by adding a more complete cleaning procedure to keep the game more hygienic.

The members of the group additionally seek to improve their team working skills and expect to utilize the experience gained from previous electrical and computer engineering courses. The members of the group look forward to gain experience in app development as well as database management as it will be required in the creation of **The Beer Grid**. The app will provide many skills required today in the industry as it will require communication with another device and will share extensive information between the two. Additionally, the creation of the table will provide significant experience as it will require a smart design in order to administrate many components including an RGB LED array and a sensor field. Moreover, other aspects such as power management is vital in the development of **The Beer Grid** as the RGB LED array will require a steady supply of power.

2.2 Beer Pong History

Beer Pong is a game that has been practiced as early as the 1950's by many college fraternities. The origin of the game is unclear and given the nature of the game it is very difficult to determine the individual who came up with such a simplistic game idea. Although, it is said that the Delta Upsilon fraternity of Bucknell University claimed that they created a game that ultimately led today's Beer Pong. Delta Upsilon's version difference was the usage of paddles instead of just throwing the ping pong ball with one's hands. The objective was still the same as the player would have to use the paddles to direct the ping pong ball into the opponent's cup. After the 1970's the variant form of the game which included paddles was dropped and the game became what is today known as Beirut which is the paddle-less version of the game. Beer Pong has been popularized enough for enthusiasts to implement technologies in the table. Prototypes by various universities and hobbyists include tables that have a built in ball washer. Additionally, tables that have lighting that respond to certain frequencies.

2.3 Beer Pong Game Description

Beer Pong is typically a house-party game played by legal-aged adults as it involves consumption of alcoholic beverages. In this game, two teams go against each other and the objective is to remove the opponents' cups off the table by launching a ping pong ball across one team's side to the opponent team's side and landing the ping pong ball in the opponent team's cups. The winner is determined by eliminating all of the opposing team's cups. The following are the rules and regulations of house play (Rules - Drinking Beirut (Beer Pong), Rules- Official House Rules | BPONG, Rules-Typical House Rules | BPONG).

2.3.1 Setup

There are specific rules for Beer Pong regarding the nature of how many people constitute a team, the sort of cups and balls permitted for play, the size of the table, and where players are allowed to stand during play. Our table will not necessarily be used strictly with these rules in mind, but should at least be able to manage official play conditions should the users decide they want it to be used that way. Customization options will allow them to adjust play to their preferences.

- A. Teams
 - a. Teams consist of two different players each
- B. Balls
 - b. Game requires at least 1 ping pong ball
 - c. Balls are considered unfit for play if they are cut cracked or dented and must be replaced with a ball fit for play
- C. Cups
 - d. A total of six cups are used to represent each team on both opposing sides of the table.
 - e. Cups are plastic and must contain a volume of 16 ounces each
 - f. The cups are each filled with between 3 to 4 ounces of a beverage to be chosen by those playing the game
 - g. The cups are arranged in the shape of a triangle with the base approximately 4 inches away from the end of the table. The following diagram shows how the cups should be arranged (Rules - Drinking Beirut (Beer Pong)):

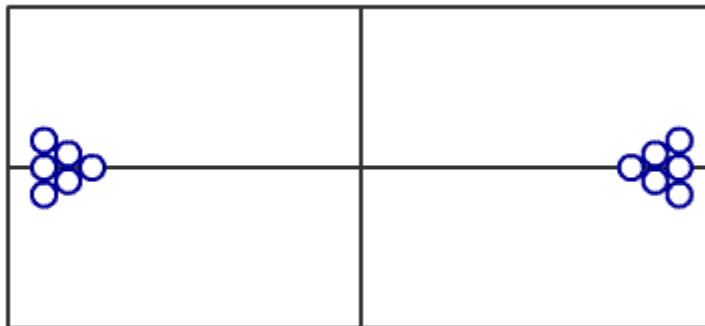


Figure 2.3.1: Diagram of the Cup Setup

- h. Cups are considered unfit for play if they are damaged in a way that inhibits their use in game and must be replaced
- B. Table
 - a. The table used must be 8' x 2' in area and about 27.5" tall

- b. There should not be any obstructions to play on the surface of the table
- c. Spilled beverages on the surface of the table should be cleaned to insure fair play.

C. Shooting Area

- a. The shooting area is bounded by the parallel edges running along the longest side of the table. Other than that, there are no boundaries on where a player can stand.
- b. Players are within legal shooting range as long as at least one foot is within the designated area. The following diagram highlights the area considered to be legal, relative to the table:

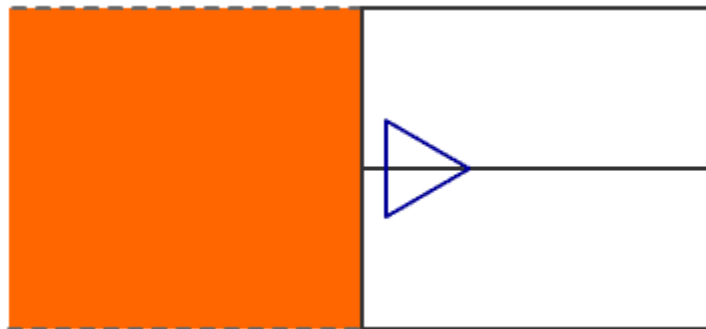


Figure 2.3.2: Diagram of legal shooting range

2.3.2 Gameplay

The following explains the official rules of Beer Pong play. They distinguish and explain the nature of “shooting”, “offense”, “defense”, “bouncing”, “elbow rule”, “Re-racking”, “Knocking Cups”, and “Rebuttal and Overtime”. Our table will be built with an attempt to keep these rules in mind to keep its design conducive to these gameplay methods. Of course, whether or not such rules are followed is up to the users to decide ultimately.

A. Shooting

- a. During each team's turn, each team shoots one ball. If a player shoots a ball and extend their arm their elbow shall not cross the table's plane. If a player makes a shot and lands an opponent's cup, then that cup is removed. If a player makes a show and misses then no cups are removed.

B. Offense

- a. The offense or "offensive team" is considered as the team that has the possession of the ball and that are in their right to propel the ball into the opposing team in attempt to land the ball in the opponents' cups. The offense team may shoot the ball in any fashion as long as they are positioned behind their side of the field and no teammate

assist the other. The offense team may only lose control of the ball if their intention was not to shoot the ball. No cup penalty is applied if the offensive team land the ball on their own cups. However, if the offensive team shoot their ball and it slips into their own cups, the offensive team must turn over the ball to the defending team effectively ending their turn.

C. Defense

- a. The defense or "defensive team" is considered as the team that attempt to legally prevent the ball to land by the offensive team by blocking the ball after first contact has occurred. The ball's first contact occurs when the ball touches the surface of the table or a cup. The only legally acceptable methods of blocking the ball include catching, swatting, hooking, and blowing. To summarize, If the offensive team attempts a bounce shot, the defensive team may attempt to catch the ball once it makes first contact with the table's surface. Additionally, if a regular shot is attempted by the offensive team and the ball happens to circulate around the cup, the defense team may attempt to legally block the shot by either blowing or using their finger to prevent the ball to enter the cup.

D. Bouncing

- a. During a player's turn, that player may bounce the ball on the table to land the ball on the opponent's cup. If a player makes a bounce shot and lands the ball then two cups are removed instead of one. The first cup removed would be the one containing the landed ball and the second ball is removed by the defending team's choice. Once the ball has made contact with the table. The defending team may block the shot once the ball has made contact with the table.

E. Elbow rule

- a. During each offensive team's turn. Players must keep their elbows and wrists behind the edge of their designated side of the table as their attempting to shoot the ball. If a player's elbow crosses their wrist or elbow the shot is deemed void and it must be re-made.

F. Re-Racking

- a. Any team can request to "re-rack," or re-arrange the cups during the beginning of their turn. This can only be done twice per game. Re-racking can only be requested when the amount of cups of any team is 6, 4, 3, or 2. Last cup can always be requested to be pulled back and centered.

G. Knocking cups

- a. If by any reason, even accidental, a cup happens to be knocked over. The aforementioned is considered "hit," or otherwise is assumed that a ball has landing on it and therefore that cup must be removed from the field.

H. Rebuttal and Overtime

- a. Given the occasion when the offensive team lands the ball on the last remaining cup of defensive team, the defensive team are allowed

a final turn called "rebuttal" turn. During the rebuttal turn, the defensive team are allowed to attempt to land the ball on all the remaining cups of the offensive team. The turns are unlimited as long as all the cups are landed or "hit" consecutively. If a rebuttal is unsuccessful then the team's final cup must be removed and the opposing team is deemed winner. However, if a rebuttal is successful, the game goes to overtime. Overtime consist of three cups being placed bad in a triangular shaped on both sides of the field. The team that landed the last cup before the rebuttal phase are awarded the first turn. No re-rack is permitted during this phase with the exception of the last cup that may be pulled back and centered. Once the overtime phase ends the team that lands the last cup is deemed winner.

2.4 Objectives

2.4.1 TBD Central Controller Objectives

Of the objectives we need to keep in mind, one of them regards our microcontroller, which our entire project will ultimately center around. The following lists the qualities our microcontroller must have with regards to each of our different design components. As long as we find one that fits these qualities, we should have everything we need to build the rest of our project around it.

- TO BE DETERMINED MCU
 - Be able to transmit and receive data from TBD MCU to all its sub-systems.
 - Obtain data from motion sensor array system
 - Obtain data from cup sensor system
 - Send data pattern to RGB LED array to alter its behavior
 - Send/receive data to exterior system such as computer/smartphone
- RGB LED Array
 - Ability to turn on each independent RGB LED in the array
 - Ability to control color combination of each RGB LED in the array
 - Responds to input given by the sensor array sub-system in real-time
 - Sends preset patterns to signal different game events
 - Winner
 - Loser
 - Score
 - Team Logo
 - Cups Remaining
 - Game start countdown
- Ball/Object Sensor Array
 - Detect contact of objects and/or ping pong on responsive surface area of table.

- Cup detection
 - Ability to detect cups located in the table
 - Can correlate the detection of cup movement as an in-game event.
- Smartphone Application
 - Used to Send/Receive data to the Bluetooth Module

2.4.2 RGB LED Array

The RGB LED Array is one of most crucial parts of the (Insert project name here) as it will be the most visible result of our project design. It will need to be able to respond to commands from the MCU made in response to input from the touch sensor array as the user desires. In a way the RGB LED Array and the Impact sensor array will work as one unit. However, due to the way the system is designed it is essentially considered as an independent unit. The goal is to give freedom to the users to fully program the RGB LED array as they desire, however the team wants the user to utilize the RGB LED array as a measure to provide visual feedback from the sensors and a way to display many aspects of the game.

Objectives:

- Responds to input from the touch sensor array subsystem
- Fully programmable giving the ability to control each individual RGB LED in the array including luminosity and color.
- Energy efficient.
- Provide an aesthetic appeal.

2.4.3 Impact Sensors

The impact sensors, otherwise known as motion or touch sensor array, will need to effectively respond to the impact of the ping-pong ball and/or any foreign object on the "main" surface of the table. This information is then conveyed to the MCU as data to process and use for signaling the RGB LED array as the user desires. The inclusion of such sensors are essential for the project because they can signal many aspects of the game such as interference from any of the teams as well as players that want perform a bounce shot.

Objectives:

- Responds to contact on "main" surface of table, including when impacted by a ping-pong ball
- Conveys information to the MCU

2.4.4 Cup Display System

The purpose of the cup display system is to have more information about the game being sent to both the MCU and the smartphone/desktop app. Each side of the table will contain 6 sensors pertaining to the 6 cups that are required to play the game. The 6 sensors will also include its own independent RGB LEDs that will

provide the user with visual information indicating that the cups are in place. The RGB LEDs will also be fully programmable to the liking of the user to provide information such as team color. The RGB LEDs and the sensors will function independently from the aforementioned RGB LED array and the motion sensor array which is why it is considered a separate sub-system. The sensors will be arranged in a triangular shape and the users will place the cups onto the available sensors as needed while still meeting the re-racking regulations of the beer pong game.

Objectives:

- Distinguish the presence of a missing cup and a placed cup.
- RGB LEDs exclusively activate to the presence of a cup.
- Respond from a removal of a cup during game session.
- Provide an aesthetic appeal.
- Fully programmable RGB LED for the users for aspects that include team color.

2.4.5 Ball Cleaner

Using a motor controlled fan and infrared sensors, we intend to construct a ball cleaner that will remove any debris and germs that have stuck themselves to the surface of the ball during play. The nature of our project is to make a table that is smarter and more attentive to the needs of users. This does not necessarily incorporate more information to the game, but does provide a convenient way of dealing with an obvious issue that occurs in the game. Given that the issue is obvious enough, it makes sense to build a feature that deals with it into our design.

Objectives:

- Effectively removes debris from surface of ping pong ball
- Turn on when ball enters tubing and off when taken from exit

2.4.6 Power Supply

To power on all the components we need several different voltages. Normally used voltages for the small components like the ones we are using range from voltages as small as 3.3 volts to 12 volts. Some components will be able to handle more voltage but normally run at these voltages.

Objectives:

- Required to power all of table's subsystems
- Required to be energy efficient
- Provide variety of voltages; 3.3, 5, 12 volts

2.4.7 Smartphone/Desktop App

The purpose of the app is to allow the users to fully customize the table to their liking. In a way, the app is the bridge between the user's aesthetic preferences and the table. From the app, the users will have the ability to start the game, create teams, and customize the table. The app will also include a robust match-making system. The app will have a database with all the players that have played in the table and keep track of their progress. To engage communication with the app and the table, the smartphone and computer will utilize the Bluetooth capability of the table.

Objectives:

- Communicates with the table via Bluetooth.
- Provide matchmaking functionality.
- Allow remote customization of RGB LEDs for features such as behavior and color control.
- Receive sensor data from MCU for game-tracking purposes.
- Robust animation creator.
- Team logo creator.
- Tournament generator.
- Keep track of players' progress.
- User Friendly.

2.5 Requirements and Specifications

The group is required to have a set of accurate requirements and specifications which every member has agreed upon since it will provide the team realistic milestones to build and test the **The Beer Grid** in an efficient manner. The following is a list of requirements and specifications for the **The Beer Grid**:

- **The Beer Grid** Requirements
 - Entire System
 - Able to play ten (10) consecutive games.
 - Able to withstand 10 ounces of spilled liquid without damaging any of the subsystems.
 - Subsystems
 - Central MCU
 - Able to handle at least ten (10) consecutive games of Beer Pong.
 - Able to independently drive 80 RGB LEDs from the RGB LED array subsystem.
 - Able to receive input from 80 Proximity sensors from the Impact sensor subsystem.

- Able to communicate with six (6) RGB LEDs and six (6) Proximity sensors from the Cup Display Subsystem.
- Able to store up to sixteen (16) different light patterns/routines for the RGB LED Array Subsystem.
- RGB LED Array
 - Responds in real-time to input from the Impact Sensors
 - Allows up to sixty-four (64) different color combinations.
 - Able to display light-patterns for up to six (6) consecutive hours.
- Impact Sensors
 - Able to detect a regulation sized ping pong ball in real-time.
 - Able to detect foreign objects in real-time.
 - Able to detect objects for up to six (6) consecutive hours.
- Cup Display System
 - Able to detect up to six different cups on its sensors
 - Allow up to 64 different colors for the RGB LEDs
 - Responds to arrangement of cups in game.
 - Able to send sensor information to Smartphone/Computer app subsystem.
- Ball Cleaner
 - Removes up to 90% of debris from regulation-sized ping pong ball
 - Able to clean up to (2) balls at once
 - Able to perform up to (100) cleaning routines consecutively
- Power Supply
 - Able to supply the (Insert Project Name Here) with 250W of power
 - Able to provide three (3) 3.3v, 5v, and 12v rails
 - Able to run continually for up to twelve (12) hours.
- Smartphone/Computer App
 - Allows remote customization of table

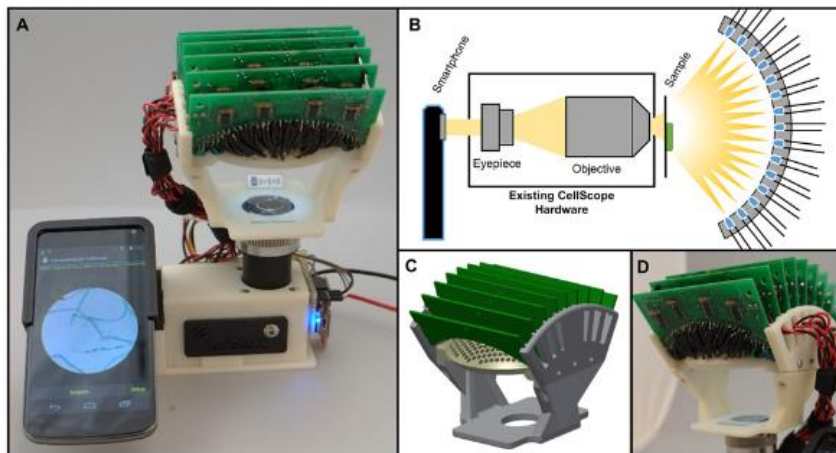
3.0 Research Related to Project Definition

3.1 Existing Similar Projects

Our project design required a great deal of research to realize the necessary components and overall feasibility of its construction. Here, we have taken the time to identify and explain several past projects with similar technologies and concepts used in their design that our project's design was influenced by. As there are over 20 projects that could be said to have some influence over how ours was made, we will focus on a limited number that were able to make the most significant contributions to our design.

3.1.1 LED Dome Array:

Of the features that our project is planned to include, the most obvious is an array of LEDs that will be able to form pre-designed patterns in addition to customized, user-input patterns. A similar system was implemented in another project involving a CellScope ("a low-cost, smartphone-based point-of-care microscope" (Phillips)) as a means of improving its existing hardware. Specifically they constructed a programmable dome of LEDs to cast light on a surface as a means of creating improved imaging with the CellScope device. This project not only involves programming an array of LEDs, but also involves working with a smartphone app to control the system, both of which are relevant to our project.



Computational CellScope. A. Device observing a sample using a Nexus 4 smartphone. B. Optical schematic of the CellScope device with our custom-made domed LED illuminator. C. CAD assembly of the dome. D. Assembled dome and control circuitry.

Copyright: © 2015 Phillips et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. (Phillips)

This project is notable for making use of nine circuit boards that were each attached to four microcontrollers. The microcontrollers used specifically were the Texas Instruments TLC5926, and each set of 4 could control up to 64 LEDs at once. All of these microcontrollers were in turn controlled by a single Arduino Micro microcontroller that could receive input commands via an ordinary Bluetooth serial link. This design not only implements an advanced array of LEDs, but also makes use of an interface with a smartphone to take commands, which is another design

goal that we are trying to implement. Given the similarities between these projects, it makes sense that we will use some of the available information on how they implemented this project to help implement ours. Specifically, this past project gives us reason to consider adopting their method of controlling the LEDs using similar, if not the same hardware. We can also consider looking into their power supply choices for our project since the one they chose was designed to “allow compatibility with a large range of power sources” (Phillips, pg. 9)

3.1.2 The DCPPTSS:

Another senior design project that was specifically built to be much the same as ours involved a limited number of LEDs that would respond to pressure input on the table from placing cups on its surface. It implemented the use of such hardware as 6 MSP430G2553 microcontrollers, 12 TLC5940 LED drivers, 46 5mW class 3b lasers, and 12 capacitive touch units, The microcontrollers give commands to the LED drivers on what patterns to take, while the laser grid and touch sensors provide input for the microcontrollers to respond to. Our project is meant to improve on theirs, so it is important to document the equipment and methods they employed in order to keep track of any similarities and differences in our design. The use of specific microcontrollers and pressure sensors in their design lends reason for us to make use of the same, but while they chose to use a laser grid for theirs, we will most likely be using an array of infrared sensors instead. Having witnessed the documentation and functioning of their past efforts, we feel that we can improve on the design by implementing the use of more LEDs with more pressure sensors on the table. Having also seen their small control panel for giving commands to the table, we will attempt to implement a smart-phone interface to operate it instead. Thus, our design will mostly be made to improve on this past design (Braun).

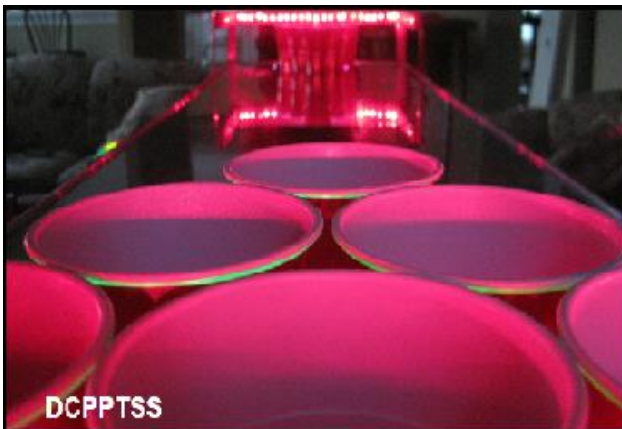


Image Taken from:
<http://www.eecs.ucf.edu/seniordesign/faq2011sp2012/g06/DCPPTSS/Home.html>

Permission Pending

3.1.3 100 Pixel RGB LED table - Interactive (touch):

An independent Do-It-Yourself engineer constructed a table that could change the color of LED lights cast on its surface in response to any touch on its surface (100Pixel RGB LED table - Interactive (touch)). The table surface was made up of 100 equally-sized pixels that could change color individually or together as part of a pattern. Our project aims to implement a similar setup, but one that can be used in conjunction with a game of Beer Pong. To facilitate the construction of our design, it would be best to use what components we can find from this project that would suit our needs.



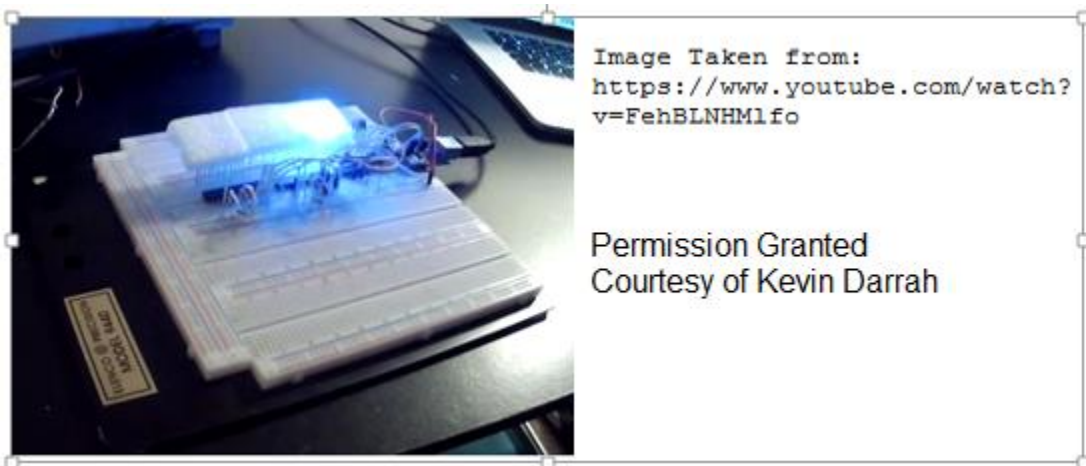
Image taken from:
<http://www.it-gecko.de/100pixel-rgb-led-tisch-interaktiv-touch.html>

Permission Pending

Of the components used to build their table, there were RGB LEDs, an ATxmega256A3 Atmel microcontroller, a frame comprised of spruce wood and plywood, a Bluetooth module, an app for customizing the table settings, and infrared touch sensors. All of these components can be worked into our design. We already intend to use LEDs in our design to illuminate our table, and using a RGB LEDs like they did will provide us with similar lighting results. They made use of an ATxmega256A3 Atmel microcontroller to control their table and process its functions, and while it did fit their purposes well, we may not necessarily use that exact model. Their LED table was shown having a frame made with spruce wood and plywood that could house individual lighting units for each pixel on the surface. The exact materials they used for the frame could easily be substituted, but their technique for building pixels into the table could be very useful to implement. At the same time, they used infrared sensors, a Bluetooth module, and an app for setting preferences in their design. We could easily incorporate infrared sensors to our design because the demonstration they gave shows us that they can still respond to the impact of a ping-pong ball on the surface of the table, and their use of Bluetooth connectivity in conjunction with a smartphone app shows us that it should not be too difficult to implement our own app with Bluetooth to connect with a table and control it remotely. Overall, a good deal of the methods used in their design could be adapted for our purposes to finish our project.

3.1.4 RGB LED Project with Arduino & TLC5940:

This particular project was meant to allow the people involved to showcase their own talents with these technologies as well as help others learn to understand it as well. It specifically features the use of LEDs, TLC5940 LED drivers, and an Arduino's microcontroller chip being used together and explains, in depth, how each of these components works. Together, they are able to program the LEDs to light up according to the patterns they design, and by making this video they could prove the value of their company with regards to programming and testing their electronic hardware. We will actually be using their efforts to learn more about how to use all of the parts they used and how to integrate them into our design.



It will actually be possible to use all of the same parts as them in our design as comparison to other possible component parts has led us to believe that the ones used in this project will suit our design idea best. Moreover, we will use the code they have provided to give us a good idea of how to program our own design, but ours will be significantly more complex as it requires a much larger amount of LEDs in its design, customization options, and the inclusion of impact sensors to communicate with the LEDs. Overall, we will just be expanding on the basic framework they've shown us to help us implement our own project design, so the information provided by this project is very valuable.

3.1.5 Interactive LED Beer Pong Table 2.0 (BPT X5):

Our search for relevant projects managed to reveal one that was very close to the design we were attempting to build. More so than any other we had encountered. In fact, it is even more complex than our design will be when finished. This project was built as an improvement over a previous version. The previous version was so popular, that this one managed to be funded through Kickstarter. It is a Beer Pong Table complete with LED displays on the surface and infrared sensors built to react to impact from the ping-pong ball. This project has a complete, detailed

instruction of how it was built, and will undoubtedly aid us in the construction of our own design.



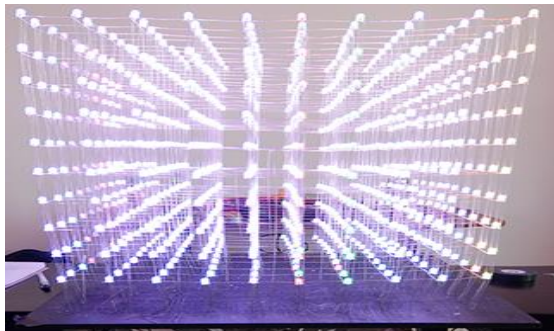
Image Taken from:
<http://www.instructables.com/id/Interactive-LED-Beer-Pong-Table-20/>

Permission Pending

There are plenty of similarities between our project and this one. In general this project implements most of the features we intend to have in our own. Both our project and theirs make use of an LED grid, ball cleaners, a Bluetooth module, and infrared sensors, but unlike their project, we will be using TLC5940 LED drivers, and the TM4C123GH6PM microcontroller from Texas Instruments. We will also not be building our grid of LEDs in quite the same way as our arrangement will make the surface of our table look like a grid of multi-colored squares, filling the surface of the table rather than rings and dots like their project had. Even so, there is plenty of information regarding the construction of the table and the arrangement of its circuitry that we can take advantage of to build our own unique project design.

3.1.6 3D RGB LED Cube:

Another Senior Design project was focused around building a 3D display using a set of LEDs that could be programmed to create patterns throughout the build. The set of 1000 LEDs were arranged in the form of a 3D cube structure with equal sides, then made to display patterns with the use of a user-friendly interface that could respond to various forms of input information. This includes, but is not limited to, audio signals and accelerometer input. The project also made use of techniques to take advantage of the concept of “persistence of vision” so that it would run more



Images Taken from:
<http://www.eecs.ucf.edu/seniorde/sign/fa2013sp2014/g15/gallery3.html>

Permission Pending

efficiently. We stand to gain a great deal of information regarding the organization and synchronization of LEDs through this project.

For the purposes of our project, we will be using their information regarding the synchronization of LEDs for building patterns on the surface of our table. We will also attempt to integrate techniques that take advantage of persistence of vision so that our table can run more efficiently. We may also examine the user interface that they built and incorporate any useful ideas when constructing the app we intend for our design to implement. Even if our design can be customized through the app that we build, it is still important that the interface is user-friendly so that it can be used by people who don't have any engineering experience. As such, if this project can provide any information to that end, we will be sure to make use of it. For the most part, however, we will focus on trying to incorporate their techniques for organizing and Synchronizing LEDs so that we can use similar techniques to manipulate our LED grid on the table.

3.1.7 Game Qube

This Senior Design project was significant not only for creating a working grid of LEDs that could be made to display patterns, but that could also be used as an interface for playing games. It could use its display system to play classic games such as "pong" and "snake" on a system of LEDs that were arranged into a 3D cube. The idea was that, while similar 3D grids of LEDs had been implemented before, this project would allow the users to play games on the 3D display, which had not been done before. Users would also be able to play these games with the use of a Bluetooth-connected controller. The ability to interact with the LED display using Bluetooth connectivity makes this project very different from others, and very valuable for our purposes.



Image Taken from:
<http://www.eecs.ucf.edu/seniordesig/fa2013sp2014/g33/Documents/SD1PprojectDocumentationGroup33.pdf>

Permission Granted
Courtesy of Stephen Monn

This project is useful for our own design because it involves a method of communicating with a system of LEDs using a Bluetooth connection. This

information is necessary for us to implement the customization feature in our project design so that the LED grid on our table can be changed through our smartphone app. In general, more data on manipulating an LED grid and making use of techniques that use “persistence of vision” displays is useful, but this is one of the only projects we could find that implements a means of interaction through Bluetooth that can affect the LED grid. It is also one of the only ones that has any information on how to format the user interface to make it user-friendly, which is just as important as our other design features since we intend for untrained users to be able to interact with the table as well.

3.2 Relevant Technologies

3.2.1 Pulse Width Modulation (PWM)

An excerpt from the book, *Pulse Width Modulation For Power Converters*, made the following statement, "One of the most widely utilized strategies for controlling the AC output of power electronic converters is the technique known as pulse width modulation (PWM), which varies the duty cycle (or mark-space ratio) of the converter switch(es) at a high switching frequency to achieve a target average low-frequency output voltage or current," (Holmes, pg.95). To put this in simpler terms, it is possible to reduce the amount of power utilized by a device by sending power to it in pulses at a lower frequency rather than constantly running power through them. Though the device may not appear to function any differently to a common user's eye, this technique allows for significant reduction in energy usage. As such, we will be attempting to implement such a technique into our design to make it as efficient as we can.

Efficiency is one of the goals of our project, and using PWM is a clever way to achieve it. The LED lights on our table don't need to run constantly to still be seen by common users. They only need to be pulsed with enough energy, at an appropriate frequency, that users can see them constantly. This allows our table to reduce its energy usage significantly. Also, we learned from the “Game Qube” project that this technique could be used for adjusting the apparent brightness of our LEDs as well (Monn, pg.11). We could easily need to make use of this information, so PWM will be a very relevant technology in our design.

3.2.2 Serial Peripheral Interface Bus (SPI)

This technology is used to allow a microcontroller to communicate quickly with nearby peripheral devices or even other microcontrollers. Our project will require a custom-designed circuit board to connect our chosen microcontroller to its peripherals, so naturally we will be making use of this in our design. It was developed by the Motorola company before becoming more common in the world of electronics and is comprised of a “simple 4-wire serial communications interface used by many microprocessor/microcontroller peripheral chips that enables the controllers and peripheral devices to communicate with each other” (SPI interface in embedded systems). The SPI also has full duplex capabilities and was

inherently designed for high-speed transfer, which relates to why it can't be very long in length or else it overheats. A simple technology, but vital for our project.

3.2.3 Persistence of Vision Display

The phrase "Persistence of Vision" specifically refers to "the phenomenon pertaining to the human eye by which an afterimage is thought to persist for approximately on twenty-fifth of a second on the retina" (Paul). Simply put, this phenomenon refers to the illusion that a shape or pattern created by high-frequency pulses of light is constant, when it is actually the result of pieces coming together at a rate too fast for the human eye to realize the difference. This concept can be used to build displays that will appear just as visible as a constant source of light, but are actually using a fraction of the energy. We will use this technique in our table design to get our LED lights to pulse at a frequency that appears constant, but actually uses significantly less power.

3.2.4 Universal Asynchronous Receiver/Transmitter

In order to test whether our programs for the microcontroller in our circuit board are working properly, we will be making use of a UART microchip. This chip manages the interface between a CPU and attached serial devices. There are specific methods by which a computer can be hooked up to a microcontroller with this chip and made to communicate with it so that output produced from programs run on the microcontroller will be sent to the computer's display. We will use this method in order to determine if our programs are working properly, and therefore this chip is vital to our project.

3.3 Component Research

3.3.1 MCU

3.3.1.1 Main Microcontroller

The main microcontroller is a very important choice for the project as is required to meet all the requirements to correctly drive every single component that will be attached to it. The main microcontroller will have the following components attached:

- LED Array
- Impact Sensor Array
- Bluetooth Module
- Cup Display System

Upon the decisions of the microcontroller, many aspects had to be studied. Aspects include available memory, number of inputs and output pins, and the clock speed. The microcontroller has to meet all the required criteria since it needs to provide the players the most real-time experience when communicating with the

LED array and Impact sensor array to provide real-time animations as well as sending real-time score information to the Bluetooth module which then will be sent to the smartphone. Therefore, the microcontroller must have enough flash memory to store all the code in order to successfully execute all of the tasks.

Memory

Based on our research, many discoveries were made. Initially, the team began researching basic projects that involved the usage of sensors and LEDs, hence, at a point, it was determined that only a basic microcontroller with a small amount of memory was required. During this point, the team determined that only a basic 8-bit AVR processor with only 32 kilobytes of memory would suffice the project capabilities. However, when the team further studied more complex projects, the team came in realization that the features in plan were disproportionate for the code required to power such features. Therefore, the team evaluated the code in accordance to the features that were set for this project. The table will require to store an initial code that will require to initiate the table in a default mode that was previously set by the last code that was sent to the table by a smartphone. In other words, the table have the ability to perform a normal operation without the requirement of being connected to the Bluetooth device, such as smartphone, tablet, computer, etc. The requirement of memory must be of enough memory to await for the table to receive a communication with a smartphone and as well to have its sensors awaiting to be activated to therefore animate the LEDs accordingly. Additionally, the table will engage in a mode that will involve direct communication with a Bluetooth device as previously mentioned which will give the users the ability to change the behavior of the table and most importantly, update scoring information of the current game session. Therefore, further increasing our memory requirement. The first project that truly established a memory limitation were the Beer Pong tables with similar functionality as the one that team is attempting to develop. Most of these tables utilized a 16-bit microcontroller of either PIC or AVR architecture with at least 128 kilobytes of memory. The tables studied had a code that controlled many aspects such as providing the user the ability to cycle through a set of animations using an infrared remote control in addition to displaying mentioned information through a small LCD display that additionally displayed information such as game score, etc. The tables also included some sort of sensor array that detected the presence of cups. Therefore, the team determined that 128 kilobytes was the minimum requirements for the table to run most the features initially planned. However, one aspect of these tables that did not fully meet our criteria was that most of these tables featured some sort of LED Array that was not RGB. One of the aspects of our code is that it will require to have information of not only black and white animations but also color. Colors immediately increase code size because more values are stored in some sort of array format come in the size of 3 colors which is completely different that running a single color that the microcontroller only interprets as on or off. RGB LEDs on the other hand require 3 times more the information to run a single LED, therefore, increasing the size of the code in such a substantial amount. Additionally, most of these tables did not involve some sort of sensor array that

detected objects real-time across the entire table in addition to a cup sensor array which in terms of sensors is the only feature that our table shared in common. These unattended features created uncertainties that would require the team to further investigate other projects that involved large RGB arrays and sensor fields. The initial projects investigated with the aforementioned criteria involved a table that met most of the criteria as its microcontroller involved driving a decently sized RGB LED Array that reacted in real-time to a sensor array that created animations and such. The memory size required for the tables increased as the number of RGB LEDs in their arrays increased as well as the sensors. The first table investigated involved 100 RGB LEDs as well as 100 infrared proximity sensors and the featured microcontroller already required 256 kilobytes of memory which is already double of what was previously pre-set. Additionally, the aforementioned project involves driving an inferior number of lights and sensors than what was required by our project so that lead to further investigation. The second table had an even higher number RGB LEDs, almost five times in numbers. However, the microcontroller powering this table involved a microcontroller of a marginal performance increase. The microcontroller had the same amount of memory, being 256 kilobytes and its only difference was the architecture which was 32 bit ARM. The only unattended features now were that these tables often had very basic functionality or that the only work required by the microcontrollers was to send inputs to the RGB LED array and receive from the sensors. All of the animation and logic processing was done by a much more powerful device such as computer or smartphone which meant that most of the code was not stored on the memory of the microcontroller itself which lead the team to uncertainty as to how much memory is required to fully have all the code with animations included in the table without the need of an external device such as a computer or smartphone. Therefore, the team investigated projects that involved a magnitude of intensity in every aspect which lead to the research of the microcontroller that powered projects such as 3D RGB LED cubes. The studied cubes involved driving a substantially higher number of RGB LED and additionally stored all of the animation information inside of the built-in memory storage of the microcontroller as well as having the ability to interact with exterior devices such as a smartphone or input controller. The microcontrollers involved in these projects were either AVR or ARM based with 256 and 512 kilobytes which is around the same required for the tables researched. Therefore, the team determined that to accomplish animations and smartphone interaction 256 kilobytes is more than enough required for all of the planned featured. However 512 kilobytes would give us more freedom but it wouldn't be a must-have requirement for our microcontroller choice.

Clock Frequency and Architecture and IDE

For the team's project. Clock frequency was not the most important aspect of the microcontroller. However, the team wanted the microcontroller to have an architecture that was easy to program for and had a vast resource library. Having a complex architecture implies having to initiate or "build" an initial platform under assembly code. The group has experience with the Texas Instruments IDE with the MSPR430 and the Arduino IDE. However, the team would like to have

additional experience with other microcontrollers and other IDEs. The frequency required is not that relevant. Since the processor will not be doing anything sound related, it does not require to have at least 44.1 megahertz which is the minimum required to produce sound. Therefore, the minimum required would be at least 32 megahertz which correlates to a processor that has the same memory requirements that we previously set on the memory section. Many of the manufactures studied included Atmel, Microchip Technology, and TI. Each of these companies flavored its own type of IDE. Upon the studies done of the previous projects, the team noticed that the most utilized microcontrollers were from Atmel due to their C/C++ compiler. However, Texas Instruments also have a very good IDE called Code Composer which also allows the user to code in C/C++. The group ultimately wanted to have a microcontroller that would allow direct programming using C/C++ as is the language the team has most knowledge and feels more comfortable with. Since all companies considered had AVR and ARM architectures available and also allowed C/C++ programming, none of these categories determined a deciding or limiting factor, instead, the architectures were evaluated in their way to save space and store memory. TI processors come in both 8-bit and 32-bit as well as AVR. 8-bit processors come with up to 384 Kbytes of memory. However, ARM processors which are the favored only come in 32-bit and 64-bit. In theory, having processor that has fewer bits allows a better usage of the memory as it takes less bits per instruction. Therefore, for our purposes, the best choice is a simple 8-bit processor. However, the memory options for all the candidates considered limit that selection.

Inputs and Outputs

The I/O or input and output requirements are also not profoundly relevant upon the selection of the team's microcontroller. The microcontroller needs to have enough input and output pins to suffice the RGB LED array, Bluetooth module, and sensor array. Although the microcontroller in essence will drive a high number of inputs and outputs it will not require to have a high number of input and output pins since the way that the team has designed the table most of the components will be connected through components that allow the utilization of fewer pins in order to control a high number of inputs and outputs. For instance, the RGB LED array and sensor array will be powered by ICs such as LED Drivers and shift registers since not a single microcontroller considered has a remote amount of input and output pins to directly control these devices. With all aspects considered, if a UART design is chosen the team requires main output pins to the RGB LED array, sensor array, and Bluetooth module. 6 pins are required for the RGB LED array and an approximate 6 for the sensor array. Pins include VCC and ground so in a sense the two main components do not consume a total of 12 pins as some of these are shared between the two. The Bluetooth, on the other hand, require 1 transmit pin and 1 receive pin along with power and ground. Therefore, an estimated 14 input and output pins are required as the only component powered by the microcontroller would be the Bluetooth module and the other two components powered by the power supply.

The ultimate decision was presented when the team compared the available microcontrollers on the market. The maker did not really come in consideration as the most important aspect of a microcontroller would be the architecture followed by the maker and not the other way around. Ultimately, the team had to decide between AVR and ARM architecture. Atmel specializes on AVR architecture but also have ARM options available. The AVR options available that Atmel produces come with memory sizes up to 512 kilobytes, up to 66 megahertz clock frequency and allow from 48 to 144 input and output pins. ARM options, on the other hand go from 128 kilobytes to 2 megabytes of memory and up to 120 megahertz operating frequencies which far exceed our requirements, ARM microcontrollers, as noted, are also manufactured by popular companies such as Texas Instruments and Atmel. Aspects also considered were that the 32-bit ARM featured Thumb and Thumb 2 Instructions sets which are known for using less storage. Thumb is for 16-bit for ARM32 processors exclusively, however, they are also known for having less functionality as a result. Thumb 2 however, not variates in length, but also has the ability to execute in both 16 and 32 bit size. ARM7 and ARM32 feature Thumb mode. Thumb 2 mode however, is only exclusive to ARM Cortex. For the purposes of the team's project, the team decided that anywhere from ARM Cortex M4 to ARM7 would suffice the project if not less since the performance speed having a higher priority in terms of performance, favoring ARM Cortex microcontrollers, as there are so many similar projects using the same architecture. Additionally, ARM 7 microcontrollers require to be initiated by using assembly code. However, the team did not want to use a microcontroller that couldn't previously get their hands on for test purposes and instead wanted to utilize microcontrollers that are packaged on an experiment board so that the team could familiarize with it during the prototype phase. The first candidate for our microcontroller utilizing ARM architecture is the Texas Instruments TM4C123GH6PM which is based on the ARM Cortex M4F architecture given by its low cost and many reasons listed above. Our second candidate, also being an ARM based microcontroller is the SAM3X8E powered by ARM Cortex M3 architecture made by Atmel which is very popular for enthusiasts and home-brewers alike. For ARM7 we have a third candidate which is the AT91SAM7S512, also made by Atmel. Our fourth and last choice involves the very popular and easy to use AVR microarchitecture, also made by Atmel: The ATmega2560.

ble 3.3.1(*Tiva™ TM4C123GH6PM Microcontroller*): The TM4C123GH6PM Features

Feature	Description
Performance	
Core	ARM Cortex-M4F processor core
Performance	80-MHz operation; 100 DMIPS performance
Flash	256 KB single-cycle Flash memory
System SRAM	32 KB single-cycle SRAM
EEPROM	2KB of EEPROM
Internal ROM	Internal ROM loaded with TivaWare™ for C Series software
Security	
Communication Interfaces	
Universal Asynchronous Receivers/Transmitter (UART)	Eight UARTs
Synchronous Serial Interface (SSI)	Four SSI modules
Inter-Integrated Circuit (I ² C)	Four I ² C modules with four transmission speeds including high-speed mode
Controller Area Network (CAN)	Two CAN 2.0 A/B controllers
Universal Serial Bus (USB)	USB 2.0 OTG/Host/Device
System Integration	
Micro Direct Memory Access (μDMA)	ARM® PrimeCell® 32-channel configurable μDMA controller
General-Purpose Timer (GPTM)	Six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks
Watchdog Timer (WDT)	Two watchdog timers
Hibernation Module (HIB)	Low-power battery-backed Hibernation module
General-Purpose Input/Output (GPIO)	Six physical GPIO blocks
Advanced Motion Control	
Pulse Width Modulator (PWM)	Two PWM modules, each with four PWM generator blocks and a control block, for a total of 16 PWM outputs.
Quadrature Encoder Interface (QEI)	Two QEI modules
Analog Support	
Analog-to-Digital Converter (ADC)	Two 12-bit ADC modules, each with a maximum sample rate of one million samples/second
Analog Comparator Controller	Two independent integrated analog comparators
Digital Comparator	16 digital comparators
JTAG and Serial Wire Debug (SWD)	One JTAG module with integrated ARM SWD
Package Information	
Package	64-pin LQFP
Operating Range (Ambient)	Industrial (-40°C to 85°C) temperature range Extended (-40°C to 105°C) temperature range

Table 3.3.2 (*SAM3X / SAM3A Series Atmel | SMART ARM-based MCU DATASHEET*): The Atmel ATSAM3X8E Features:

1. Features

- Core
 - ARM Cortex-M3 revision 2.0 running at up to 84 MHz
 - Memory Protection Unit (MPU)
 - Thumb®-2 instruction set
 - 24-bit SysTick Counter
 - Nested Vector Interrupt Controller
- Memories
 - 256 to 512 Kbytes embedded Flash, 128-bit wide access, memory accelerator, dual bank
 - 32 to 100 Kbytes embedded SRAM with dual banks
 - 16 Kbytes ROM with embedded bootloader routines (UART, USB) and IAP routines
 - Static Memory Controller (SMC): SRAM, NOR, NAND support. NFC with 4 Kbyte RAM buffer and ECC
- System
 - Embedded voltage regulator for single supply operation
 - Power-on-Reset (POR), Brown-out Detector (BOD) and Watchdog for safe reset
 - Quartz or ceramic resonator oscillators: 3 to 20 MHz main and optional low power 32.768 kHz for RTC or device clock
 - High precision 8/12 MHz factory trimmed internal RC oscillator with 4 MHz default frequency for fast device startup
 - Slow Clock Internal RC oscillator as permanent clock for device clock in low-power mode
 - One PLL for device clock and one dedicated PLL for USB 2.0 High Speed Mini Host/Device
 - Temperature Sensor
 - Up to 17 peripheral DMA (PDC) channels and 6-channel central DMA plus dedicated DMA for High-Speed USB Mini Host/Device and Ethernet MAC
- Low-power Modes
 - Sleep, Wait and Backup modes, down to 2.5 µA in Backup mode with RTC, RTT, and GPBR
- Peripherals
 - USB 2.0 Device/Mini Host: 480 Mbps, 4 Kbyte FIFO, up to 10 bidirectional Endpoints, dedicated DMA
 - Up to 4 USARTs (ISO7816, IrDA®, Flow Control, SPI, Manchester and LIN support) and one UART
 - 2 TWI (I2C compatible), up to 6 SPIs, 1 SSC (I2S), 1 HSMCI (SDIO/SD/MMC) with up to 2 slots
 - 9-channel 32-bit Timer Counter (TC) for capture, compare and PWM mode, Quadrature Decoder Logic and 2-bit Gray Up/Down Counter for Stepper Motor
 - Up to 8-channel 16-bit PWM (PWMC) with Complementary Output, Fault Input, 12-bit Dead Time Generator Counter for Motor Control
 - 32-bit low-power Real-time Timer (RTT) and low-power Real-time Clock (RTC) with calendar and alarm features
 - 256-bit General Purpose Backup Registers (GPBR)
 - 16-channel 12-bit 1 msp/s ADC with differential input mode and programmable gain stage
 - 2-channel 12-bit 1 msp/s DAC
 - Ethernet MAC 10/100 (EMAC) with dedicated DMA
 - 2 CAN Controllers with 8 Mailboxes
 - True Random Number Generator (TRNG)
 - Register Write Protection
- I/O
 - Up to 103 I/O lines with external interrupt capability (edge or level sensitivity), debouncing, glitch filtering and on-die Series Resistor Termination
 - Up to six 32-bit Parallel Input/Outputs (PIO)

Table 3.3.3 (AT91SAM ARM-based Flash MCU SAM7S512 SAM7S256 SAM7S128 SAM7S64 SAM7S321 SAM7S32 SAM7S161 SAM7S16): The Atmel AT91SAM7S512 Features:

Features

- **Incorporates the ARM7TDMI® ARM® Thumb® Processor**
 - High-performance 32-bit RISC Architecture
 - High-density 16-bit Instruction Set
 - Leader in MIPS/Watt
 - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- **Internal High-speed Flash**
 - 512 Kbytes (SAM7S512) Organized in Two Contiguous Banks of 1024 Pages of 256 Bytes (Dual Plane)
 - 256 Kbytes (SAM7S256) Organized in 1024 Pages of 256 Bytes (Single Plane)
 - 128 Kbytes (SAM7S128) Organized in 512 Pages of 256 Bytes (Single Plane)
 - 64 Kbytes (SAM7S64) Organized in 512 Pages of 128 Bytes (Single Plane)
 - 32 Kbytes (SAM7S321/32) Organized in 256 Pages of 128 Bytes (Single Plane)
 - 16 Kbytes (SAM7S161/16) Organized in 256 Pages of 64 Bytes (Single Plane)
 - Single Cycle Access at Up to 30 MHz in Worst Case Conditions
 - Prefetch Buffer Optimizing Thumb Instruction Execution at Maximum Speed
 - Page Programming Time: 6 ms, Including Page Auto-erase, Full Erase Time: 15 ms
 - 10,000 Write Cycles, 10-year Data Retention Capability, Sector Lock Capabilities, Flash Security Bit
 - Fast Flash Programming Interface for High Volume Production
- **Internal High-speed SRAM, Single-cycle Access at Maximum Speed**
 - 64 Kbytes (SAM7S512/256)
 - 32 Kbytes (SAM7S128)
 - 16 Kbytes (SAM7S64)
 - 8 Kbytes (SAM7S321/32)
 - 4 Kbytes (SAM7S161/16)
- **Memory Controller (MC)**
 - Embedded Flash Controller, Abort Status and Misalignment Detection
- **Reset Controller (RSTC)**
 - Based on Power-on Reset and Low-power Factory-calibrated Brown-out Detector
 - Provides External Reset Signal Shaping and Reset Source Status
- **Clock Generator (CKGR)**
 - Low-power RC Oscillator, 3 to 20 MHz On-chip Oscillator and one PLL
- **Power Management Controller (PMC)**
 - Software Power Optimization Capabilities, Including Slow Clock Mode (Down to 500 Hz) and Idle Mode
 - Three Programmable External Clock Signals
- **Advanced Interrupt Controller (AIC)**
 - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
 - Two (SAM7S512/256/128/64/321/161) or One (SAM7S32/16) External Interrupt Source(s) and One Fast Interrupt Source, Spurious Interrupt Protected

Table 3.3.4 (Atmel ATmega640V-1280V -1281V-2560V-2561V 8-bit Atmel Microcontroller with 16/ 32/64KB In-System Programmable Flash SUMMARY):
The Atmel ATmega2560 Features:

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 64K/128K/256KBytes of In-System Self-Programmable Flash
 - 4Kbytes EEPROM
 - 8Kbytes Internal SRAM
 - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/ 100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
 - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix acquisition
 - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four 8-bit PWM Channels
 - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
 - Output Compare Modulator
 - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
 - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
 - Master/Slave SPI Serial Interface
 - Byte Oriented 2-wire Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
 - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
 - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
 - RoHS/Fully Green
- Temperature Range:
 - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
 - Active Mode: 1MHz, 1.8V: 500µA
 - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
 - ATmega640V/ATmega1280V/ATmega1281V:
 - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega2560V/ATmega2561V:
 - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega640/ATmega1280/ATmega1281:
 - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
 - ATmega2560/ATmega2561:
 - 0 - 16MHz @ 4.5V - 5.5V

Manufacturer	Model	Architecture	Memory kB	Clock	I/O Pins
Texas Instruments	TM4C123GH6PM	ARM Cortex M4	256 kB	80 Mhz	~100
Atmel	SAM3X8E	ARM Cortex M3	512 kB	84 Mhz	103
Atmel	AT91SAM7S512	ARM7	512 kB	64 Mhz	55
Atmel	ATmega2560	AVR	256 kB	16 Mhz	100

Table 3.3.5: Microcontroller Candidates

The programming interface is also an aspect that requires research. AVR Microcontrollers were really attractive for the team as they are easy to program since many libraries are as standard as C. Additionally, there amount of open source codes available online makes much easier for the team to learn how to program with the board. Another positive aspect of choosing AVR, is that is fully supported by the Arduino programming interface which allows easy programming. ARM microcontrollers on the other hand, require a little extra work to start. This however, has changed significantly in recent years as developers has adopted the ARM platform thus making it popular and more accessible. Especially if the team chooses the TM4C123GH6PM which is fully supported by Code Composer studio, a tool that's given by the university, or another excellent tool which is Energia that is completely free and allows easy access based on the easiness of AVR IDE. If the AVR route is taken, however, the team has the option of the Arduino's own IDE with the SAM3X8E ARM option as well as the ATmega2560 AVR microcontroller.

Additionally, the team wanted a microcontroller that would be available in a popular experiment board as it would greatly help understand the microcontroller as well as allow the team to work on the table before the creation of a printed circuit board that has all of the components attached to it. Experiment board allow the team to test initial software, test the RGB LED array, or at least a smaller version of it, develop the Bluetooth aspect of our project and potentially write the majority of the code for the final board with the final microcontroller. The team has many available options such as a Raspberry Pi or the Beagle Board. However, these boards far exceed any single category as well as include features that the project doesn't require such as GPU or HDMI output. The valuable aspect of many these experiment boards is that the community support is very notable. The first candidate was the Arduino Mega powered by the ATmega2560, but after some testing the team detected that the clock frequency of the ATmega2560 could lack and hamper the performance of the project. Nonetheless, the ATmega2560 could be a good prototype board for the initial phase of the project but there's a feature that the team did not like which is that the operating voltage, being 3.3v could interfere with the other subsystems and would require some resistance compensation in order to make these subsystems work. Additionally, the price of the Arduino Mega is quite high when compared to the other experiment board

candidates. The case is the same with the Arduino Due which is powered by the Atmel SAM3X8E microcontroller.

Microcontroller and Experiment Board Choice

Finally, the team reached a consensus and decided on the microcontroller that would best suit the needs of the project. That microcontroller would be SAM3X8E from Atmel. The reason why the team chose that microcontroller is because it includes the latest, as of the making of the project, ARM architecture available at the most affordable price. Additionally, since the microcontroller is included in the Arduino Due Experiment Board, from Arduino LLC, there is a lot of official support as well as the community. Additionally the price of the Arduino Due experiment board is one of the lowest among all of the candidates. Additionally, the fact that the Arduino Due is fully supported by Arduino IDE which is a free IDE that borrows libraries from the Arduino IDE environment gives it the best of both worlds.

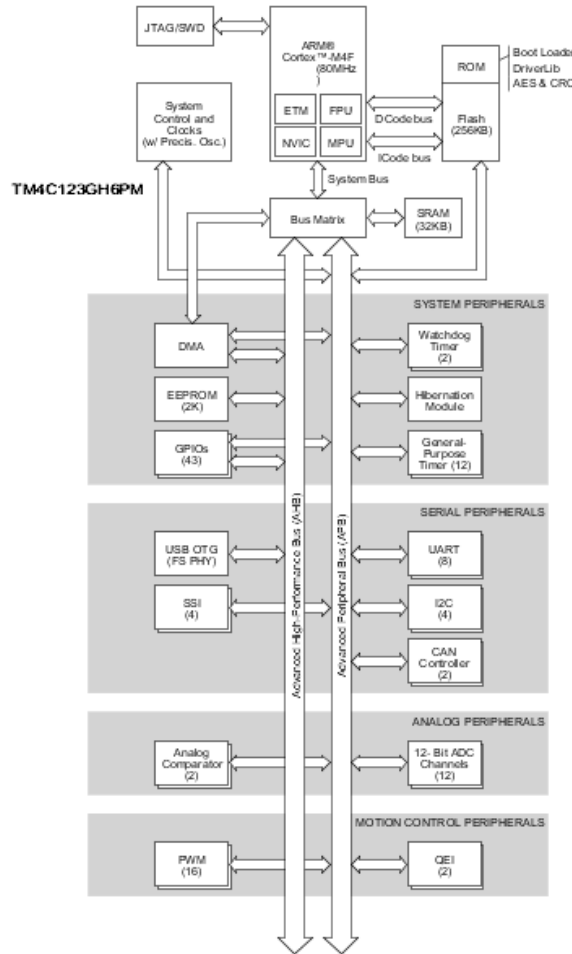


Figure 3.3.1: SAM3X8E Functional Diagram
Printed with Permission, Courtesy Atmel

3.3.2 LEDs

As our project is specifically meant to involve an array of LEDs that will allow us to display patterns on our table surface, it is necessary for us to research the different types that we can choose from. First, it is important to understand the technology behind LEDs. LED stands for “light emitting diode” and this technology specifically comes with the advantage that they don’t have a filament that burns out the way incandescent bulbs do (Harris). They are simple to install, durable, efficient, and will fit the purposes of illuminating the surface of our table much better than if we were to use small light-bulbs instead. This way, there will be little need to replace the lights on our table and even if we need to, they can be removed easily.

Single-Color LEDs

This type of LED is built by using a single diode that projects light into its design, usually with either a Red, Blue, or Green color. This is encased within a lens to magnify the effect, and if our project were to make use of this specific type of LED, we would need to use at least 3 in each pixel on our table to get them to project different colors on the surface. LEDs are cheap enough that we could still obtain the necessary amount, even if we would need to get three of them for each pixel, but programming all of them to work together would mean extra work and in the end the color they would produce might not be to our liking as designs based around the use of 3 independent LEDs can have trouble with their resolution. Given the fact that we would need to purchase more of them and the potential drawbacks they would have on our design, we will likely be making use of one of the other options at our disposal, which typically have three light-producing diodes together within a single lens instead of just one. Such LEDs require slightly more programming to operate than a single-color LEDs, but would overall be more efficient to use.

Round RGB LEDs

This type of LED is distinguished by the lens used to modify the effect of the light-producing diode. A truly rounded LED lens “can provide increased efficacy, better uniformity, and superior light quality in many applications” (DeMilo). One of the main reasons that this type of LED is considered efficient, however, is because of the fact that it has been designed to allow light to be emitted from all sides, rather than just one general direction. Since the amount of light produced is the same regardless of an LED’s shape, not using a round LED is technically inefficient. A square LED would end up losing light within its corners rather than projecting it properly, for example. Unfortunately, we don’t have a means planned to account for a round LED to be used effectively in our design. Even though it would technically be projecting more of the light it can produce, we would not have a means of directing the extra light where we wanted it to go. Simply put, we only want light shining towards the surface of the table, and don’t have any real need for it to be directed elsewhere. As such, we may not use a round LED in our design unless the other options prove less effective my comparison.

Square RGB LEDs

These LEDs are considered more common and adequate for purposes of basic lighting. They are reliable enough, but not as efficient as round LEDs, nor are they able to provide the same level of resolution. Using this type of LED would serve little other purpose than to limit the abilities of our design in for no other purpose than that we can. Since this specifically is an “RGB” LED, it is more advantageous for us to make use of it than an ordinary, single-color LED, but beyond this there are few reasons to use this type. It isn’t too cost-effective to use a square LED over any other type, so unless there are blatantly obvious reasons for us not to make use of the other options, we will likely not use this type since we have better options to choose from.

Multi Flashing LEDs

These are a particularly unique type of LED that is capable of an oscillating signal. By using an oscillating signal it is able to make the LED flash regularly and can even act as a kind of switch on the circuit that can activate and deactivate other sections of circuitry. While this is certainly a very useful function in certain situations, it has no serious bearing on our project design and would be unnecessary to include. The LEDs in our design have the specific need to respond to signal input and nothing more. As such, it we will not be using this type of LED as its extra abilities do not provide any benefit and could potentially be a hindrance to work around.

Design choice

After careful consideration of the choices we have, we have decided to make use of Round RGB LEDs in our design. They will provide superior resolution to that of single-color LEDs when attempting to create different color patterns on the surface of the table, as well as enable our design with more efficiency because of their design. Single-color LEDs are less efficient to build into our design and will have inferior resolution, Square LEDs don’t project light as efficiently, and Multi-flashing LEDs have unnecessary extra functions. Therefore, our design will make use of round LEDs instead.

3.3.3 LED Controller

The LED controller was an additional aspect that is very much needed for our project. The design problem begins with the fact that no of the microcontrollers research, and to be more precise, our candidate microcontroller does not have enough pins to drive the entire RGB LED array as the array is composed of over 250 RGB LEDs. As far as the team’s knowledge goes, there isn’t a single microcontroller with enough pins to drive over 250 RGB LEDs, and even such microcontroller existed, there could be a potential current issue where the microcontroller would not have the required power to drive all of the LEDs. The team could use shift registers to solve this engineering problem but they also have the same problem as they do not have the capability to drive the required current.

Therefore the team researched and come with the conclusion that the most adequate solution would be an LED Drive as it gives the ability to extend the number of outputs without sacrificing many input and output pins of the microcontroller and providing the RGB LEDs with sufficient current for them to light up at an adequate brightness level.

TLC5940 LED Driver with PWM

The TLC5940 from Texas Instruments is a solid choice for the team as research showed that it was the most popular amongst every project. This didn't stop the team, however, to find any other choices. This IC allows up to 16 channel outputs in PWM to control the brightness of each one of those channels and it has a built EEPROM that stores the memory of the last state of the output. It also has current sink with dot correction which is really helpful when dealing with many LEDs because some of them may be brighter than others given by the imperfection of them. Dot correction will adjust the brightness of every single LED. The way the TLC5940 Works is that the memory input works serially. This method significantly reduces the amount of I/O. One thing to keep in mind about this IC is that it does not provide current directly. Each one of the single LEDs connected have their own current source and the IC acts as a switch for each individual LED that have their current provided to their anodes.

This the team's most logical choice as this IC has the ability to supply the current for all the LEDs as well as giving the ability to control each individual LED attached to it. One thing that is to be noted is that there are 16 channels available but each RGB LED utilizes 3 channels giving you a theoretical maximum of 5 RGB LEDs per each TLC5940, another important aspect is that these ICs can be connected in series. Some pins need to be connected in parallel and then serial out from IC "A" to the serial-in from IC "B" and so forth.

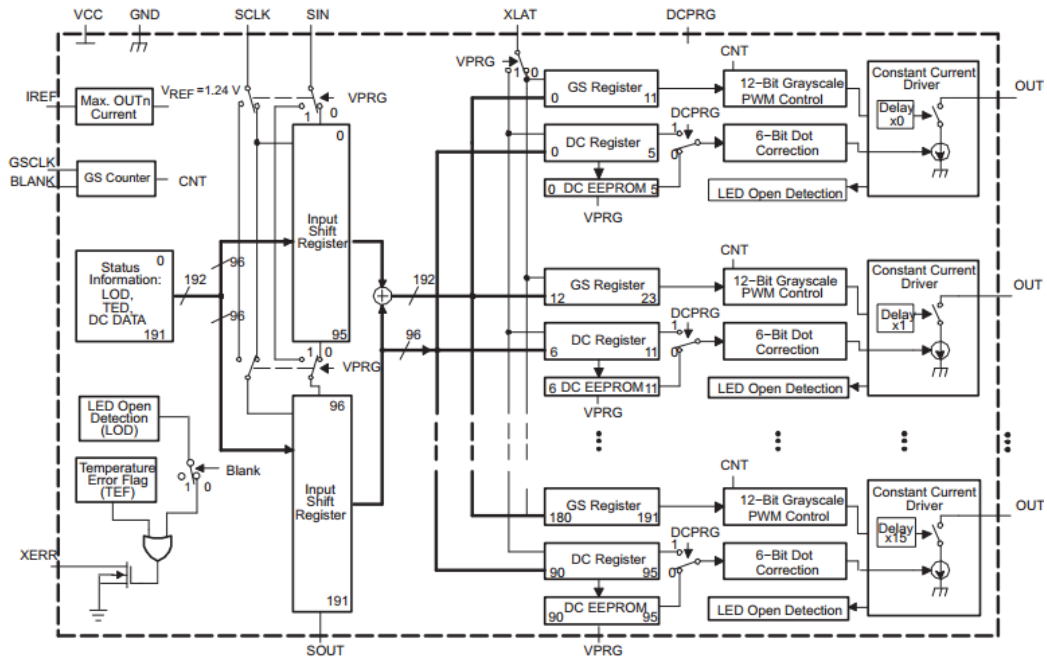


Figure 3.3.2: TLC5940 Block Diagram
 Printed with Permission, Courtesy of Texas Instruments

TLC 5940 Detailed features:

- 16 Channels
- 12 bit (4096 steps) Grayscale PWM Control
- 6-Bit dot correction storable in integrated EEPROM
- Drive capability 0 mA to 60 mA when Vcc is less than 3.6V
- Drive capability 0 mA to 120mA when Vcc is greater than 3.6 V
- Serial Data Interface
- Vcc acceptable from 3V to 5.5V
- 30Mhz Data Transfer Rate
- CMOS Level I/O
- Controlled In-Rush Current
- LED Power Supply Voltage up to 17 V

A6282 LED Driver

The A6282 from Allegro is another fine choice for the team as a LED driver. However, this option lacks PWM which would affect the brightness of the LEDs attached to the IC marginally. Besides the lack of PWM the A6282 is very comparable to the TLC5940 from Texas Instruments. Both offering up to 16 constant current outputs. The A6282 works with input shift register with data latches. Another aspect that is to be noted is that the A6282 also lacks the dot correction. This implies further testing of the RGB LEDs for the team since due to the imperfection of the LEDs one could be brighter than the others thus making the project somewhat unappealing. Another negative aspect during the research was that Allegro is no longer supporting this line, thus, making finding community codes or tutorials on operating this IC a bit cumbersome. However, the price of this IC is significantly lower than the TLC 5940 which would return in some savings on the creation of this project since the team will purchase a significant amount of these ICs.

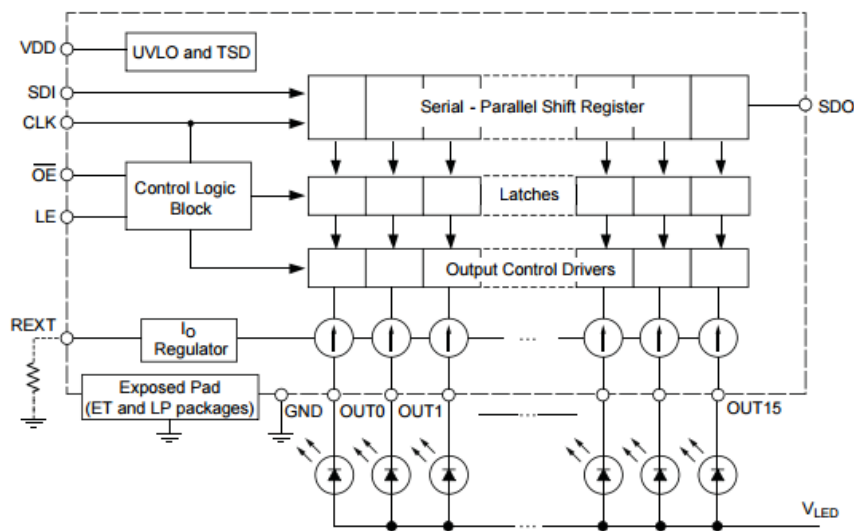


Figure 3.3.3: A6282 Block Diagram
Printed with Permission, Courtesy of Allegro.

A6282 Detailed features:

- 16 Constant-current outputs, up to 50 mA each
- LED Output voltage up to 12 V
- 3.0 to 5.5 V logic supply range
- Schmitt trigger inputs for improved noise immunity
- Power-On Reset (POR), all register bits = 0
- Low-power CMOS logic and latches
- High data input rate: 30 Mhz

- Output current accuracy: between channels $<\pm 3\%$ and between ICs $\pm 7\%$, over the full operating temperature range
- Interval UVLO and thermal shutdown (TSD) circuitry

LED Driver Decision

For the team the choice was very clear. The Texas Instruments TLC5940 would give tremendous benefits for our project. The latter choice although being more economic as an individual part, would require the team to purchase additional ICs to add the PWM functionality and every other feature that the Allegro IC lacks. Additionally, the amount of documentation and projects involving the TLC5940 is so vast that the team could find a substantial amount of useful information to become more acquainted with the TLC 5940.

3.3.4 Impact Sensors

Another relevant component required for the creation of the project is the choice of the impact sensors or otherwise sensors that would detect motion or force applied to the table. For this particular feature the team spent a significant amount of time as they were in search of the most efficient way of achieving the previously set objectives.

The sensors had to meet the following criteria:

- Ability to detect a ping pong ball as it bounces in the field.
- Ability to detect a cup being removed.
- Detection must be real-time and accurate.
- Must not be intrusive for the table.

Microsoft Kinect

The team proposed the idea of having the Kinect as a single module that would identify the cups and ping pong balls bouncing through the field. The Kinect would be placed on a ceiling or high position facing down completely capturing the table. Since the Microsoft Kinect has the ability to detect the distance of objects and all the different sensors on its disposal it could solve the objectives of the cup detection and ball bounce. The cup detection could be resolved with some sort of template matching utilizing the color of the cups as template or even a picture of the cup from an upwards field of view. In regards of the ball detection, some programming logic could determine that if any object reaches a certain distance then it could register as a ball bounce. The Kinect is relatively easy platform to program for as it has an API and another reason why is considered is because one of the members of the team already has a Kinect in their possession. The Kinect however, has some negatives as it has been proven to be often unresponsive as the purpose of the sensors are not fully designed for the project's usage. Additionally, the positioning of the Kinect could potentially hinder the design of the

table making it more complicated. Another issue is that the Kinect could potentially require to have optimal lighting conditions in order for it to function correctly as well as having a limited framerate that could not meet the real-time or otherwise snappy requirements.

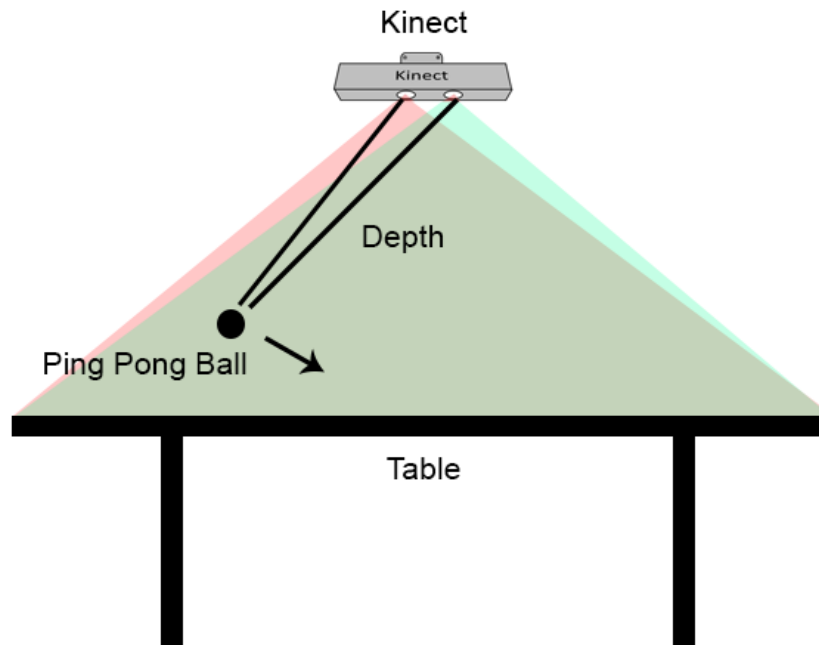


Figure 3.3.4: Kinect Application to Project Diagram

TE Connectivity Measurement Specialties 1005939-1 Vibration Sensor

The second option would involve a vibration sensor, or more specifically, a cantilever-beam accelerometer. In the team's case, the Measurements Specialties 1005939-1. In any case, any vibration sensor with comparable specifications. The idea behind the implementation of this type of sensor would involve having an array of these vibration sensors in such a way that they would detect the vibration of when the ping pong ball hits the main area of the table. The way these sensors often work is by having a mass at the end of the beam that if placed horizontally, could detect impulsive vibration that generates a frequency that is transferred as a charge or voltage output that can be sensed by the microcontroller. These types of sensors are used in many fields where vibration needs to be sensed to alert an event such as anti-theft devices, tamper detections, vehicle motor sensor, etc. The sensor would require to be strategically placed in order to register the vibration of the ball. The amount of channels required are only two thus making it a very attractive choice for the project. The microcontroller would most likely interpret them as an auto-filled large array that would have ever-changing numbers as it would pick up vibrate for many external sources. Therefore, some sort of algorithm would be required that whenever a spike appears on such array, that object with the highest value would correlate to the designated RGB LED that correspond with that specific sensor. However, after deeper examination it was determined that the

impact sensor could not sensitive enough to detect a ping pong ball. Additionally, the nature of the sensor could be problematic since all of the sensors could activate at any time, it would require some sort of high threshold to activate the aforementioned sensor. Another issue could be presented that since the table is intended for activities such as social gatherings that often include music being played a high volumes and crowds; these external factors could potentially generate vibrations strong enough that could activate the aforementioned sensors beyond the pre-established threshold and could recognize as a triggered sensor thus potentially making the sensors not fully responsive and somewhat inaccurate, thus hampering the experience of what the original objective was set. If the team wants to utilize the aforementioned sensor then it is much recommended that extensive testing must be done before fully investing in a large quantity of the aforementioned sensors.



Image Taken From Vishay Semiconductors

Permission Granted Courtesy of Vishay Semiconductors

Figure 3.3.5: TE Connectivity Measurement Specialties 1005939-1 Vibration Sensor

TCRT5000L IR Infrared Reflective Sensor

The TCRT5000L IR was another discovered solution that could solve our impact sensor problem. Many of the projects previously done that involved an array of LEDs that react to nearby objects were powered by some sort of infrared proximity sensor for either each single LED or a group of LEDs. The infrared proximity sensors would function as an array, the same way as the vibration sensors. The infrared sensors are also digital and variable as the vibration sensors. The values increase or decrease depending on how they are programmed and how close the object is from the sensor. Basically, the way that these sensors work is that they send an infrared signal, invisible to the human eye, the beam shoots in an upwards position and this infrared signal then “bounces” or reflects off the nearby object and an infrared receiver then senses this signal and based on how much it receives it can detect how far the object is. These sensors, in particular, the TCRT5000L are not entirely accurate as they are not meant to detect the distance of an object but to tell how approximate the object is in reference to the sensor. The usage of these sensor varies in many fields such as robotics. Often, these sensors are used

to tell if a robot near collision of an object to avoid an obstacle or a wall. It is quite reliable and the benefit of using this package is that it not only includes the infrared receiver but it also has a built-in infrared emitter in a single package which could potentially reduce the cost of the impact sensor array as well as wiring. One potential negative for the usage of this sensor is that in social gathering, some odd lighting conditions could occur that could trigger the aforementioned sensors making them potentially unreliable. However, as test shows, this seems to be very rare and it does not seem to affect the performance of the aforementioned sensors. Another aspect that needs to be evaluated, which also affects the previous option, is that the microcontroller does not have the required amount of pins for the array as more than 200 of these sensors will be used, therefore, the usage of an integrated circuit that would increase the number of inputs and outputs of our microcontroller, such as shift register or mux is required. Overall, the usage of an infrared proximity sensor could prove useful.



Image Taken from Vishay
Semiconductors.

Permission Granted Courtesy of
Vishay Semiconductors

Figure 3.3.6: TCRT5000L IR Infrared Reflective Sensor

Sharp IS471F Infrared Detector

Another option would be the Sharp IS471F Infrared sensor. It utilizes the same technology as the previous where it detects an infrared signal from an emitter. The benefits of using the Sharp IS471F over the TCRT5000L is that the Sharp sensor is capable operating under any lighting condition as it impervious to external disturbing lights, which could benefit the performance of the impact sensor array as the table needs to be able to fully operate under any lighting condition as many of the aforementioned social gatherings do not have the best lighting conditions. One negative, however is that the package does not include an infrared emitter thus making this solution marginally more costly than the previous. Another positive, however, is that the IS471 infrared detector includes a built in pulse driver, or in other words, the required components to attach an LED emitter with pins readily available to create a somewhat similar proximity sensor setup as the previously mentioned option, thus, somewhat compensating for the lack of an emitter by making the wiring slightly more simple.



Image Taken from Sharp -
World.com

Permission Granted
Courtesy of Sharp

Figure 3.3.7: Sharp IS471F Infrared Detector

Impact Sensor Decision

The impact sensor choice came in two flavors, the TCRT5000L IR Infrared Reflective Sensor, or the Sharp IS471F Infrared Detector. Neither option could hamper the performance nor requirements of the project as both have proven to be the most correct solution to the problem at hand. However, only one choice could be taken and after evaluating the economic aspects, the TCRT5000L was the logical sensor to utilize in the project. This could change, however, after further tests are made to verify that the TCRT5000L is capable of performing in most lighting conditions. If it is determined that the TCRT5000L is not capable of performing on the aforementioned conditions then it would be replaced by the more superior and expensive Sharp IS471F.

3.3.5 Cup Display System

This system is a means of detecting whether or not there are cups on the table for use in play. It requires a means of sensing the placement of the cups at specific locations on the table, as well as the use of LEDs to light up at those positions to indicate that they are being properly sensed and that the table will know to account for them. We will make use of the same LEDs as those in the LED array, but we needed to decide from several different options on what type of sensors we would use to detect the cups. Several of our options were taken into consideration based on their use in other past projects, but we did a fair amount of research into novel methods as well to make sure that we found the best fit for our design.

One of the options we considered, having seen it in use in a similar project, was a “capacitive touch sensor”. These are commonly used in touch-screen interfaces such as the smartphones and tablets that are frequently used today. They operate by using a principle of electronic theory which states that the charge of an object can change as a result of the influence of an external object. Specifically, an object containing a liquid conducive to electrical charge, like how the human body is

mostly made of water, can create a charge in an object under certain conditions. Knowing this, we found it was possible to use a capacitive sensor to detect whether or not a cup was in play on the basis of whether or not the cup was filled with liquid and placed over the corresponding spot where the sensor was positioned. We found enough evidence that this was a feasible option for our design in that the cups would definitely have liquid in them when used for play and that capacitive sensors are often used to detect external influences even when covered by some sort of plastic shell. The history and successful use of this technology in the past drew us towards it as an option for our design, and that is why it was considered for use.

Another option was the use of impact sensors such as the 1005939-1 Vibration Sensor mentioned earlier. Our research found that they were sensitive enough that they should have been able to detect the impact of cups placed directly on the surface of the table in much the same way that these sensors could be used to detect ping-pong balls hitting the surface of the table. That said, we wanted to consider other options for our design because the idea of needing to impact the surface for detection of the cups seemed like a faulty design. The option did have potential though, so it was considered for use.

One of the more simple options given for consideration was the use of a simple button that could be pressed by the cups when they were in the appropriate positions on the table. It requires no advanced theory and could easily be implemented into the design. There are a number of simple switches available for less than a dollar that we could purchase easily and work into our design, but it would specifically require that we create openings in the surface of the table that could potentially allow leaks of water and/or other liquids from games of beer pong to seep through and affect the circuitry. This is a risk that we would prefer not to deal with if possible. Actually, there are plenty of other options at our disposal for the project design that allow use to keep the surface in one piece, so we would likely avoid making use of any buttons in our table's surface. It's important to realize why we could have used this technology though.

A very basic option that was sure to be considered was the use of a pressure sensor to detect when the weight of a cup was placed on top of a certain point on the table. From what research we did, it was found that most pressure sensors are analog in design and can be very small in size. However, every pressure sensor that we found was significantly more expensive than any of the other options we were willing to consider. A pressure sensor could definitely be worked into our design and be able to sense the weight of objects from across the surface of the table, but it's possible that we might need several sensors placed on any one point on the table to get them to achieve the effect we are looking because they are so small. Furthermore, while the cups usually contain some liquid to give them a significant amount of weight, they could also potentially be empty and have virtually no weight to them. There is concern that the pressure sensors would be unable to accurately detect the presence of the cups when they aren't holding enough liquid,

which is a bit of a design flaw. It's worth giving our other options consideration if we can avoid this potential issue.

The final sensor we would consider for use is the infrared sensor. Thanks to the research we've done on similar projects, we know for a fact that infrared sensors can be used to detect objects through the surface of the table and accurately reflect when ping-pong balls strike the surface. This tells us that we can easily set up a series of sensors for the purpose of detecting the position of the cups when they are ready for play. Infrared sensors are cheap enough that we can purchase as many of them as we need and work them into our design easily, but they need to be set to know what distance to react to the presence of an object. Even with that to account for, it shouldn't be difficult to make use of this technology.

This system will make use of the same technology used in both the Impact Sensors research and the LED research. As such, the Cup Display System will consist of the Round RGB LEDs and the TCRT5000L infrared sensor. Together, they will provide a simple and effective solution to this part of our design by allowing us to simply detect whether or not the cups are in place for a game with the infrared sensor of our choice, and reacting to their placement by displaying the colors chosen via the LEDs we've decided to use. Beyond that, it merely needs to be programmed properly, which is dependent on entirely different parts of our design that are discussed in other sections.

3.3.6 Shift Registers

As briefly mentioned on the previous section, another integrated circuit, or IC, is required to further increase the amount of inputs of the selected microcontroller as this one does not possess the required amount of input and output pins to have an attached a Bluetooth module, the RGB LED array and the Impact Sensor array. As an LED Driver was chosen, a Shift Register is also required to house the large number of sensors required to satisfy the Impact Sensor array objectives. The criteria for this selection is not as relevant as the shift register is not required to have any special features other than low cost.

At least two of the projects that we have based ours around made use of shift registers of their own, so we immediately examined those before anything else to determine if we could make use of them. In the Game Qube project, which involved building a 3D LED cube that could be used as interface for simple games, they made use of a 74HC595 Shift Register/Latch. This component not only has the basic functionality we expect from such a component, but also has a latch function. This extra function would allow us further control over whether the bits from the register are read or not. The latch can be used to keep the signals from the register from being received until it has been activated. Ordinarily the signals would just be sent out as soon as they were available, but the latch keeps them from being released until it has been signaled that they are allowed to do so. This component certainly has plenty of potential, but we planned on making use of a simpler design that had no practical need for the use of latch in addition to the shift register.

Accounting for the latch in programming would just be an extra inconvenience, so we opted to find a shift register without the extra feature.

The only other shift register used in one of the projects we researched was a 74HC165 Shift Register. This had been used in the Interactive Touch LED RGB Table. Our project design came to make use of a lot of the information from this design in terms of how we will construct our table, so it made sense to give higher consideration to their choice of parts. The shift register they use has an appropriate number of output pins for us to increase our pin count effectively without much cost, and given its implementation into a project of similar design to our own, it is a very good candidate for our purposes. This way we only get the functions we need and don't need to be distracted by unnecessary extra features.

As there did not appear to be any significant differences between these shift registers and any others on the market that would affect our project design, we decided to make a choice between these two. We opted to use the 74HC165 because we found it suited our purposes, had no unnecessary features to work around, and came at a reasonable price for us to make use of. With this part chosen, we moved on to the next component to be decided.

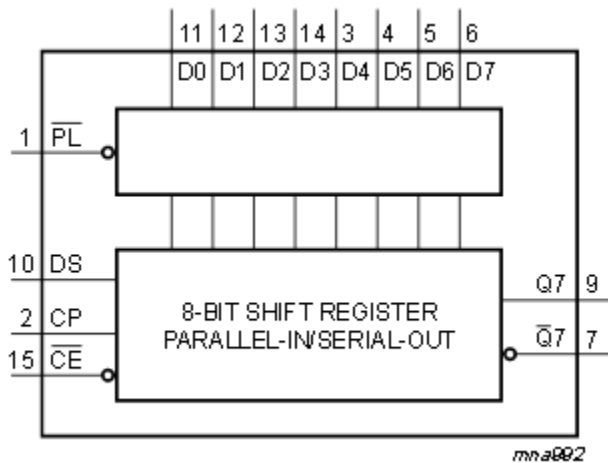


Image Taken from:
http://www.nxp.com/documents/data_sheet/74HC_HCT165.pdf

Permission Pending

Figure 3.3.8: Diagram of a Shift Register

3.3.7 Ball Cleaner

One of the structures our table will feature is a kind of "Ball Cleaner". This will sanitize the surface of a ball that is put into it. It is a well-known fact that balls used can become covered in dust and other debris that affect their ability to be used in play. Therefore, we decided to include this convenience in our design. We considered the use a few different design examples including a water pump, a golf-ball cleaner with bristles, and an air pump.

The first option we considered was the water pump design. This was used in a previous project of a similar design to our own, which means it was found useful and practical enough in this setting for them to actually build into their design. The design we found specifically requires use of a submersible pump that jettisons

water at a ball in order to clean it with a combination of water pressure and the use of a cleaning solution in the water. The pump needs to be light enough not to create a significant issue in the construction of the rest of the design as well as capable of being run indefinitely. It also requires a combination of water-tight piping that would not react with any chemicals put into the water for cleaning purposes. All of these seemed like fairly simple requirements to meet, but we wanted to be sure that we considered all of our options before going with a specific design.

Another alternative was the use of a golf-ball cleaner that had a brush. There is a fairly long history of golf-ball cleaners that feature the use of a brush to clean dirt and grass off of golf-balls during play. They are simple enough to build that it would not be any real challenge to incorporate one into our design. They merely consist of a small tank of a mild cleaning solution to soak the ball and brush in, and a holder that keeps the ball in place while the users focuses on using the brush to clean the ball. There are plenty of tutorials online to explain how to build these devices, but it begs the question of whether or not it is complex enough to put into a senior design project to begin with. Furthermore, what if the bristles of the brush scratch the surface of the ping-pong ball? The brushes used for golf-balls are designed to be thick and fairly rough because they are responsible for removing substances like dirt and sand from the surface, and these can be troublesome to remove. We obviously want our cleaner to be effective at its job, but scratching the surface of a ball could be a potential deterrent to gameplay. At the very least, we would want to avoid damaging the ball in the process of cleaning it, or use of the cleaner will be limited by this factor. With this in mind, we would consider at least one other option before making a final decision.

Our third option was to use a motorized fan that would be strong enough to remove debris from the surface of the ball. To turn on and off this motor we could use an infrared sensor. This was also built into another previous project similar to ours. This idea would naturally require the use of an fan and PVC piping, similar to the water pump design. However, we don't have to worry about the piping being completely airtight in this case because we won't need to worry about water spilling out of it. We also would be able to clean balls with this design and not need a cleaning solution applied to them in the process, which is a feature that neither of the previously mentioned designs have in their favor. An air pump is naturally less effective for cleaning any surface than a water pump, but our table is meant to be used indoors where the threat of soil and sand clinging to the ball in heavy amounts is less of an issue. Therefore, this design could still be useful enough for our purposes if we decided to go with it.

Having gone through these different design considerations, we finally decided that the combination of a motorized fan and sensors would work best for our project. We are capable of building a water pump, but that design's requirements are slightly more complex because of the need to insure there is no leaking. This is also one of the reasons working against the golf-ball cleaner's design, but the golf-ball cleaner specifically doesn't really seem like the sort of thing that should be included in a senior design project. There isn't much of a reason to include it unless

it can demonstrate our engineering abilities, so it wouldn't be more than a waste of time to include. As such, the motorized fan design appeared to fit our purposes best. It would accommodate the users by providing a means of keeping the balls clean for play, as well as demonstrate our engineering abilities by having us build it. Meanwhile, we wouldn't need to worry about leaking fluids since none are required for this design to work. With this decided, it should make a fine edition to our project design.

3.3.8 Power Supply

A power supply unit or PSU is an electronic device that supplies power to another device. From an outlet it receives power and can either transfer, change, or convert the power. The most common power supply used is an AC to DC converter. Most computer or electronic devices used require DC voltage to power them. There are several categories of power supplies which include regulated, unregulated, and adjustable power supplies. A regulated power supply holds a constant output voltage or current. An unregulated power supplies voltage and current that can change according to its load. Adjustable power supplies can be set by mechanical controls or input. The power supply is the first place a surge or spike can occur, so most power supplies are designed to handle any fluctuations. Most power supplies have fuses that will blow instead of damaging the other equipment. A fuse is a low resistance resistor that in a sense sacrifices itself to save the other components. It is much cheaper to replace a power supply than the device it is powering. It is a great idea for equipment to use a UPS or surge protector. A UPS is an electrical device that supplies emergency power to a load when there is an interruption in the input power. A surge protector is an appliance that is typically designed to protect electrical devices by shorting to ground any voltage spikes.

Types of Power Supplies

There are many ways to define a power supply. A power supply can be a regulated or unregulated power supply. It can also either be a linear or switching power supply. These all depend on what the power supply is being used for and the device it is powering. The difference between a regulated and an unregulated power supply is just how it sounds.

Unregulated Power Supply

When we talk about an unregulated power supply, we mean that the voltage or current in the output is not controlled by a regulator. This means the DC output is reliant on the amount of current used by the electrical load the power supply is supplying. The first concept to grab when discussing unregulated DC output is they are designed to produce a maximum current output to the load. The output voltage will decrease as the output current from the power supply increases to the load. For example, if the unregulated power supply is not connected or supplying no power to any electronic device at the moment, then the output voltage will be greater than its output rating. This happens because the current is lowered because it isn't supplying power. When it is hooked up to an electronic device with

the requirements met of the power supply, then the current increases to its normal rate and the voltage decreases to its normal rate. The second concept of unregulated output is to understand that they do not produce a clean output. The closer the output measurements are to the loads requirements, the more pure the DC voltage will be. The noise is a ripple in the AC voltage that is supplied from the outlet and then turned to DC voltage. This can be reduced by adding a capacitor that smooths the ripple. The last concept of the unregulated power supply which also pertains to a regulated power supply also is that the output current should absolutely never exceed the loads maximum current listed on the load. Doing this could result in cause the voltage to drop below its rated value and overload the power supply. Overloading a power supply can lead to damage of the supply if precautions are not taken or even start a fire due to the heating. When selecting or designing an unregulated power supply there are key things to look at. First you want to determine the loads power requirements of current and voltage. Then you want to design or find one that meets the similar requirements as close as possible. Don't forget that the loads maximum current rating should not exceed the maximum current rating of the power supply.

Regulated Power Supply

A regulated power supply is a power supply that converts AC (alternating current) voltage into a constant DC (Direct current). It can also supply a stable current also but is less likely than a constant voltage. Regulated power supplies are less common than unregulated power supplies but mainly due to the cost. The DC voltage output will always stay inside the designed values regardless of the current drawn from the load. There are three parameters that a systems voltage regulation depends on. The first parameter is load regulation. Load regulation is the measure of the capability of maintaining a constant output voltage regardless of the loads size. The next parameter is line regulation. Line regulation is the measure of capability to maintain a constant output voltage regardless of changes in the input voltage or source voltage from an outlet in most cases. The last parameter for a regulated power supply is temperature dependence. Temperature dependence is the capability to also maintain a constant output voltage regardless of changes in temperature of the components in the power supply. When choosing a regulated power supply you also want to check power requirements just like the unregulated power supply. When designing your own regulated power supply you want to design it to power all of your devices or as many as possible.

When building an AC output to DC output power supply and designing it, the steps in doing so are very similar despite the type. The methods of designing an unregulated power supply apply to regulated power supplies also. Regulated power supplies go through all the steps of an unregulated power supply but add the regulation or control of the output power. The first step in designing both is to change AC output into DC output. This step is called Rectification. The first thing to know is the basic standards for voltage coming from the power source. The North American standard for a normal outlet that we are using is 120 volts AC at a

frequency of 60 Hz. From here we know the input into the power supply and can start designing hence forth.

Step Down Voltage

The first step in rectification is to step down the AC input to a lower AC input that is more suitable to use. This process is called step down and is done using a transformer. Transformers transfer electrical energy using electromagnetic induction using primary windings, secondary windings, and a core. Transformers are used to increase or decrease the voltage of an alternating current from an electrical device. To step up voltage in a circuit the transformers secondary windings has more coils of wire than the primary windings. Since we need to lower the alternating current we are using a step down transformer which has more coils of wire on the primary windings than the secondary. The alternating current in the primary coil produces a magnetic field in the core which then produces a potential difference in the secondary windings. If we were to assume the transformer worked at 100% efficiency then the equation used to calculate the potential difference would be:

$$\frac{V_p}{V_s} = \frac{n_p}{n_s} \quad \text{and} \quad V_p \times I_p = V_s \times I_s$$

Half Wave Rectification

Once the step down of alternating current has been done then we are ready to convert the alternating current into direct current. Diodes conduct current only in just one direction going from the anode to cathode and not in the opposite direction. Thus it makes sense to use them to convert a two directional alternating current into a one-way direct current. The input form in AC shows a sinusoid that alternates between positive and negative half cycles. The output from a diode connected is rectified only having either positive or negative half cycles but not both. This type of design is called Half Wave rectification. A simple half wave rectifier is not a very complicated design. It consists of just a single diode. This diode is a pn junction diode which is connected into series to the load resistor. So in this design the input is the alternating current which has been stepped down by the transformer and the reduced voltage is given to the diode. During the positive half cycles of the input sinusoidal wave the diode is forward biased and during the negative half cycles of the diode it is reverse biased. Thus the diode only lets current pass through during the positive cycle of the wave. When the wave hits the negative half waves it instead reads a zero output. The disadvantage of a half wave rectifier is pretty obvious since the waves are not continuous, thus creating a high rippling effect. If we applied it to an audio connection then there would be a high amount of noise and result in a bad quality of audio output. The advantages of a half wave rectifier are that they are cheap and simple to make or apply. Thus a more reliable and superior design is to use a full wave rectifier. The schematic and input/output waveforms are shown below.

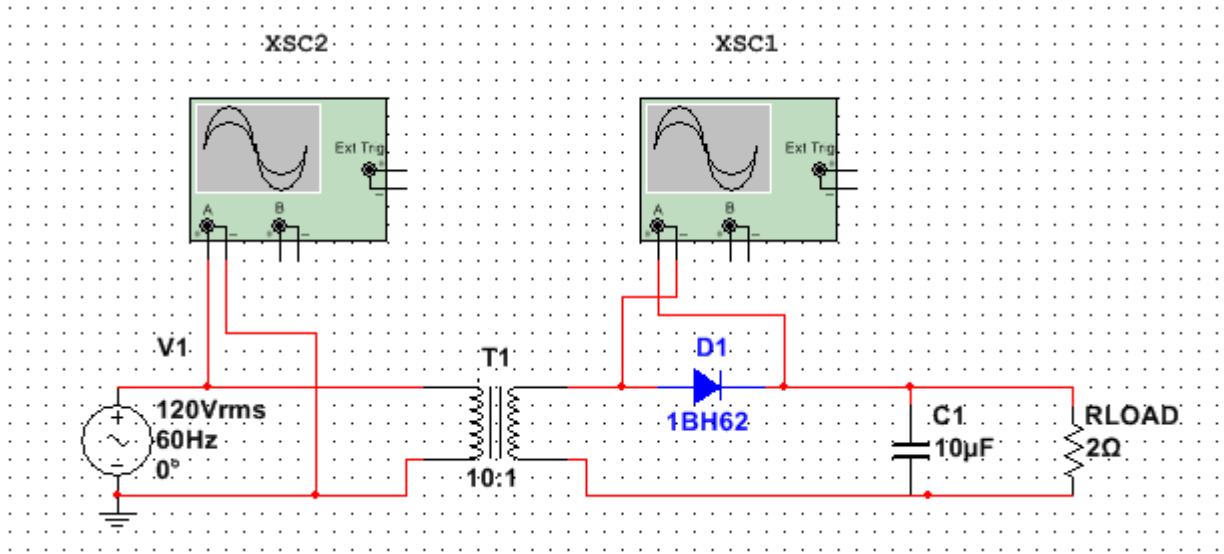


Figure 3.3.9: Half Wave Rectifier Circuit

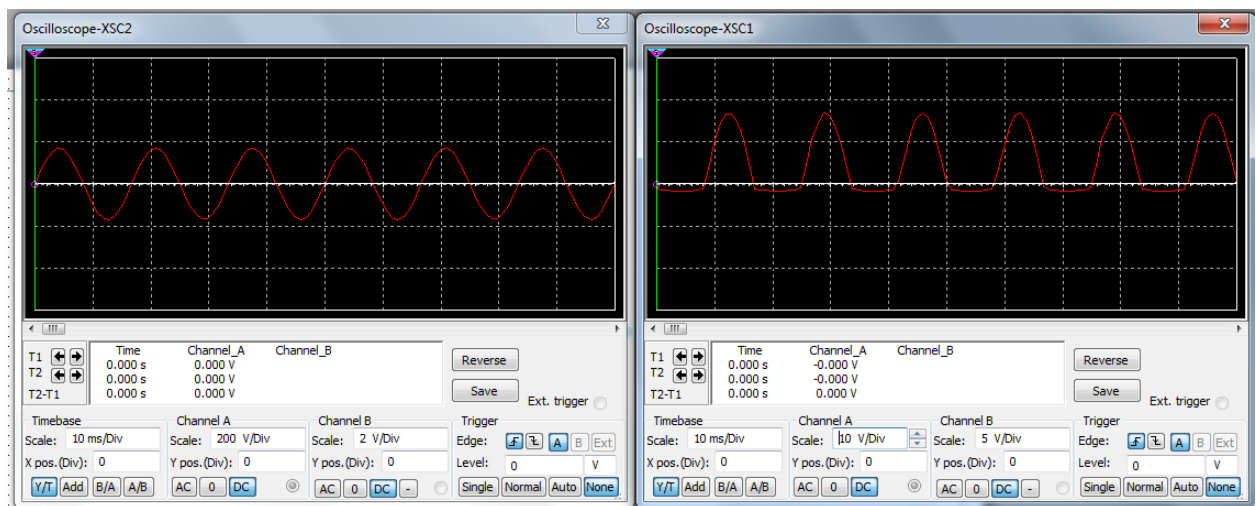


Figure 3.3.10: Half Wave Input waveform (left) and Half Wave Rectified waveform (right)

Full Wave Rectification

Just like a half wave rectifier a full wave rectifier consists of diodes. But instead of one diode, it uses more than one. In a full wave rectifier two diodes are connected to the circuit. The first diode is for one half of the cycle and the other diode is used for the second, eliminating the zero result from the negative cycle. This creates a continuous positive cycle in the output wave instead of a half positive waveform. In this design we need a transformer with a centre tap connection though resulting in a larger transformer. Each diode is connected to one end of the secondary windings on the transformer and the load resistor connected to the centre connection on the transformer. Each diode conducts in turn producing a half cycle each combining to make the continuous full wave cycle. The two diodes and

resistance load are connected on the other end which allows the diodes to take turns supplying power to the load resistance. When point 1 of the transformer is positive with respect to point 3, the Dtop diode conducts in the forward direction. When point 2 is positive or also called the negative half cycle in respect to point 3, the Dbot diode conducts in the forward direction. Thus where the empty spaces are for a diode, the other diode is filling them in making the output DC voltage double that of a half wave rectifier. The schematic and output waveform is shown below.

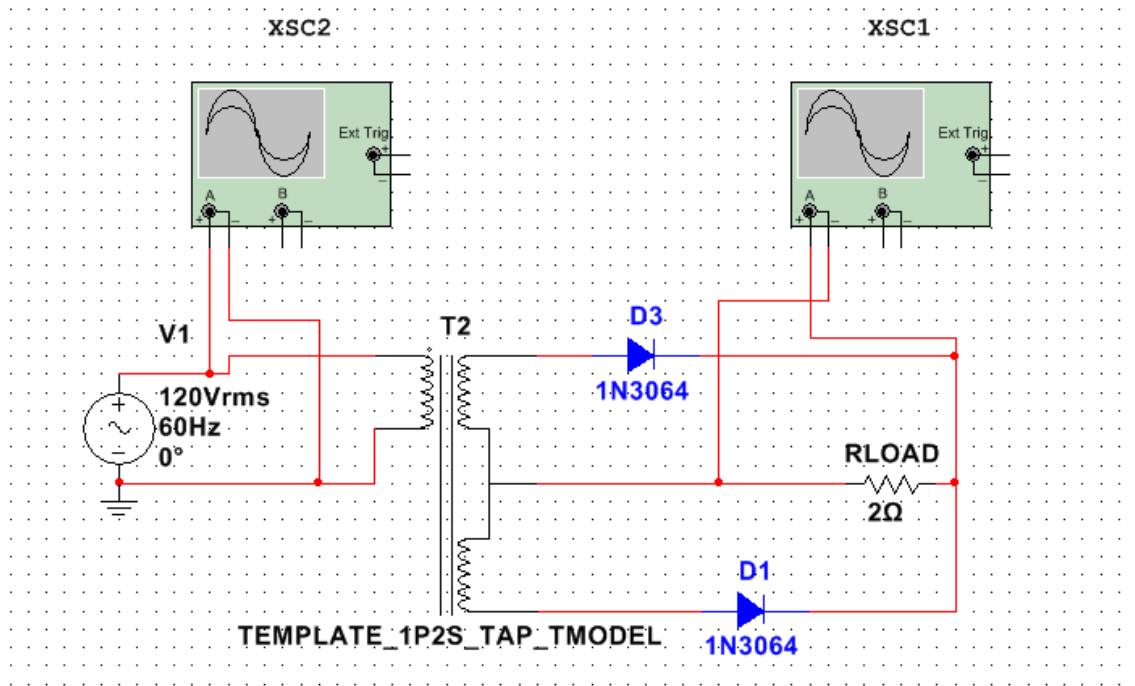


Figure 3.3.11: Full Wave Rectifier Circuit

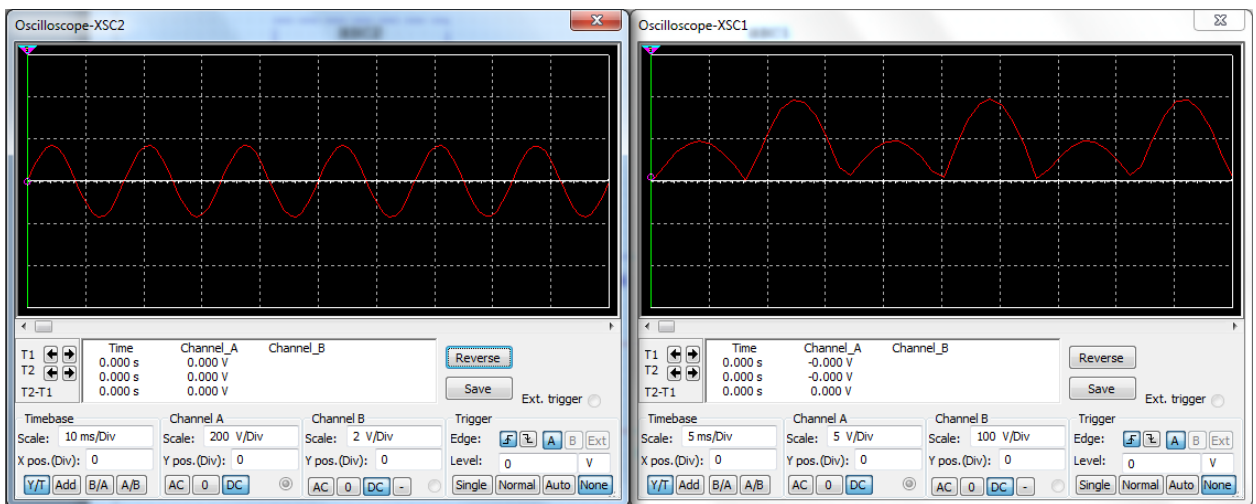


Figure 3.3.12: Full Wave Input waveform (left) and Full Wave Rectified waveform (right)

Full Wave Bridge Rectifier

To get rid of the center point in the transformer we use a full wave bridge rectifier. It produces the same output as the full wave rectifier but does not require the centre point of the transformer. Instead of two diodes in the full wave rectifier, the bridge rectifier consists of four individual diodes connected in a closed loop. Eliminating this center point reduces the cost and size of the transformer and makes the schematic a minor amount simpler. The transformer secondary windings are connected to the full wave bridge rectifier as shown below. The four diodes are arranged in series pairs with both pairs producing power during each half cycle just like the full wave bridge rectifier. During the positive half cycle of the input D1 and D2 produce an output and conduct in series while D3 and D4 diodes are reverse biased. During the negative half cycle of the input D3 and D4 are forward biased and produce an output in series but D1 and D2 switch to reverse biased. Although the direction of current changes in the rectifier and transformer, the current through the load remains the same during both positive and negative cycles. The figure below shows the direction of flow during both positive and negative half-cycles. You can make a full bridge rectifier using four diodes but there are premade bridge rectifiers in several different ranges and sizes.

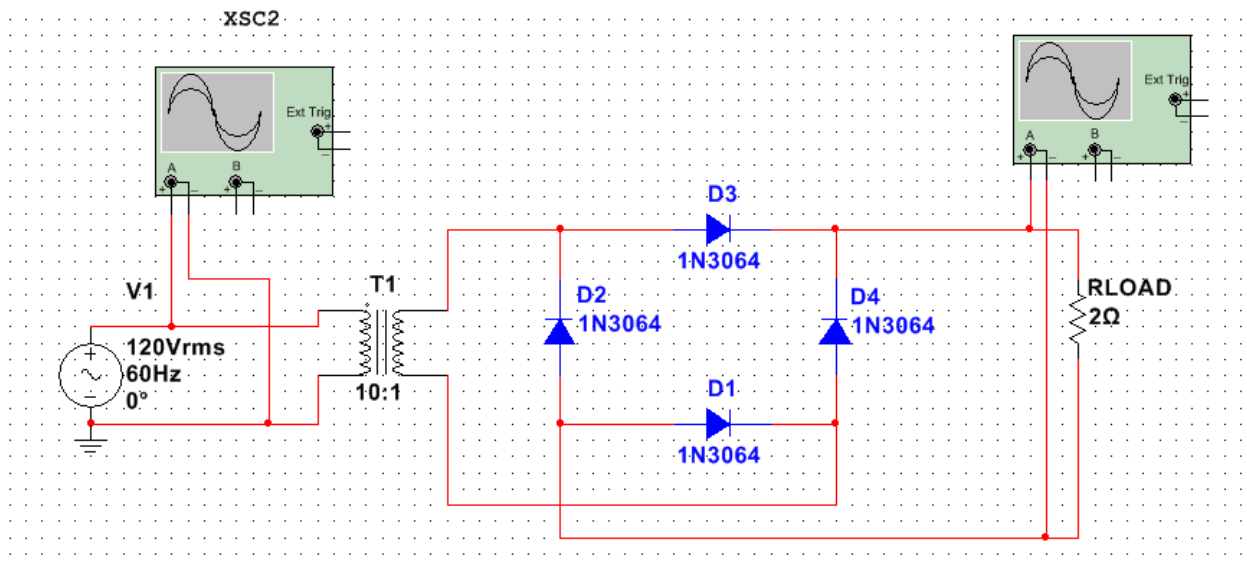


Figure 3.3.13: Full Wave Bridge Rectifier Circuit

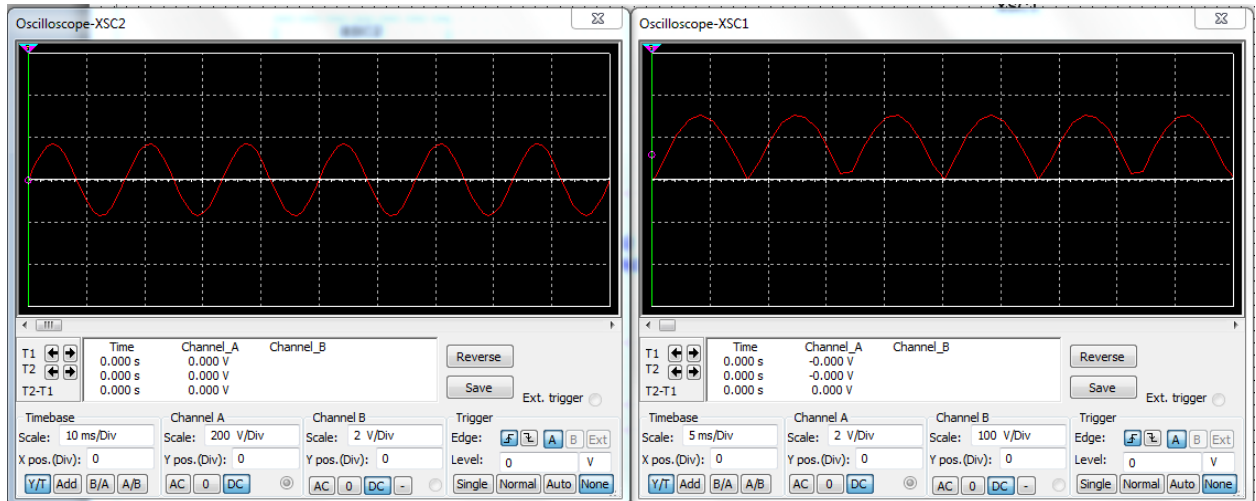


Figure 3.3.14: Full Wave Input waveform (left) and Full Wave Rectified waveform (right)

Smoothing Process

Before we start to regulate Power we want to cancel out as much ripple effect as we can. Smoothing is the process of stopping the DC voltage from the rectifier from falling. This is done by a high value capacitor acting as filler by applying current to the output when the voltage wants to decrease. The capacitor charges as the voltage reaches its peak then discharges smoothing out the output voltage. The smoothing capacitor converts the full wave output with ripples into a much smoother DC output voltage. When designing the smoothing capacitor into the circuit we have to make sure its working voltage is not higher than no-load output amount of the rectifier. The ripple voltage is not only affected by the capacitor but also by the frequency and current. We aim for a ripple voltage no higher than 100mV peak to peak, thus we use the formula below:

$$V_{(ripple)} = \frac{I_{(load)}}{f * C} \text{ (Volts)}$$

The fundamental frequency of a full wave bridge rectifier is twice that much of a half wave rectifier due to the doubling of its output waveform. Thus the Ripple voltage is half that of a half wave rectifier if the current load and Capacitor are the same ones used for each. Thus we can use a smaller capacitor for the full wave rectifiers and the smoothing is much cleaner as shown below.

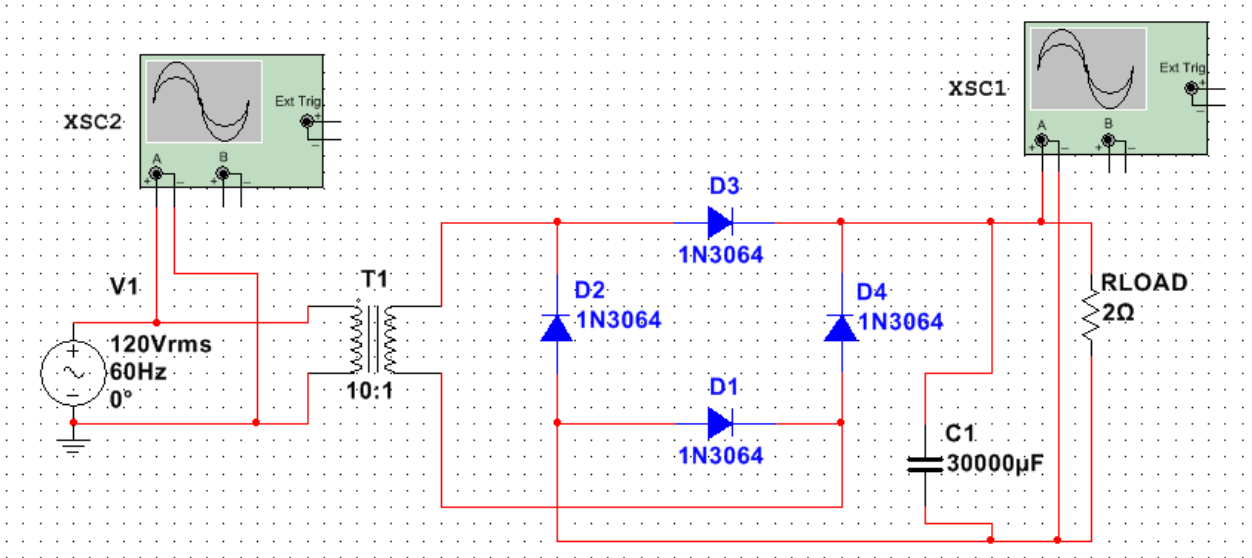


Figure 3.3.15: Full Wave Bridge Rectifier with Smoothing Capacitor

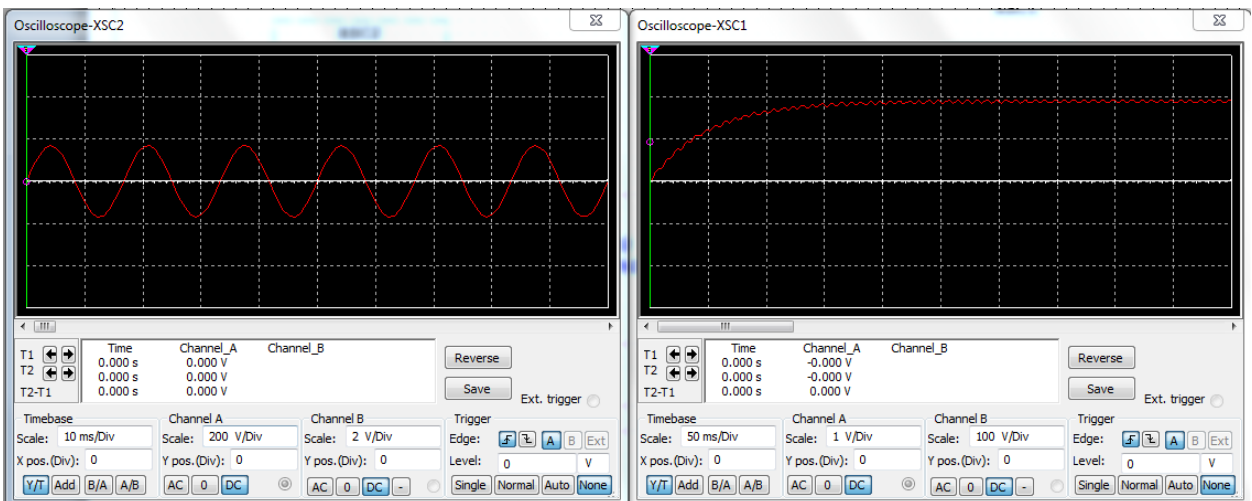


Figure 3.3.16: Full Wave Input waveform (left) and Smoothed Full Wave Rectified waveform (right)

Regulating

There are generally two basic types of ways to regulate a DC power supply. Linear and Switching are the two main ways to regulate power. This is the most complex part of the design because it controls most of what your output is going to be and how it's going to be used. There are key things to look at when choosing between the two. For example, when designing the power supply you have to choose its efficiency, size, weight, complexity, noise, heat dissipation, power factor, and much more.

Linear Regulated

The first type of regulator design we will discuss is the linear regulated design. In this design the output voltage is regulated by dissipating the unwanted power in the form of heat. This is done by usually a resistor or a passing transistor in the active mode. The size and weight of a linear regulated power supply is generally heavier and bigger than switching regulated power supplies due to usually more heatsinks because of the heat variable. The linear regulated power supply has less parts usually also compared to the switching regulated power supply. Linear power supplies have a high level of performance. They get their name from the obvious reason that they use linear techniques to regulate the output. Linear power supplies are mostly used when noise is an important part of what is being supplied. For example, if audio electronics are being used we would want to use a linear power supply because audio components work best when noise is minimized to achieve the best performance. Explaining how a linear regulated power supply is simpler than designing it but the first thing that usually occurs is the output voltage after the smoothing process compares to a reference voltage. The difference of the two is then fed to a transistor which allows more or less current through. A MOSFET type transistor is usually used for this design and functions in a linear region. The problem with the linear power supply is the size, weight, and efficiency. The size is needed due to its low frequency and its efficiency is lower because it is dissipating heat which is wasted power.

Switching Regulated

The second method of regulated output is by using a switching power supply. The switching regulated power supply has a much higher frequency thus the transformer size and weight decreases. The switching regulator is generally usually off or on. The transistor has a small voltage drop across it thus less power is being wasted. When it is off then no power is being wasted. Another benefit of the switching regulator is that because the frequency is higher we can then reduce the smoothing capacitor. Most switching regulated components range from 50 kHz to around 1 MHz. Another advantage of using a switching regulated power supply is that the switching can be regulated with a feedback circuit that turns the MOSFET transistor on and off. There are disadvantage of using a switching regulated power supply also though. Due to the high frequencies this means there is a higher level of noise or electromagnetic interference (EMI). Also, since the switching regulator needs current to run, the power supply has trouble running at low voltages and often has a minimum output voltage requirement.

Specification	Linear Regulated	Switching Regulated
Size and Weight	Heavier Due to Transformer size and heat sinks.	Lighter weight and fewer parts.

Efficiency, Heat , Power dissipation	Low Efficiency (30-40% avg), dissipates larger amount of heat, wastes power in form of heat.	High efficiency (75% avg), transistor is on or off reducing wasted power.
Complexity	More Complex and more parts.	Less complex and less parts.
EMI Interference	Low noise and EMC interference due to less frequency.	Higher frequency results in more noise.
Output Voltage	Cost goes up for more power	Cannot supply a low voltage, costs much less to increase power.
Output Current	Can produce a low amount.	Must supply minimum current.

Table 3.3.6

3.3.9 Smartphone/Computer App

The smartphone application and computer application are important for this project because these applications will be used for the user interface component of the **The Beer Grid**. So in order to determine how to implement the user interface of the **The Beer Grid** the project group has to determine certain things about the phone application and the computer application.

3.3.9.1 Smartphone App

The smartphone app is important in that it will be the main component for the user interface. For the smartphone, the project group will have to take into account several things in order to determine which mobile operating system will be best suited to fit the needs of the group. The factors that we will be considering when choosing which smartphone operating system to use, the group will take into account such things as how much it will cost to develop, how many users are currently using smartphones with the desired operating system, and the groups accessibility to develop on the designated operating systems. When it comes to cost, the group is looking for the most cost effective smartphone operating system. Besides cost, the project group is also looking for a platform that has a high amount of users to enable the smartphone app to be used by a good amount of people. The project group wants to use the system that the group can most readily take our current experiences and quickly incorporate it into developing for the desired operating system.

iOS Operating System

To develop on the iOS operating system, the project group will need a Mac computer running OS X 10.9.4 or later. The development of an iOS app will require the need for Xcode integrated development environment (IDE) and the iOS Software development kit (SDK). In order to use the Xcode IDE, the project group will have to obtain it from the App Store for free on the computer running the Mac OS. When downloading the Xcode IDE, the group will also receive access to the iOS SDK is included in the Xcode download. Although the required IDE and SDK are free, due to the fact that the group does not have access to a Mac computer, the group would need to invest in obtaining one if the group choose to make an iOS application.

Another option the group would have to consider would be the amount of users that are currently using the iOS platform. According to IDC, iOS has an 18.3% market share during the first quarter (Q1) period of 2015. This statistic shows that it is the second most abundant mobile operating system on the market. Even while being the second most mobile OS on the market, the iOS is only used on one line of smartphones called the iPhone. So when developing an app for the smartphone, the group would not have to worry as much about bugs that may appear as the result of developing for different lines of phones.

The last thing that the group will have to consider is to see if they can readily build on the iOS operating system. As of the moment this paper is being written, no one in the group has developed or worked on an iOS application. This means that the group would be unfamiliar with the Xcode IDE as well as the iOS SDK. The programming language used for Xcode is Objective-C programming language which is a program language the project group has no experience working with. However, Objective C is built on top of C programming language and it provides object-oriented capabilities. The project group has experience working with the C programming language and object-oriented programming. With this prior experience, the project group should not have too much of an issue programming with Objective C should the group choose to create an iOS application.

Android Operating System

To develop on the Android operating system, no specific computer operating system is needed. So if the project group wanted to, they can develop on a computer running Windows, Mac OS X, or Linux. If the group were to develop on Windows, the oldest version required to do so is Microsoft Windows 2003. For Android application development on Mac OS X, the group would need a computer running at least Mac OS X 10.8.5 or higher. Should the group decide to work with Linux, a GNOME or KDE desktop that runs GNU C Library (glibc) 2.15 or later is needed to develop an Android application. No matter what operating system is used, the Java Development Kit (JDK) 7 and the Android SDK is needed, but that

can be obtained for free. Android has its own IDE ready for anyone to use called Android Studios but the group can choose another IDE such as Eclipse in order to develop the Android application. Both IDEs are free to download. So since the project group has access to computers running versions of Windows that fit the system requirements for Android Application development, the project group would basically spend no money in order to develop our desired application using the Android Operating System.

Just like with Mac iOS, the group has to take into account the population size of the users that are using Android based smartphones. Well according to IDC, the Android operating system has a 78.0% market share for the first quarter period of 2015. This is by far has the largest portion of the market share of any mobile operating system. However, although it is the most widespread mobile OS within the US market, the Android OS, unlike the iOS, consists of many different lines of phones the more well-known ones are the Samsung Galaxy and HTC One brands. So due to the fact that the Android operating system is spread out more between different types of smartphones, if the project group were to develop on the Android platform, the group will need to take into account of bugs and different features that may appear on different smartphone lines as the development of the app takes place.

The group will now have to determine how well their prior experiences will translate to working on the application using the Android operating system. For building the Android application the project group can use Android Studios but they are not limited to only that IDE. Android application bases its development on Java. Besides Android Studios, the Eclipse IDE is also a capable choice when it comes to developing for Android. In order to develop on Eclipse, the free plugin called Android Development Tools (ADT) is needed. While only one in the project group is familiar with Android Studios, everyone else has prior experience working with the Eclipse IDE or has some background when working with Java. Although the project group has limited experience when working with Android Studios, due to the experience with java programming, it should not be too difficult for the group to become more familiar working with the IDE. However since the group the majority of the group is more familiar with working on Eclipse IDE, the group will probably become more acquainted to developing the Android application on it instead. No matter which IDE the project group decides, the group should have no issues to develop the application portion of the project on the Android operating system.

Windows Phone

To develop on the Windows OS for smartphones, known as Windows Phone 10, we need a computer that runs on a Windows version of either Windows 8, Windows 8.1, or Windows 10. Once the computer has the desired Windows operating system, the next software we need to develop a Windows application is the Windows 10 SDK. This SDK includes all the tools the group would need in order to create a Windows Phone application. In this SDK is Microsoft Visual Studios 2015, SDK templates and tools, Windows Phone emulator, and more. The project has access to a Windows operating system that can be used to develop a Windows

Phone app and all the software for it can be acquired through the means of free downloads. So the project group can start developing a Windows Phone application for a relatively no money.

The next thing the project group has to take into account are the amount of people that currently has a smartphone that uses the Windows mobile operating system. According to IDC, the Window Phone OS has a market share of 2.7% at the end of the first quarter of 2015. Of the three mobile operating systems that the project group is considering using so far, Windows Phone OS has by far the lowest user support compared to the other two mobile operating systems. This means that the amount of people that have a Windows smartphone is comparatively rare compared to those that have iOS and Android OS smartphones. Although having a much smaller size of the market share, Windows Phone OS is similar to Android's in that they both have multiple different lines of phones that run their OS. So, if the project group decides to develop on the Windows platform, like android, the group might have to be on the lookout of different bugs and different features that the different types of phones may have.

To work on the Windows phone OS, the project group would need to use the programming languages C#, C++, or JavaScript within Microsoft Visual Studios 2015. The project group is relatively familiar with working with these programming languages. However, the group are not too familiar with working on Microsoft Visual Studios. With the group having some sort of experience with the programming languages, it should not be too hard for the group to get used to working in Microsoft Visual Studios.

Smartphone App Platform Choice: Android

The project group had to take into account several conditions when choosing which mobile operating system we wanted to develop our project's application on. The first thing we had to consider was price. If we wanted to develop our application using iOS, the group would have to invest in a Mac in order to build it. However, both the Android OS and the Windows Phone OS requirements are setup to be more compatible with the hardware the group already owns, so development on both platforms would be relatively cheap. Next the group took user count into consideration. Android leads the trio of mobile operating systems by a lot with them having almost all of the market share is made up of phones with their operating system. While android has the most, iOS has a relatively adequate amount of users with their operating systems taking about a fifth of the market. Now while those two have a good amount of users, Windows Phones do not even come anywhere close to the other two in user population size. Not too many people use Windows Phones. Finally the project group had to take into account which platform will be the best to work on giving the skills that the group are already in possession of. No one in the project group has any experience working on iOS or Windows Phone development. However, between Windows and iOS, Windows Phone OS would be easier for the project group to work with due to the fact that the group does have some familiarity with the programming languages being used for development. But the group does have experience with working on Android

Studios as well as being familiar with the programming language being used. So with taking price, user base size, and accessibility into account, the project group has decided to use the Android operating system for the development and platform of the project's application.

Applet Technologies

For the applet we need a technology that can handle interacting with a database and tracking scores. Some of the things the project group has to take into consideration for choosing this technology is which technology is readily accessible to the project group given what the group has experience in. Besides that, the project group just has to determine whether or not the animation applet technology has the features needed in order to create what the project group needs to for the animation applet.

Java

One of the implementations of applets the project group is thinking of using is the Java applet. To write a java applet, the group would need to be familiar with programming in Java programming language. It can be used in a variety of Java supported IDEs such as Eclipse which the project group is very familiar with it. To embed the Java applet into the web page is to just add some lines to an HTML file. This way of developing the applet can be immediately picked up by the project group.

Flash

The next applet technology the project group is considering is Flash. Flash applets are created in xml files. These files can be edited in numerous IDEs like Code Blocks, which the project group has previous experience with. Although the project group can readily start developing on a Flash platform, the tools for flash development are not free. However, compared to Java, Flash applets are known to be faster and more stable than the Java applets.

Jython

Another applet development technology that the project group has taken into consideration is Jython. Jython is an implementation of the python programming language designed to run on the Java platform. Jython is capable of handling anything a Java applet can handle so it is suitable for this project. Jython, as the name indicates, is a combination of python and Java modules. So since our project group has experience in both programming in Java and programming in python, Jython is a viable option to develop the project's applet.

Applet Technologies Choice: Flash

Although there may be a monetary expense with it, the group has decided to try and implement the project's design applet by using Flash. Although our group is familiar with Java and Jython our group has decided to go with Flash for several reasons. First is that flash is known to be faster and has a more stable implementation than Java. The group did not want to work with Jython due to the fact that Python applets have known to be slower and the group prefers not to work with python again.

3.3.9.3 Database

The database for the project will require to store various information for the project such as user information and game history. This information will be presented on the web application and smartphone application. When it comes to how to decide which database to use, the project group must take into account whether it can handle what the project group needs it to do. The project group must also take into account the accessibility of the database to allow the project group to get a better understanding of the database.

MS SQL Server

Microsoft SQL Server is SQL scripted database management system. Their standard version has several key missing components. This does not have hot-add memory which is what allows memory to be added while the server is still running. However, due to it being a SQL scripted server, the group would not have any problems working with it due to the fact the group has a lot of prior experience handling SQL.

IBM DB2

IBM DB2 is the next database server system that the project group was considering. For scripting, the database uses SQL and XQuery. IBM DB2 has APIs for various languages that include C, Java, and PHP which are all languages the project group is acquainted with. The project group is also familiar with Eclipse which supports the integration of DB2. Given all the familiar aspects of the database server system, IBM DB2, the project group can safely assume that it would be possible to use this server system to handle the database of the project's system.

MySQL

MySQL is an open source database that uses SQL scripting. Of all the databases, the project group has the most experience working with MySQL. It is a free database service that has all the capabilities needed for the project. So,

the project group can be confident in the decision that MySQL can be used as this project's database system.

Database Choice:

All options of databases that the project group has explored, are all viable options for this project. However, due to the fact that its better financially and that the project group has previous experience with it, the project group has decided to work with MySQL for the database needs of the project.

3.3.10 Table Building Materials

Naturally, our design involves building a table, so we needed to decide on what sort of materials we would be using to build it. This isn't the most technical part of our design, but our project is dependent on the materials used here to be capable of sustaining the weight of our circuitry and components, as well as the weight of cups on the surface of the table, and the possibility that a person or multiple persons will be leaning across the surface or on the edges. Simply put, we need a reliable table to serve as the platform for the rest of our design. Our most feasible options included the following: wood, plastic, metal. Regardless of our choice, the materials would either need to be capable of being adjusted to our specifications. This means that the materials could either be carved or molded like wood or plastic, or it would have to be available in pieces that could be connected in the right way to meet our requirements. Metal is much harder to change the shape or size of than wood or plastic, but there are plenty of stores that will offer to do the shaping or provide appropriate, pre-crafted pieces for us to use. With these options in mind, we decided to take a closer look at our options to make a final decision.

The first choice to come to mind was to use wood. Wood can be easily obtained without too much expense, and shaped to the purposes we had in mind without too much trouble. One of our project members, Colton, already has possession of the necessary materials to build a table out of wood. As long as the wood is arranged in a sturdy design, there is no reason to think it wouldn't be capable of sustaining the necessary weight for our purposes. However, wood needs extra care in order to make it safe for common users. The edges and surface need to be sanded in order to keep common users from getting splinters carelessly. We would also need to put some sort of coating onto the surface to make it waterproof. This table is supposed to be used in a game that distinctly requires fluids to be left on its surface, so it would be preferable if as much of it was made water resistant as possible. This way, water wouldn't damage the table's structural integrity or seep into the circuitry and cause it short-circuit. We can make the surface water-proof if we want to, but it would require applying some sort of coating after the table is completely built. This could take extra time and we can't be certain if we'll be able to manage this or not. In any case, we need to look over our other options before we make a final decision.

The second option we considered was plastic. Plastic on its own is ideal for our purposes just because it is a simple and waterproof material that would hold together well with the right design. To construct a table out of plastic we would need to first confirm our design for it, then construct a mold to our design and fill it with liquid plastic that would dry later. This isn't too difficult, but once built, making sure that we are able to modify the design here and there as needed could be a little troublesome. We might need to drill holes to let wires through or make adjustments to the inside space of the table where the circuitry is held. This could be a little troublesome since plastic isn't as easy to modify once it turns solid. It's still possible, of course, but trying to modify it carelessly could result in cracks that would damage the structural stability of the table. The best way to avoid dealing with such problems might be to find pre-constructed parts made from plastic, like "K'NEX" pieces or LEGOS and putting them together in the form of what our design needs to be. We could even purchase a table with the right size and add extra parts to it to house our circuitry and function as needed. The trouble here, however, is cost. The number of sets of K'NEX parts we would need to purchase to ensure that our table was sturdy and fit our design would easily come to about \$80 at least. Purchasing an actual table would be at least \$60. This option could work, but to make sure that it fits our purposes, we need to give our last option some consideration first to make sure that we are choosing the most economical and reliable parts for our design.

Our third option is to use metal in our construction. Metal inherently costs more to use and is definitely harder to customize. Whatever metal we used would definitely be sturdy enough to hold our design together, but we would need to put several safety measures in place for the sake of our users. One specific issue that would need to be worked around is making sure that the inside of the table has enough insulation to prevent the circuitry from potentially sending a charge through the table and hurting our users. This is an especially dangerous possibility when you consider that any spilled liquids could extend the reach of the table's charge. Even if we managed to setup effective measures against this possibility, building a table from such materials is no simple task. We could just purchase a table already made and add some adjustments for our specific purposes, but that could run a couple hundred dollars altogether. None of our members have the appropriate experience for altering metals themselves so we would have to pay for the parts we wanted and pay for them to be altered in any way. There are different pieces of metal available for purchase at hardware stores though, so the total of everything we would need could measure to about \$50. This isn't so bad really, we just need to be sure that we have what we need to make the design safe. Knowing that there is such a potentially dangerous drawback of using metal in our design, however, we should take a final look at all of our options before we make a final decision.

Having weighed each of our options equally, we decide to use wood as our primary building material for the table. Wood would naturally act as an insulator against the circuitry inside the table to keep the framework from carrying charge that could potentially hurt the users. It is also easy to obtain and modify cheaply because one

of our team members has the tools for doing so. Plastic makes a good insulator and has the added bonus of being waterproof, but is more difficult to use and modify. Purchasing it in pieces doesn't justify the cost either. Metal is sturdy and easy to find and have made to our specifications, but the safety of its use is of greater concern. Since it isn't necessary to use metal specifically for our design, it is best to avoid the risk altogether. Knowing that our framework will be made of wood, we have decided that we will take whatever necessary measures we need to so that we can prevent liquids seeping into the circuitry and try to waterproof the table any way that we can to insure its structural stability. This wooden framework, plus a frosted glass pane for the top surface, which serves to distribute the light from our LEDs, will serve as the structure for the rest of our project to be built around.

3.3.11 Ball Design

There was some concern as to whether or not we would need a special type of ball to be used in the design of this table. We needed to make sure that we could sense when a ball hit the surface of the table so the rest of the system could react. Maybe making the ball heavier than normal or sticking an infrared emitter inside might make it more reliable with our design. There is only so much we can do to adapt the design of a simple ball without making it unnecessarily cumbersome to use, so besides these options our only real choice is to stick with the use of an ordinary ping-pong ball. As such, we decided to look through each of these options shortly before making a final decision.

The first thing we considered was making the ball heavier somehow. We could always try using a rubber ball and it would easily still fit the purposes of the game. It could bounce off of the surface of the table and be heavy enough to trigger any pressure sensor. This wouldn't matter if we decided to use an infrared sensor, of course, but we were still uncertain as to which option would best suit our purposes at the time. The only real trouble with using a rubber ball is that users might be frustrated by the fact that they needed to use rubber balls instead of ordinary ping-pong balls given that ping-pong balls are the traditional option. Overall, the use of rubber balls was dependent on whether we felt it was to our advantage to use a pressure sensor over any other type in our design.

The next option put under consideration was the use of the use of infrared emitter in the ball. We could put infrared receivers into the table for the emitter inside the ball to send signals to. The trouble is making sure that the ball stays balanced, even with the emitter inside, and that the emitter always manages to signal the right receiver when it lands. This would require putting multiple emitters into a single ball in order to make sure that there would always be one to set off the receivers in the table, but this hardly seems feasible. The most we could assuredly managed might be two emitters in a single ball, without having to adjust the size of the ball significantly. Even then the ball would probably be unbalanced and might not work the way we need it to. Furthermore, there is some concern that one of the

ball's emitters would set off a receiver other than the one it lands on. Were this to get out of hand, it would rather defeat the purpose of our design to react to impacts from the balls in play. Therefore, this would be the least favorable of our options. Just the same, it deserves equal consideration until we make a final decision.

Last but not least, we could always use a simple ping-pong ball. The game is designed to make use of them normally, and there are pressure sensors and infrared sensors that can be used to detect their motion and provide input for our system. The concern, however, is that we may not be able to find sensors that can detect the ball's impact from beyond the surface of the table, like we want to in our design. We would need to be certain that the impact from a ping-pong ball would still be strong enough in the case of using pressure sensors, or be able to reflect enough infrared light to trigger an infrared sensor. If these conditions could be met, there would be no need to adjust the properties of the balls used in play, which would be more pleasing to users since they wouldn't need to worry about losing the balls or needing to replace them. Therefore, to make a final decision, we needed to be able to say how the sensors we plan to use would react.

As mentioned earlier, much of our decision in this matter was dependent on the nature of our sensors on the table. After some testing with both pressure-based and infrared sensors, we finally determined that there were no difficulties with either sensing an ordinary ping-pong ball. As such, we decided to take advantage of the opportunity to use ordinary balls in our design so that we would not need to spend unnecessary time working on an extra component that was tedious to build and difficult to implement properly. Therefore, our final decision was to make use of a simple ping-pong ball in the design.

4.0 Standards

4.1 Safety Standards

Our project will involve some specific safety concerns that need to be met in order to insure that it will be able to function in the presence of untrained users without being a threat to their safety. One such concern is the fact that our project is an electrical device that is specifically designed to interact with water. To insure that those interacting with the table will not be in any danger from the spills of liquids, several standards will be used for reference in implementing our design.

In addition to the concerns with exposure to liquids, we need to be sure that our table will be able to support the weight of the equipment involved in its design. There are a few basic standards (E2126-11) that we can use to test the strength and stability of the materials involved to make sure that they can sustain the weight and even deal with the possibility of moisture absorption. We will examine these and use them to determine our methods of testing.

4.1.1 ASTM D2633-13a

This standard specifically concerns the jackets used for wires and testing them for usability purposes. Taking some information from this standard could be important to insuring that our waterproofing methods are safe enough. If they can match up to this standard we can definitely have some safety assurance since most cable jackets have a small level of waterproofing. That said, we may not necessarily use wire-jackets specifically, but something similar to them like a waterproof seal on a wall.

The properties that the wire-jackets were tested for include a water absorption test to determine how they react in a wet environment. The test specifically requires that the wire jackets be kept at a certain temperature and then cooled before being submerged in water that is kept within a specific range of temperatures for a period of 14 days. Afterwards, the jacket is removed and its properties are examined to determine if any significant wear has occurred. Specifically, they look at the thickness of the insulation, the capacitance values, and the conductor's size. If these properties change beyond acceptable values, then they know that the jackets are not suitable for their purposes.

For our case specifically, we rely on these test examples to build test of our own in determining the safety of our device. We can test our own wire-jackets to this same standard if we desire, or adapt them to test other materials that will be involved in its design. The basic idea is to submerge and object and judge its changes after several days, so we may try this approach on the surface of our table to determine if it will absorb any excess liquids. We could also stage tests to determine if leaking occurs by using materials with absorbent properties to plug crevices in the table as a test to see if they have to absorb any liquids as a result of insufficient water-proofing in our design. The exact tests developed for our design will be discussed later.

4.1.2 ASTM D2395-14

This standards specifically concerns how to test “wood and wood-based materials” to determine their “specific gravity (relative density)” (D2395-14). Testing this property of the materials involved in our table’s structure is necessary to determine whether or not it will be able to support the load of all of the equipment involved in its design. Once our building materials have been tested, we can decide what to use for the physical framework of the table and fit our device components in as needed.

There are six different test methods given in the standards: Volume by Measurement (Method A), Volume by Water Immersion (Method B), Flotation Tube (Method C), Forstner Bit (Method D), Increment Core (Method E), and Chips (Method E). Method A is designed for objects that have smooth surfaces and regular shape because they can easily be measured and their mass determined through calculation, but when objects have a rough or uneven surface and/or an irregular shape to them, it is more viable to use Method B. Method C isn’t as accurate, but can be used easily if precision isn’t heavily required. Methods D and E are actually built for handling the measurement of living trees or “in-place

elements” (something that is too difficult to obtain a smaller sample of to examine easily) that can’t be easily measured and therefore need to be implemented carefully to insure accuracy. Meanwhile, Method F is simply meant to be used for measuring the density of wood chips. Between these different options we should easily be able to determine the properties of the materials we consider in our table’s framework.

For the purposes of our design, we should be able to effectively determine the density of our building materials from Method A. This was designed to handle the measurement of objects at any moisture content as long as they are regular in shape and have smooth surfaces. We can easily find our build parts for our table with those properties. As for the other test methods, Method B would only be viable if we waterproofed our building materials first, which may become a necessary addition to our design but shouldn’t be involved with our tests of structural stability. Method C only works on objects with a slender shape, and does not provide the same level of accuracy in this case that we can afford with Method A. Method’s D and E are unnecessary because we aren’t testing a living, organic specimen and or are we measuring the properties of an “in-place element”. Lastly, Method F is meant for wood chips and our table will not be built with wood chips so much as whole pieces of wood. As such, Method A best suits our purposes and can be used to determine the density of our building materials, and consequently their mass. This information can then be used for determining strength test calculations and static structural stability calculations in our design.

4.1.3 ASTM D4442-07

As part of our project, we need to determine how much moisture has been taken in by the framework of the table because it could potentially damage the structural stability. It could also be a safety hazard because of the possibility of the moisture reaching the electrical wiring. This standard specifically concerns how to determine the moisture content of “solid wood, veneer, and other wood-base materials, including those that contain adhesives and chemical additives” (D4442-07). This will allow us to waterproof our materials for the table, then test them to determine if they have taken in excess moisture.

To specifically test whether or not our materials have taken in moisture, there are four different methods: Primary Oven-Drying Method (Method A), Secondary Oven-Drying Method (Method B), Distillation (Method C), and Other Secondary Methods (Method D). Method A is considered to be the most accurate, but is not always desired or justified to use. The process requires that the object under testing be placed in an oven and dried until no change in its weight can be detected. A calculation is then run based on the gathered information to determine the moisture content that the object originally had, but it requires the use of “fresh desiccant” which may not be available to us. Methods B and D will prove easier to implement by comparison as they use materials more available to us, but in general require the same method of measuring an object’s weight before and after drying it to determine the amount of moisture that was contained within it.

Meanwhile, Method C involves a very complicated chemistry procedure to extract water from the wood. Regardless of method, each of them has special procedures to account for the addition of chemicals to the wood when determining change in moisture, so they can be very useful to our project.

Most likely we will be using either Method B or D, depending on the supplies we can acquire. We will attempt to obtain those necessary for Method A, but they include "desiccant" and "closed weighing jars" which we may not be able to find. Method C is unnecessarily time consuming and inefficient for our purposes, so it is better that we avoid using it for our tests. With that in mind, Method B should prove sufficient for our purposes, provided that we can find all of the materials for it. If the materials are unavailable, we will look up the alternative methods described in Method D and make use of them instead. Thus, these methods should allow us to test our waterproofing of the table's framework and insure the safety and stability of its design.

4.1.4 ASTM E564-06(2012)

This standard specifically concerns how to determine the static load that a structure is able to support to prevent it from toppling over its side. Our table needs to be able to support the weight of the equipment and devices involved in its design, common cups, and possibly even the weight of people leaning against it. Therefore we will need to use tests from this standard to determine the ability of our table to resist these common forces that it will encounter in regular use. Specifically though, this test will tell us how well our table is able to resist being flipped over one of its sides.

This was specifically designed for testing walls built along-side support columns rather than the columns themselves, but should still provide useful information to determine the stability of our table. The basic test for determining the load a structure can support is determined by "anchoring the bottom edge of the wall assembly and applying a force to the top edge oriented perpendicular to the wall height dimension and parallel to the wall length dimension" (E564-06(2012)). This basic test is meant to apply the concept of "moment about an axis" to the structure in question and use the base of the structure as its axis. By keeping the "bottom edge" of the structure anchored and then applying force to the "top edge" we can determine how much load the structure can withstand without breaking along the length of its height. These tests are called "forcing a racking deformation", and a minimum of two different tests are required as part of this standard to determine a structure's stability when force is applied on either side of it.

The following is the explanation given in the standard on how to perform the test: "Racking loads shall be applied parallel to and at the top of the wall, in the central plane of the frame, using a hydraulic jack or similar loading device capable of maintaining a constant displacement rate for continuous load to failure or holding a static load in the case of incremental loading. Loads shall be applied at a constant rate of displacement to reach the target limit (that is, limiting displacement of ultimate load) in no less than 5 min" (E564-06(2012)). We specifically may not be

able to obtain a hydraulic jack and may need to use an alternative means of applying force to the structure to determine its stability. We also need to test its strength against a load situated directly on top of the table instead of against the side as a test to determine if it can sustain the load of our equipment, cups in play, and any human weight that may come into play. In any case, this standard should prove very useful to insuring the safety of our design.

5.0 Realistic Time Constraints

5.1 Economic and Time Constraints

One of the goals of this project is efficiency, so the price and time it takes to build it would need to be kept within certain parameters. The price of materials should not exceed other projects of a similar nature with too strong an difference. Beyond that, the team's ability to fund the project is limited to somewhere around \$1000. Time-wise, the project needs to be completed within the semester of Senior Design because otherwise none of the team members will be able to pass, regardless of whether this project succeeds or not.

5.2 Health and Safety Constraints

The manufacture of this table requires certain restraints be followed. Obviously, no liquids are ever to be allowed near the table until it is guaranteed to be waterproof. Care needs to be taken when working with the materials for the construction of the table to ensure no injuries are sustained from misuse of the necessary tools. Beyond this, all members should keep aware of their own personal health to ensure that it does not inhibit their ability to work. If it does, they will need to take measures to ensure that they can recover from whatever ailments they are met with in a timely manner because they can't risk hurting themselves or others by failing to account for this when they work.

5.3 Manufacturing and Sustainability Constraints

This group has limited resources for manufacturing and sustainability. They members are in contact with people that have experience with electronic and construction, but they themselves have limited experience and will be limited in this way by what they can build. There is also a limit on the sort of tools that can be accessed by this group.

5.4 User Constraints

Users cannot be expected to have the same knowledge of electronics and coding as the group members who built this project. They also can't be expected to have any tools for making adjustments to the table. Users will only be capable of average feats of physical activity as there is no reason to expect them to be able to do more. The table must be built with these matters in mind.

6.0 Hardware and Software Design Details

6.1 Final Design Architecture, Related Schematics, and Printed Circuit Board (PCB)

The schematic made for The Beer Grid required two sheets that are explained in detail in the following subsections. The first sheet was purely dedicated for the microcontroller while the second for the LED Driver and Shift Register arrays as well as some power precautions.

6.1.1 MCU Schematic Details

The microcontroller chosen for The Beer Grid required an extensive amount of components just for basic functionality. In total, an approximate 90 components were required for the creation of the microcontroller circuit for basic functionality. In order to save money and reduce the size of the printed circuit board, surface mounted devices (SMD) were chosen for the creation of the printed circuit board. 0603 SMD size was chosen due to the easiness of the soldering of the board. Below is an overall view of the schematic of the microcontroller unit:

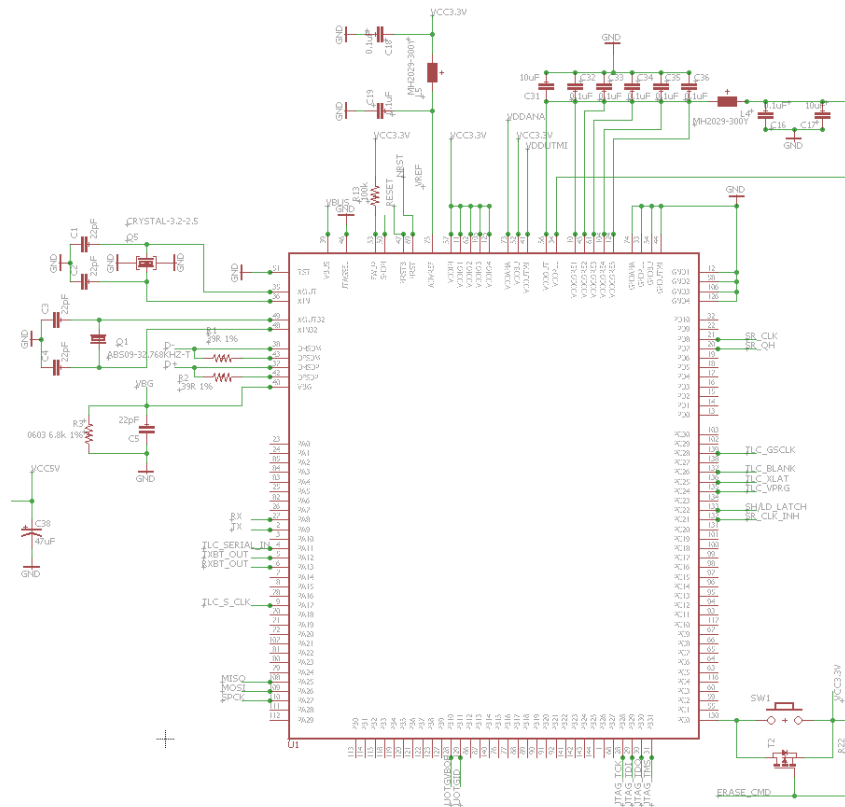


Figure 6.1.1: Overall Schematic ATSAM3X8E Microcontroller

According to the schematic checklist of the ATSAM3X family of processors it was recommended to include fuses and decoupling capacitors for every voltage pin and voltage core pin for the microcontroller. Therefore, for the desired schematic of the microcontroller it was vital to include the recommended components to prevent any malfunction given by the microcontroller as shown:

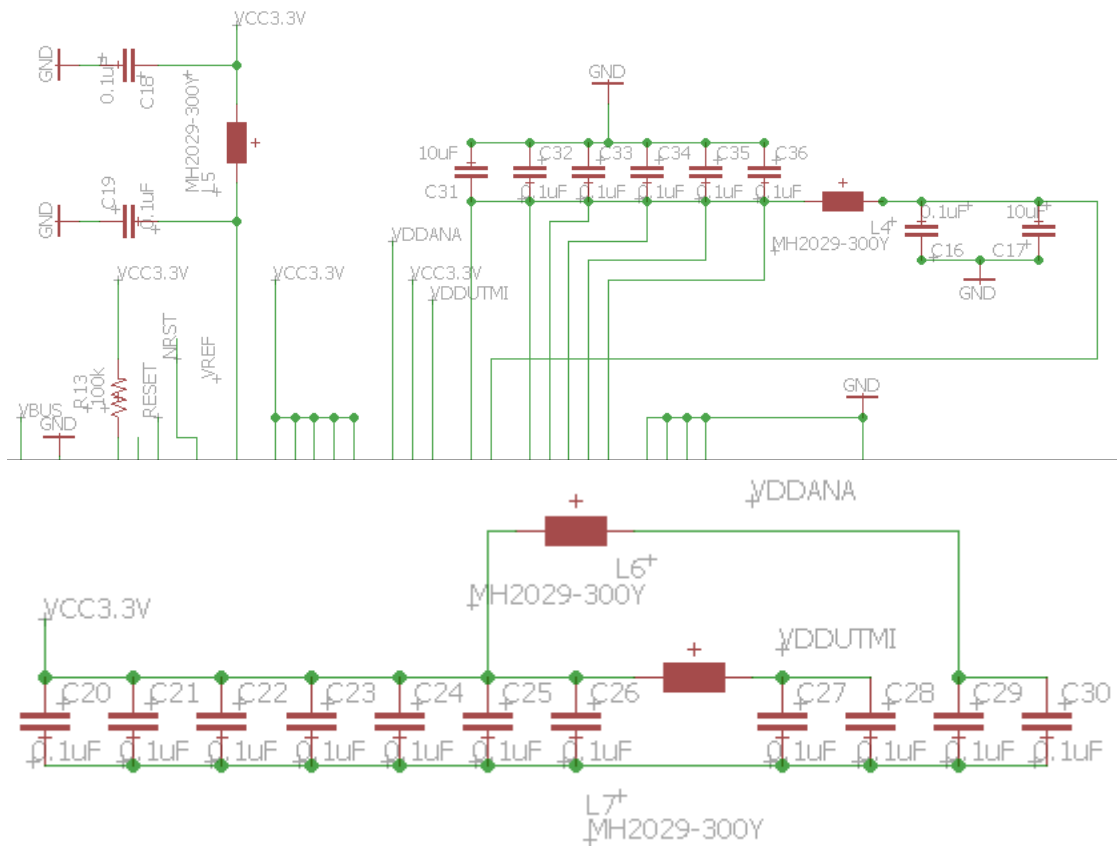


Figure 6.1.2: Closeup Schematic ATSAM3X8E Microcontroller power decoupling capacitors

Upon the research of the creation of the MCU schematic. It was additionally discovered that the ATSAM3X8E microcontroller featured a native USB port for programming. This feature resulted both useful and problematic. On one hand it allowed the team to be able to program the microcontroller directly without the need of a JTAG programmer or any special device at the cost of also including a voltage regulator and a DC to DC converter since it would connect to a USB port running at 5V. Therefore, the team decided to add the following components to accommodate the USB functionality and the components are shown as follows:

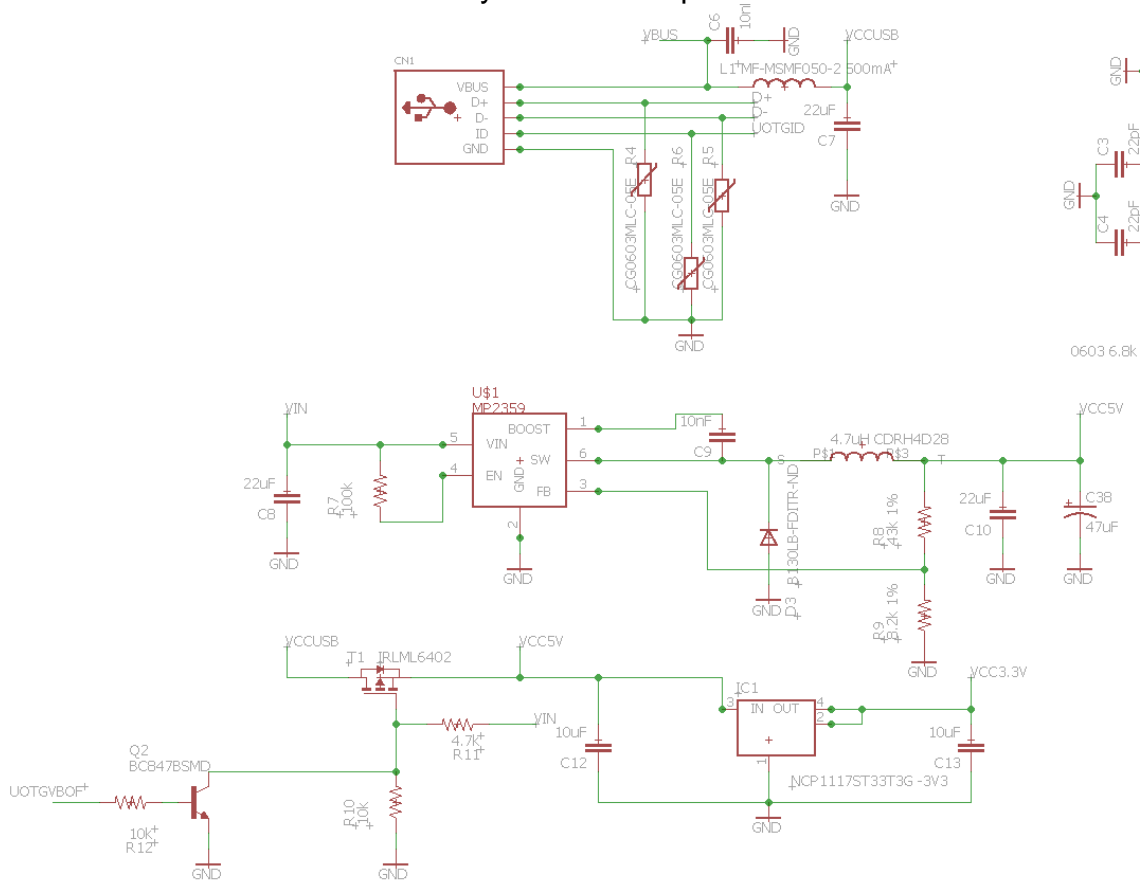


Figure 6.1.3: Closeup Schematic ATSAM3X8E Microcontroller Power components

Finally, even though the ATSAM3X8E included all the listed components above for easy programming. The team nonetheless included all the required headers for JTAG programming and even direct access to RX/TX pins in case the USB communication failed for any given reason as well as other headers to attach the Bluetooth module as shown below:

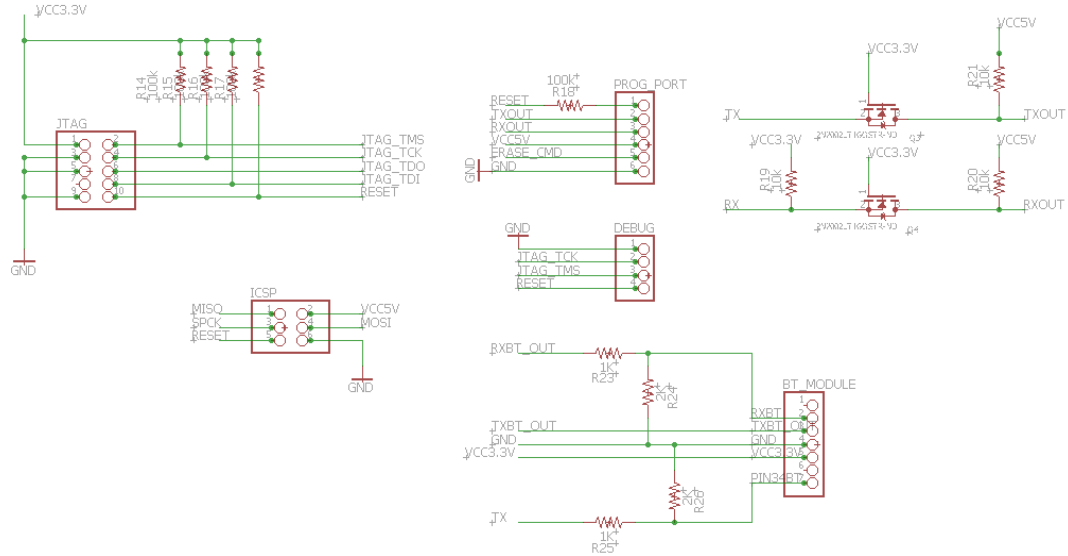


Figure 6.1.4: Closeup Schematic ATSAM3X8E Microcontroller Programming ports

6.1.2 LED Driver and Shift Register Schematic Details

For The Beer Grid, additional schematics to house other essential components were made. The most important was the creation of the schematic sheet containing the LED Drivers and shift registers. Below is the LED Driver array containing 19 TLC5940 LED drivers for 90 RGB LEDs

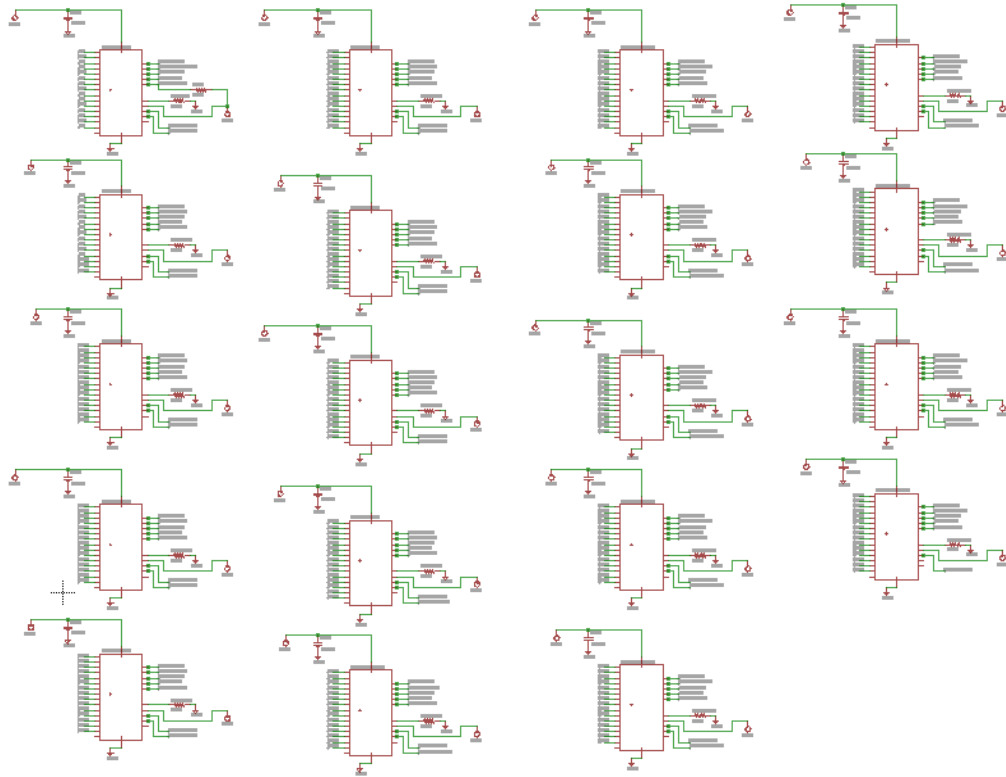


Figure 6.1.5: Schematic of TLC5940 in Daisy-chain configuration

A closer look of the LED Drivers with readable pin-outs is also included as it is essential for the sake of documentation to include all the required connections for the TLC5940s in cascaded or daisy chain configuration.

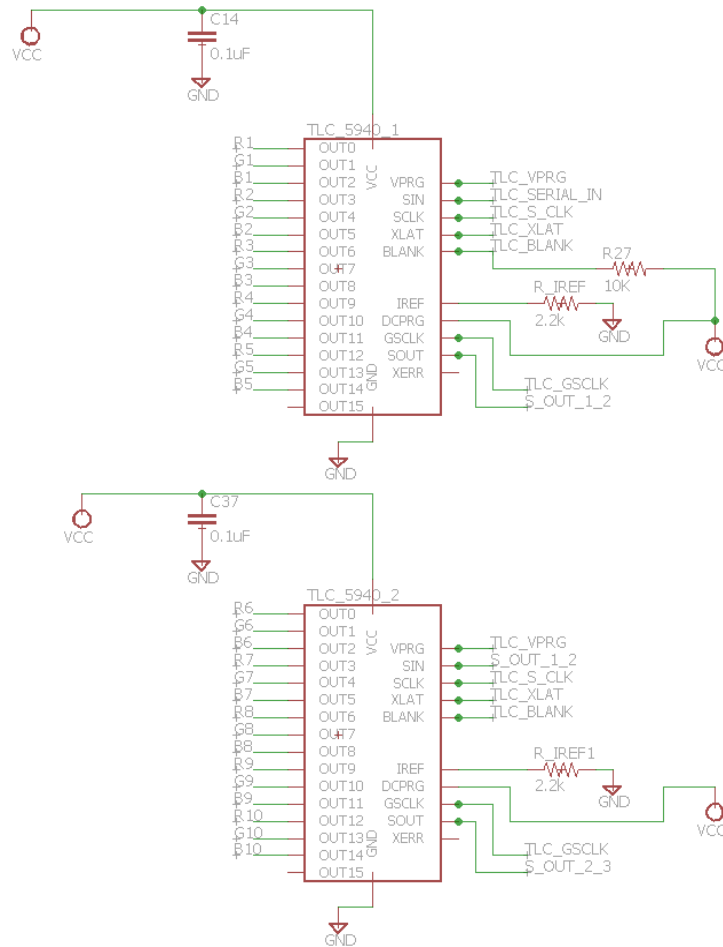


Figure 6.1.6: Closeup schematic of TLC5940 in cascaded configuration

The shift registers were also required in order for the microcontroller to receive data from the infrared proximity sensors. In this case, 19 shift registers were required in cascaded configuration as shown:

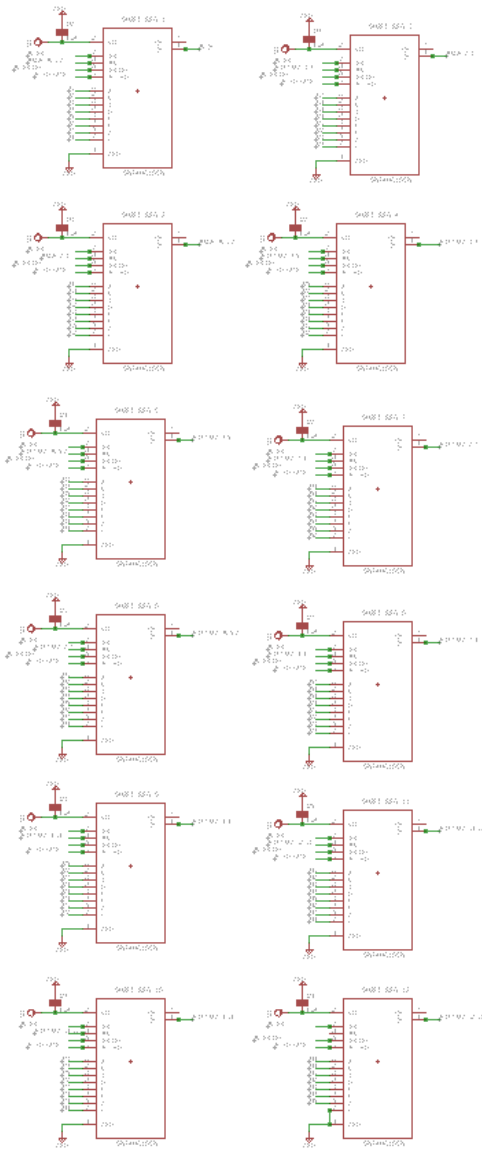


Figure 6.1.7: Schematic of SN74HC165N Shift registers in cascaded configuration

A closer look of the shift registers with readable pin-outs is also included as it is essential for the sake of documentation to include all the required connections for the SN74HC165N in cascaded or daisy chain configuration.

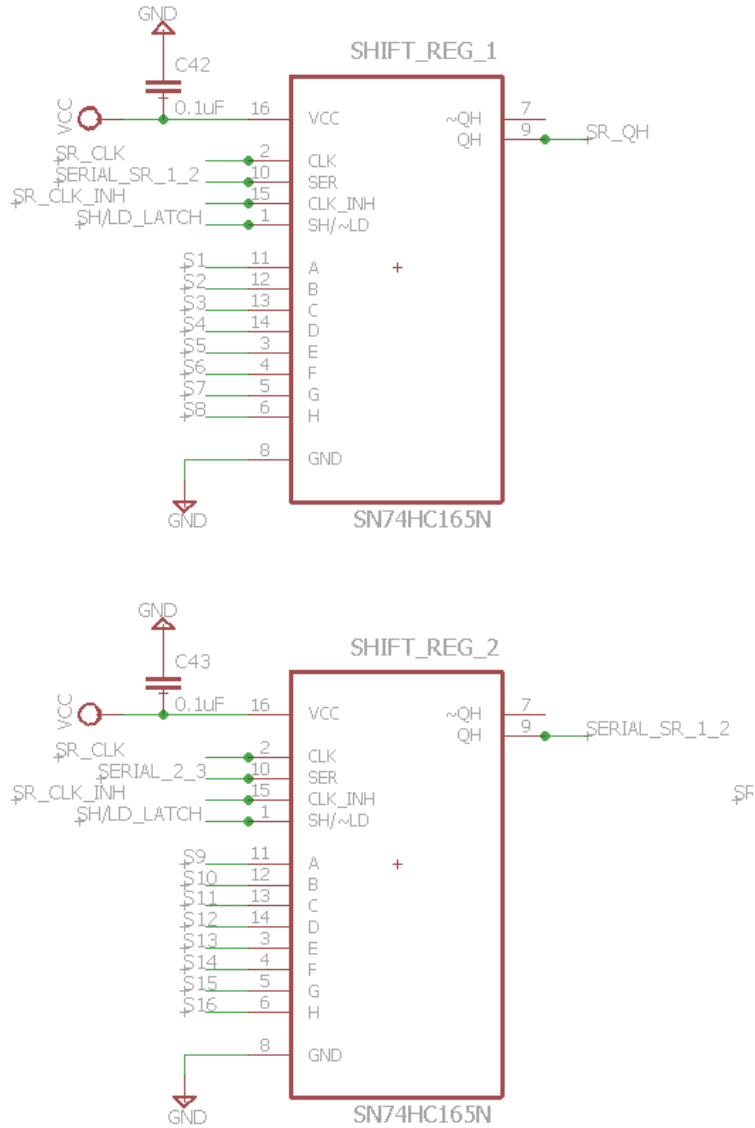


Figure 6.1.8: Closeup Schematic of SN74HC165N Shift registers in cascaded configuration

Additionally, since 90 RGB LEDs are being powered and accounted for on a single board. As recommended, bigger decoupling capacitors were required. Below is an image of the power input pins of the board as well as the recommended decoupling capacitors for optimal functionality.

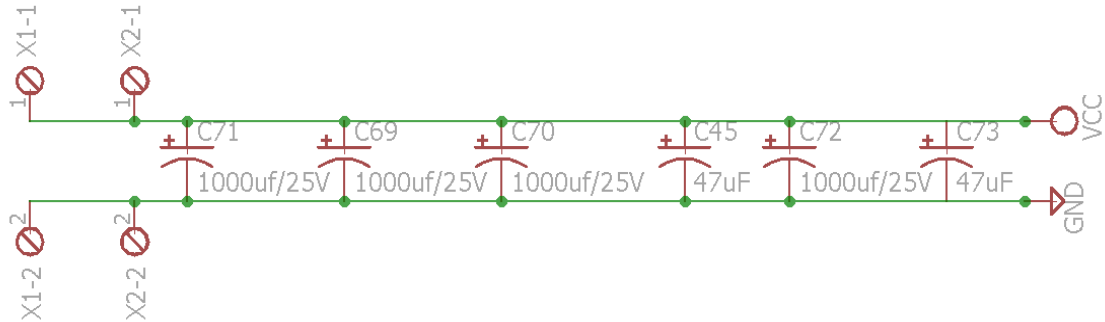


Figure 6.1.9: Schematic of RGB LED and Sensor Nodes Power

Headers for each individual RGB LED were made. In this case the team decided to break the required connections apart for each node into one 4-pin header and one 2-pin header. The 4-pin header containing 4 connections for the RGB LED and proximity sensor while the 2-pin header is for common ground and VCC as shown below.

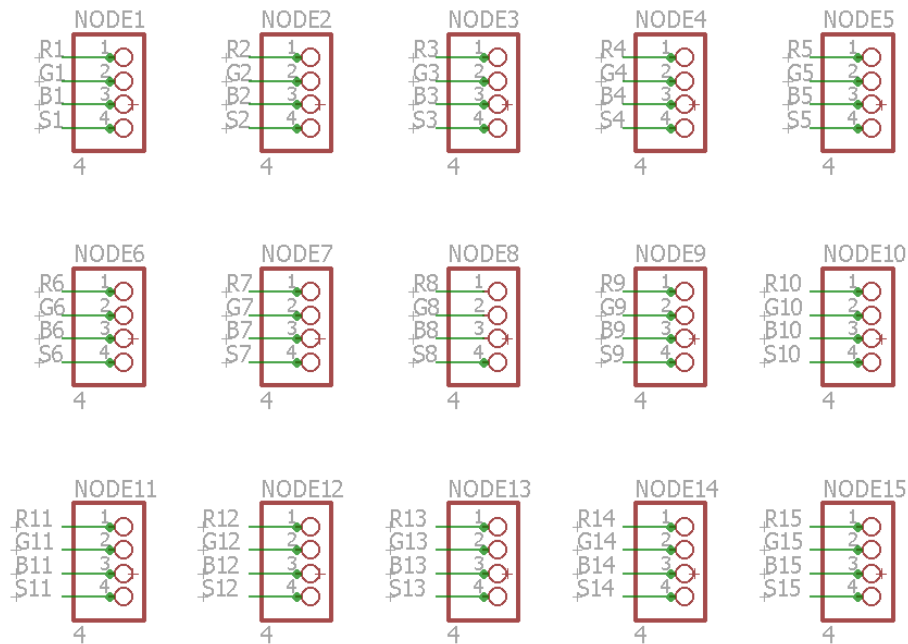


Figure 6.1.10: Schematic of RGB Nodes and Sensors Headers

6.1.3 PCB Design Details

The manufacturing of the PCB was proven to be quite complicated and the project itself required a rather large PCB. The exact measurements of the PCB are 14" by 10". In order to maintain a clean design a data bus was created to connect all of the shift registers and LED drivers which would ensure a clean logical connection between the aforementioned devices and microcontroller. A picture of the PCB as shown below

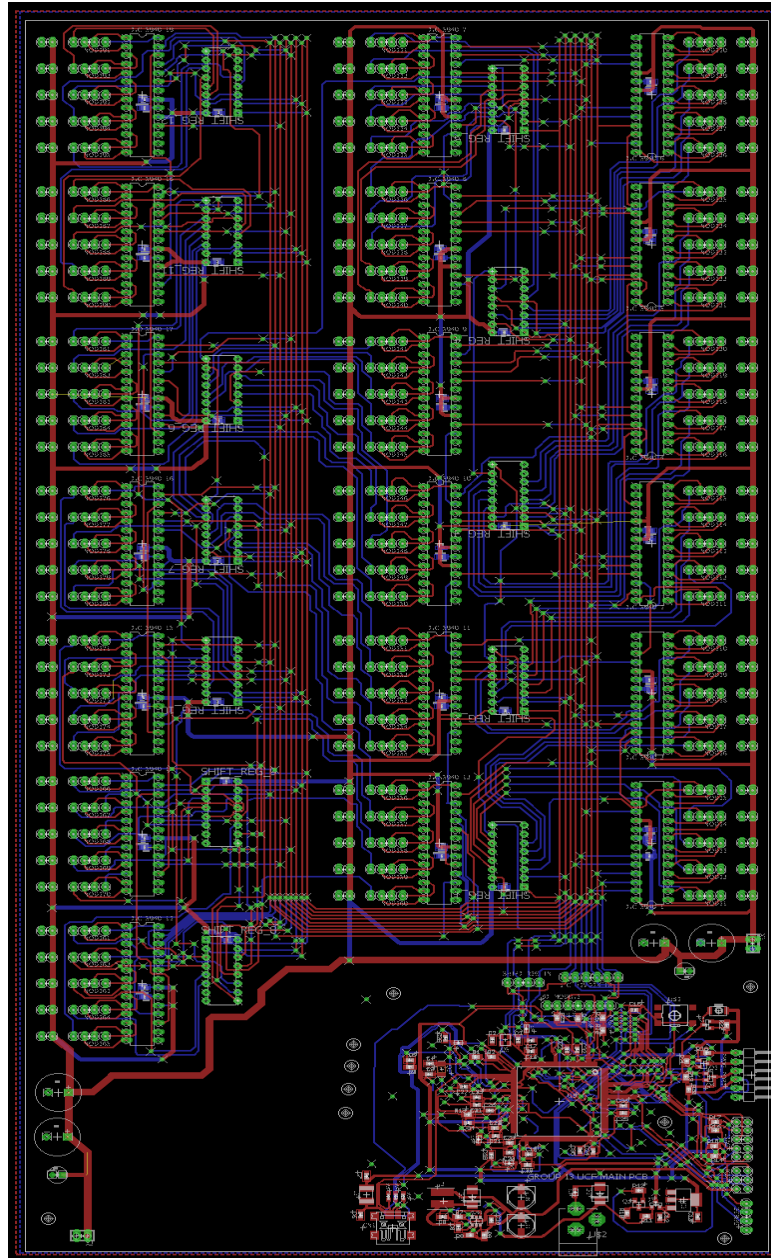


Figure 6.1.11: Final PCB Design containing all components required for project

6.2 MCU

This component is responsible for controlling all of the subsystems in our design. Without it, most of our design is rendered virtually useless. Shown below is a small diagram illustrating how information is passed throughout our system. The green lines indicate information passage, the blue line signifies wireless communication, and the red lines indicate the passage of energy to different components

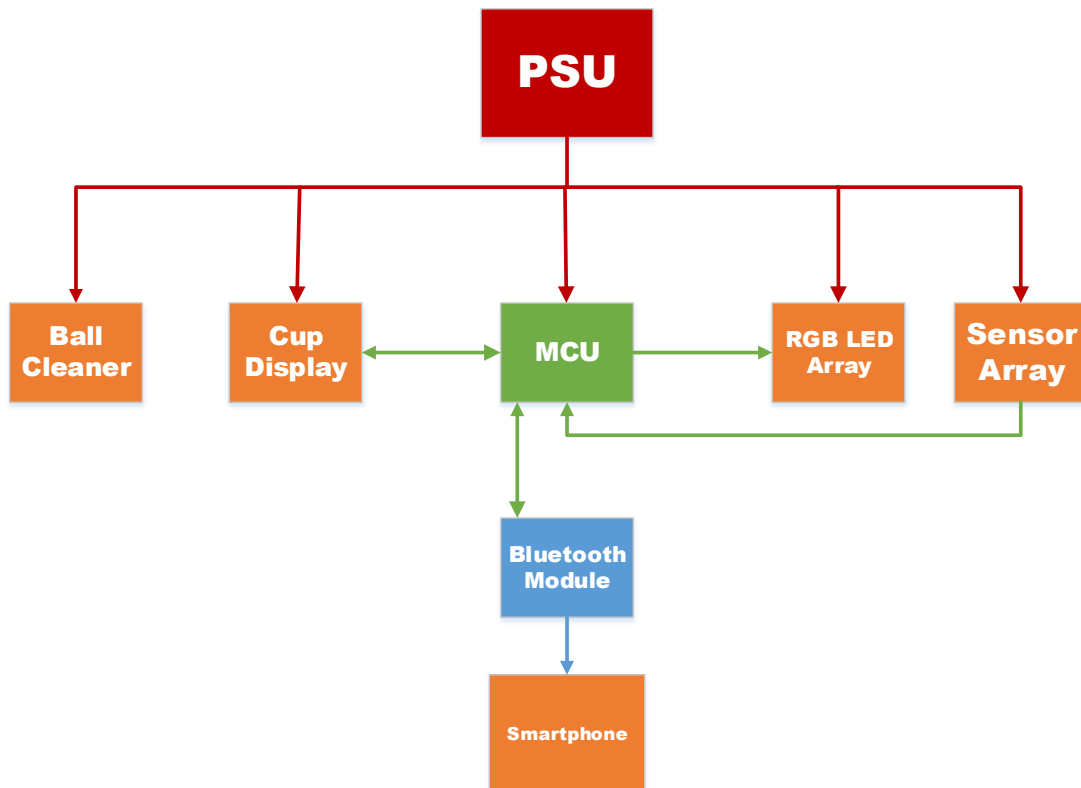


Figure 6.2.1: Block Diagram of Table System

6.2.1 Hardware

The MCU of our choice will be placed inside a custom designed PCB, then this will be connected to the different components of our system. The two main subsystems that have to be connected properly are the LED Array and the Sensor Array. The Bluetooth Module comprises a small, independent system as well that will allow wireless communications with other devices, such as those that will be running our app. Meanwhile, we also have the Cup Display system, which reuses much of the same hardware that was in the LED Array and Sensor Array, though to a much simpler degree.

To connect the MCU to the LED array, we make the appropriate connections between the first TLC5940 chip in the LED array. We need to specifically connect the “Master Out Slave In” (MOSI) pin on our MCU’s PCB to the “SIN” port on the TLC5940 chip. This is the one pin responsible for sending the signals that will program the LEDs, while the other pin connections will be used for telling the chip when it should or shouldn’t process signals and how they should be processed. We then cascade the remaining TLC5940 chips that we need to the first one, and by doing so properly we are able to control all of the chips and send independent signals to each of their output channels so that we can control any LED connected to them. This will allow us to control our complete grid of LEDs and make whatever patterns that we want.

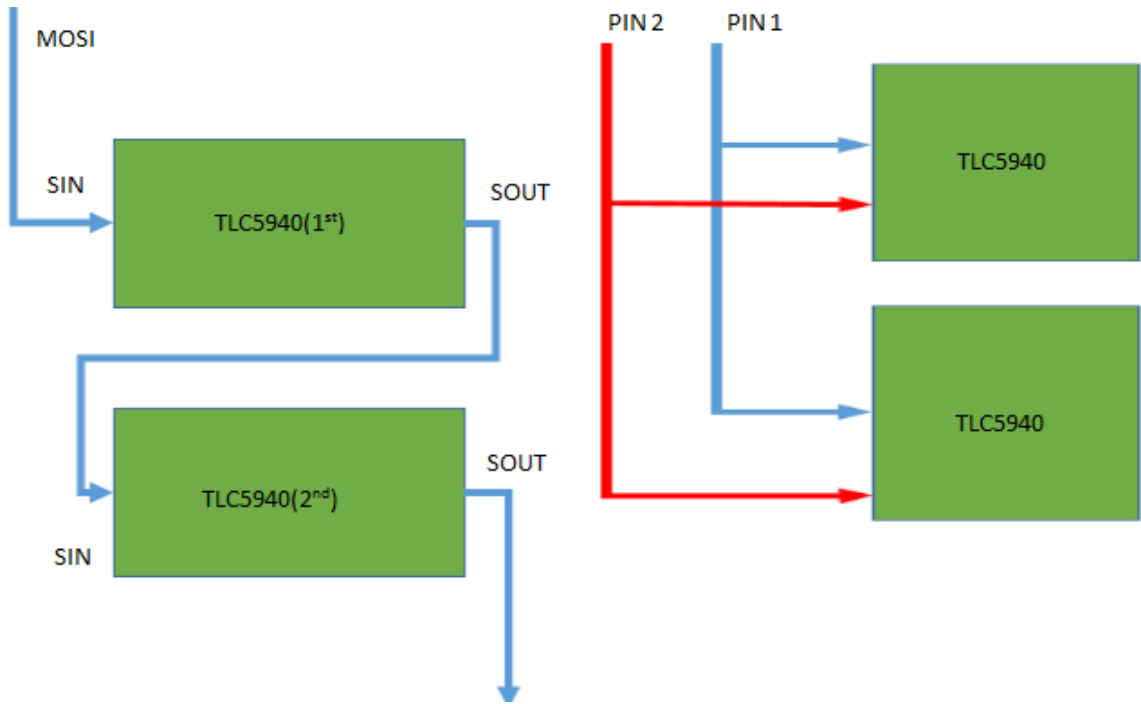


Figure 6.2.2 (Left side) and Figure 6.2.3 (Right Side): The figure on the left illustrates connections involving the SIN and SOUT ports on the TLC5940s, while the figure on the right illustrates how all other connections between the chips are made.

With regards to the Sensor Array, the setup is very similar to the LED Array. Instead of LED Driver chips, however, it will be using the SN74HC165 Shift Register chips to connect the PCB that holds the MCU to the TCRT5000L infrared sensors in the table’s design. The chips will be cascaded together in much the same manner as the LED Drivers, but instead of connecting the “Master Out Slave In” pin we are actually connecting the “Master In Slave Out” pin to the Shift Registers. This allows the MCU to receive information from the Shift Registers in the form of a stream of bits. Each of the bits corresponds to signals from the infrared sensors that the Shift Registers are attached to. Thus, assuming all signals are normally set to high, we can detect whether any of them were activated or not on the basis of whether we find a low signal instead.

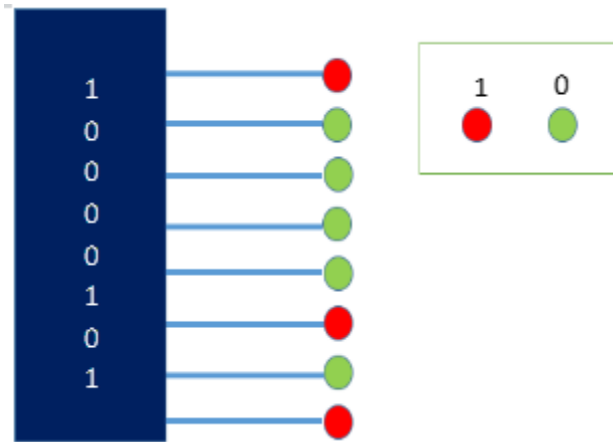


Figure 6.2.4: Illustrates the way information is collected by the Shift Register from individual infrared sensors.

Finally, the Cup Display system use the afore-mentioned technologies for the purpose of detecting when cups for games of Beer Pong are properly in place. When a cup is in place, an LED will be activated to signify that it is being detected properly. The only real difference between this system and either the LED Array or Sensor Array is the way the system will be programmed and the fact that the Cup Display system consists of only twelve different LEDs and twelve sensors because only that number of cups will be used for the table's games. Other than that, the setup is basically the same.

6.2.2 Software

The bulk of the main program needs to be given to the MCU. If the main program does not function on the MCU, it won't be usable at all. The main program needs to capable of the following: activating the table, deactivating the table, accepting customization of the table appearance, and providing information for the apps to use. It will be written is Basic C code rather than assembly code because that will allow us to organize the table's functions more easily and still be able to run on the MCU we've chosen.

The MCU will be equipped with a few default setups so that users will be able to play games on the table even if they have not tried to customize it yet. The default game options will include some combination of a set of functions that can be chosen by the user in the customization feature. These functions will include: solid color pixels, color-changing pixels, deactivating and/or reactivating pixels, and some combination of these elements to create images or effects on the table. Apart from these default modes, there will be a setting that allows the table to remain idle until such time that it receives input via the Bluetooth connectivity module. This will allow it to download a customized table setup for users to play as well.

Regardless of what customization functions are activated, some parts of the program remain the same regardless. The core of the program is able to access the system of information nodes that each correspond to one of the LEDs as well

as access input from each of the Infrared Sensors. Specifically, the Main program sends information to a set of Shift Registers which in turn send information to LED Drivers in order to trigger the LEDs. The Impact Sensors also interact with the Shift Registers in order to communicate with the rest of the table, so the MCU needs to be programmed to interact with these peripherals as well. Beyond accessing the table's display systems and sensors, the MCU needs to be able to manage the different processes that the table has within it.

The MCU needs to be able to keep track of all the input information from the sensors and coordinate the LEDs in response without the program crashing. Any coordination of the LEDs to create an effect that spans the entire table will be the responsibility of the main program to execute. These effects will include, but are not limited to, cascade effects that cause the pixels to change from one color to another by sweeping across from one side of the table to another. Such an effect can only be achieved by getting the system of LED nodes to communicate with one another. As long as the MCU is capable of carrying out processes for the purpose of triggering the LEDs and accepting information from the Sensors, as well as organizing these processes so that the table won't crash, it should fit our software requirements

6.3 LED Array

6.3.1 Hardware

To prepare the LED Array for programming, first we must make all of the necessary connections between the MCU, LED Drivers, and the LEDs themselves. The MCU connects through the LED Drivers, which in turn connect to the LEDs. The design is specifically setup this way because the MCU lacks a sufficient number of output pins in order to control the number of LEDs we have in mind without the use of the LED Drivers. The LED Drivers are also needed in order to simplify the process of activating the LEDs as each LED Driver can be used to connect and control the current and frequency of PWM used by multiple LEDs at once.

Each LED Driver has 16 output channels numbered from 0 to 15 (though we never use the 15th channel for the sake of convenience). There must be one LED connected to every 3 adjacent channels until the grid is full because each of the three bulbs in a single RGB LED requires its own channel in order to program different PWM values to each one. Doing so grants control over each LED in such a way that they can produce different colors on command. Once the LEDs are connected to their respective channels from the LED Drivers, the Drivers themselves are controlled by connecting their pins to the MCU in a specific way. Below is a diagram of the chip that illustrates its ports:

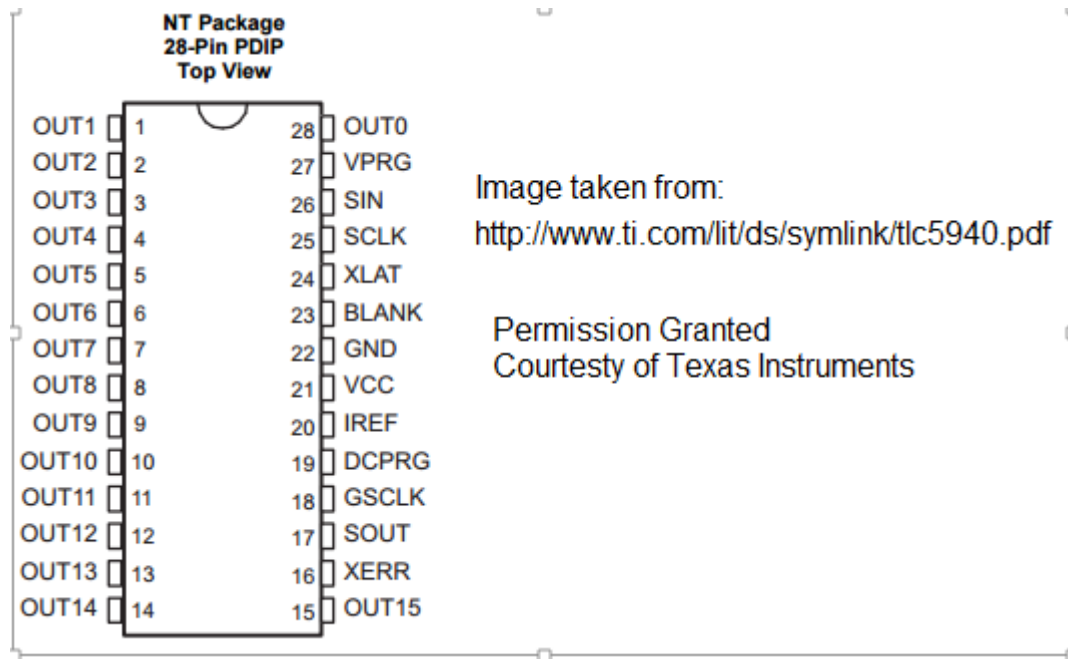


Figure 6.3.1: Illustration of the ports on the TLC5940 chip

The table needs to be able to access the “Grayscale PWM Control” mode on the chip. In order to do this, the “VPRG” pin is connected to ground directly because that is used for activating “Dot Correction” which allows for even more control over LED colors, but will not be needed for the purposes of this project. The “SIN” port is connected to a serial output pin from the PCB, while the “SCLK” port is connected to a “SPI CLK” pin from the PCB. This allows for serial communication with the chips as the “SIN” pin receives signals to send to the LEDs and the “SCLK” port receives signals that tell the chip when to switch from one channel to the next. The “XLAT” pin tells the chip when to change data and when to hold it constant, and the “BLANK” pin is used to deactivate all output channels and reset the GS counter. The “GND” port and “VCC” port obviously connect to ground and a 3.3v power source respectively, and the “IREF” pin is run through a 2.2k resistor to ground because it is responsible for regulating the current that goes through each of the channels. Running it through that particular resistance provides the resulting current that is needed to drive the LEDs properly. The “DCPRG” pin is set to 3.3 volts directly because that is required to access “Grayscale PWM Control”, and the “GSCLK” port needs a pin attached to it so that a clock of 4095 values can be referenced for programming the PWM values on each port. Last but not least, the “SOUT” port is connected to the “SIN” port of the next chip in the series to cascade them, while all other ports are connected in parallel to the PCB in the same manner as how they are connected to the first chip in the series. Once completed, this hardware setup grants full control over programming the LED.

6.3.2 Software

The Array of LEDs will be programmed using a system of class variables that will function as nodes of information on each LED. The nodes will hold the following

information about each LED: current color, a variable to signify the LED is active, a variable to signify when the LED is blinking, a unique number to distinguish that LED, and four pointers to reference adjacent nodes. While the LED nodes are used to hold the current state information of each LED in the array, they will be referenced by Pattern Classes that will be used to adjust the information in the nodes as well as the actual current state of the table. There are several different Pattern Classes to activate different effects on the table. There is a “Fade Color” class, a “Switch Up” class, a “Rip” class, and a simple “activateColor” and “deactivateColor” class. Below is a diagram of how the “LedNode” class is related to the various Pattern Classes:

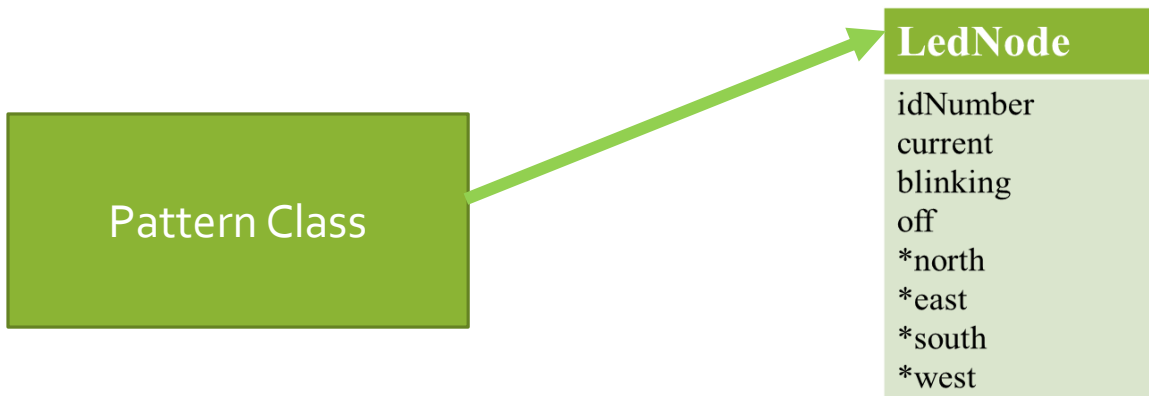


Figure 6.3.2: Illustration of the relationship between the code for LedNodes and the Pattern Classes that will change them.

To program the different effects into their individual classes, the main program makes use of a library of functions that were made specifically to work on the MCU being used in the table’s PCB. This library includes a simple function called “setGSDData()” which can be used to choose a specific channel from the chain of TLC5940 chips and decide what PWM value will be used on that channel. There is also the “sendGSDData()” function that lets the TLC5940 chips be update with the newly set changes. With some creative programming, this allows for full control over the LEDs to implement any desired effect.

For the sake of making the Pattern Classes easier to build, there are several variables and smaller functions that were built to make accessing the LEDs and managing their effects easier. One such tool is an enumerated type called “COLOR” which holds the following values: BLUE, GREEN, RED, YELLOW, PINK, PURPLE, ORANGE, WHITE, and BLACK. The variable within the LedNode that tracks the current color displayed by that LED is based on this enumerated type. It is evaluated in a few of the other program functions for various purposes. There is also a static COLOR variable called “library” which is used as an array containing each of the different values of the enumerated type, COLOR, in its slots. This variable is sometimes referenced by pattern classes to compare the current COLOR value of the LedNode and determine its value. Meanwhile, a small class titled “initializeN” is used for the purpose of setting initial values of each LedNode before they will be used for actual operations in the table. Additionally, there is a function called “channelCheck” that is used to account for the fact that channel 15

of each TLC5940 chip is never used in the circuitry. Normally the channels corresponding to a particular LED are found by multiplying their LedNode "idNumber" by 3, but since channel 15 of every chip is skipped, the later channels are thrown off. The "channelCheck" function determines how many channels have been skipped by the LED in question and calculates the proper value of the corresponding channels to that LED based on that information.

Other than these tools, there are three different grids of "int" variables called "B", "G" and "R". Each grid has 9 rows, but as many columns as there are LEDs. In the case of the full table design, there are 80 columns for the 80 different LEDs. Each of these grids store PWM values so that they can be referenced by classes that need to activate the different LED colors. Sometimes the LEDs don't produce a uniform output of color in spite of the fact that they may have the exact same PWM values. To deal with this, these three grids hold the PWM values needed by each of the 3 different channels for each LED to display a unique color. For example, when an LED needs to display the color Blue, there are 3 channels that have to be set for that particular LED: the blue bulbs channel, the green bulbs channel, and the red bulbs channel. To set the blue bulbs channel, go down the rows of the "B" grid to the value of the corresponding color. Remember, the colors are all given a unique number, and number for the color Blue is 0. Therefore the information for that color is stored in row 0, the very first row. Then the column with the same number as the LedNode's "idNumber" is accessed. If the "idNumber" was "0" it would be the first column on that row, and if it was "5" it would be in the sixth column from the left side of the grid. This same method is used to find the PWM values for the green bulb and the red bulb using grids "G" and "R" respectively. This way the unique PWM value for each LED can be stored and accessed easily based on the color that LED is meant to display and its unique "idNumber". There are other smaller classes like these, but they pertain to the programming of the Sensor Array, so they will be discussed later on.

The simplest of Pattern Classes is the "activateColor()" function and the "deactivateColor()" function. The "activateColor()" function takes in a specific LedNode variable corresponding to a specific LED on the table as well as a number signifying the value of one of the 9 different colors that have been programmed. The colors are numbered from 0 to 8 in the following order: Blue, Green, Red, Yellow, Pink, Purple, Orange, White, and Black. With this information, the "activateColor()" determines which of the rows from the "B", "G", and "R" grids to access and then moves over the appropriate number of columns in each to access the unique PWM values for the LED corresponding to that LedNode to use. This information is then combined with the function "setGSData" from the library made for the TLC5940. Once the information is set, it the table acts on these changes with the "sendGSData" function, which is also from the library made for the TLC5940. Also, the variable "off" will be switched to 0 if it has not already been so to indicate that the LED has been activated. On the other hand, the "deactivateColor" class merely needs to take in a specific LedNode and set all of the channels for the corresponding LED to 0 PWM as well as set the "off" variable to 1 to indicate that the LED is inactive. It is important to note that the status of

“current” in the LedNode will not be changed by the “deactivateColor” class so that if the LED is reactivated, it merely needs to reference the same information to display again.

The two functions mentioned earlier, “activateColor()” and “deactivateColor”, are both used by several other classes within the code to make operating the LEDs more efficient. Among these classes is the SwitchUp class that enables the LEDs to activate and deactivate in response to sensor input. This code actually works in conjunction with classes designed to interface with the infrared sensors in the table, but the specifics of how will be discussed in a later section of this document. When the information from the specific sensor that has been paired with a particular LED has been sent to the MCU, that is then processed as a signal to activate the SwitchUp class. This class is designed to use the “activateColor” function on the corresponding LED to the sensor when that LED is inactive. Conversely, it causes the corresponding LED to the sensor to use the “deactivateColor” function when the LED is active. In this way the table can interact directly with a game of Beer Pong by showing when the ball has been sensed by the infrared sensors or not. Games could even be made to create images on the grid of LEDs by hitting specific points on the table and getting them to react with this class. The class is compatible with any of the 9 different colors (including BLACK though that wouldn’t be obvious to a user) and requires only that the specific LED it will be used on is taken in when the function is called. It then relies on the information within the corresponding LedNode to determine how it will react. An LedNode with the variable “off” will cause its corresponding LED to activate when “off” is equal to 1. When “off” is equal to 0, then the LED deactivates in response to the SwitchUp function. Additionally, the value of “off” is switched to reflect the change made to the LED, so if it was originally 1 it will be 0 after the function is run and vice versa.

Another, more complicated class, is the FadeColor class. This class takes in the value of a specific LedNode and an “int” variable that represents one of the nine different colors that have been programmed. The class is designed to slowly change the color of an LED from its current color to the color specified when the function is run. To do this, first the three PWM values that are currently used by the LED are recorded. Then, the PWM values for the color that the LED will change to are recorded. Afterwards, the function checks whether the two sets of value match or not. As long as they are not equal, then the function will slowly increment or decrement the values of the LED until they match up to those of the color that was input into the function. There is a small delay put into each iteration of the “while” loop used by the function so that the change in color is visible to users, and after the values have been matched, properties from the corresponding LedNode are altered to record the new changes. Specifically, the value of the variable “current” in the LedNode will be changed to the color input into the function. If that color is BLACK then the property “off” will be set to the value of 1 as well.

A useful class for debugging purposes is the NodeFire class, which is simply used to flash an LED to a specified color. Some of the earliest tests of the table would

make use of a similar function, so it was included in the final version of the code to make sure that the option was always available. In terms of debugging, it allows for a quick and reliable method for activating LEDs on the grid to test that proper connections are in place. As an effect for the table, it can create what is often referred to as the “KnightRider” effect where a line of LEDs light up one after the other down the line and then back again. It is a simple function that can be use with any of the nine colors programmed onto the table, and could be used in a variety of displays for different game modes. It was also very good practice to build this so that a more advanced effect could be designed in the form of the “Rip” class.

Probably the most advanced function in the code is the “Rip” function, which can be used to create a ripple effect across the surface of the table with use of the reference pointers stored in each LedNode. The program is designed to intake the value of a particular LedNode and an “int” variable that represents one of the nine different colors programmed into the table. The function begins by activating the specified LED, waiting a short delay, then deactivating the same LED. Then, the program uses the reference pointers that have been built into the LedNode to determine what the adjacent LEDs on the grid are. It then activates all adjacent LEDs, waits a short delay, then deactivates them. It then goes on to use the pointers available in the LedNodes of the adjacent LEDs to find the LEDs surrounding them next. Then these are activated and deactivated in the same manner described above. The result is the appearance of a ripple-like effect on the surface of the table. This function can be specified to work with any of the nine different colors programmed into the table and makes the most intensive use of the dynamic memory allocation methods built into the table’s code.

With all of these effects programmed and ready to be used, the main program loop merely needs to initiate the necessary variables into play and call the functions accordingly. The main loop of the code is responsible for spawning all of the necessary LedNodes into a double pointer so that they can be accessed the same way to access a grid of variables. Afterwards, the initial settings of these nodes are activated using the “initializeN” function. There is also a small “for” loop that makes all of the necessary connections between LedNodes and their reference pointers so that they are available later on. Once these are in place, game modes can be made by creating a “while” loop that will run any functions repeatedly until an ending condition is met. It is necessary to run games this way because if the main loop ends, all of the information will be reset and a game will not be able to work properly. It is important to note that before the main loop ends, all the memory used by the LedNodes is set free. This is important since they rely on dynamic memory allocation to be used at all.

6.4 Impact Sensor Array

6.4.1 Hardware Details

The Impact Sensor array consists of about 80 infrared sensors connected to about 10 Shift Registers. These are in turn connected to the PCB so that they can be controlled by the MCU. The specific infrared sensors used are TCRT5000L sensors that can be programmed to detect objects at up to 15mm away. Meanwhile, the Shift Registers used are SN74HC165 Shift Registers. By using the Shift Register to connect to the infrared sensors instead of connecting to them directly, it is possible to control up to 80 infrared sensors or more with a limited number of output pins from the PCB. This is because the Shift Registers can be cascaded from one to another in much the same way that the LED Drivers in the LED Array are.

The Shift Registers are designed to transmit a stream of bits to indicate whether they have received information from the components on their input channels or not. As a direct result, the sensors cannot be set to more precise settings on what distance we will and won't accept data, however this can easily be remedied by testing the distance where they react and send information. Afterwards, they can be set into the table at the appropriate distance from the surface.

Each Shift Register has 8 input pins that can each be used to accept information from an individual sensor. As such, about 10 of them are needed to connect to the 80 sensors needed to complete the grid. To connect and control the sensors through the Shift Registers, each of the sensors needs to be connected to ground in a specific way while also being connected to the Shift Register so that information from them can be detected properly. Meanwhile, the Shift Registers need to be connected to the PCB so that they can in turn communicate properly with the MCU. Below is a diagram of a Shift Register:

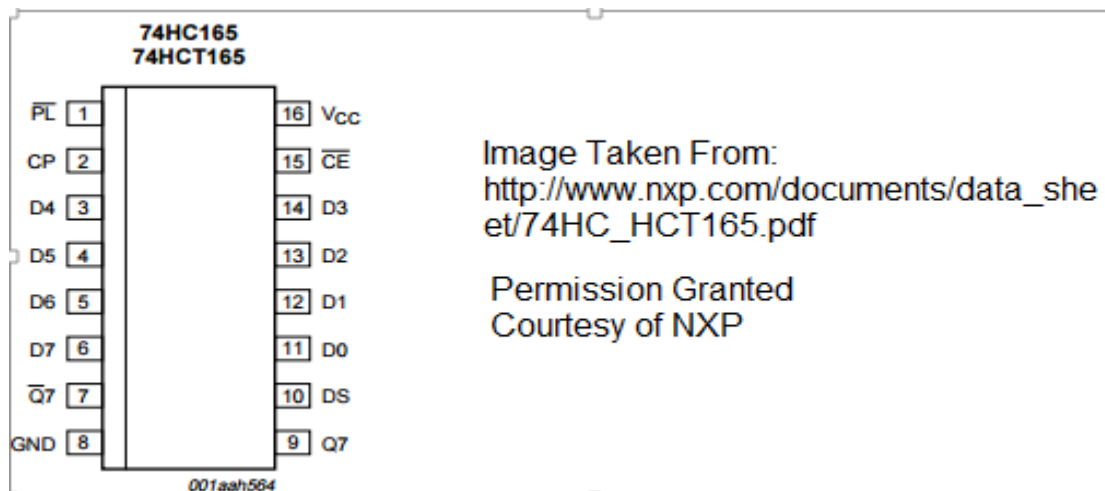


Figure 6.4.1: Illustration of the port locations on an SN74HC165 Shift Register

The specific port from this Shift Register that need to be connected are the “PL” port, the “CE” port, the “Q7” port on the bottom right, the “CP”, the “DS” port, and the “Vcc” and “GND” ports. “Vcc” is connected to the power source, while “GND” is connected to ground. The “PL” port loads information from the input pins when it is set to low, so we need it connected to the PCB to decide when to accept that information. The “CE” port is used for enabling all other pins than the “Q7” port when they are needed. The “Q7” port is used for sending information to the PCB, and the “CP” port is pulsed regularly to reflect any changes made to the other ports. If the “CP” port is never pulsed, then all other signals are rendered useless. Additionally, when the chips are connected in series, one chip receives information from the next one in series by connecting its “DS” port to the “Q7” port of the next chip in line. Doing this lets multiple Shift Registers be connected and controlled using a limited number of pins from the PCB.

Meanwhile, the infrared sensors need to be connected to the input channels of the Shift Register in a specific way. The concern when connecting them is to make sure that the appropriate amount of current is sent through the infrared emitter and the infrared receiver individually, that both the emitter and receiver are grounded properly, and the sensor is properly connected to the Shift Registers so that data from the sensor can be transmitted. Below is a diagram of how the TCRT5000L needs to be connected:

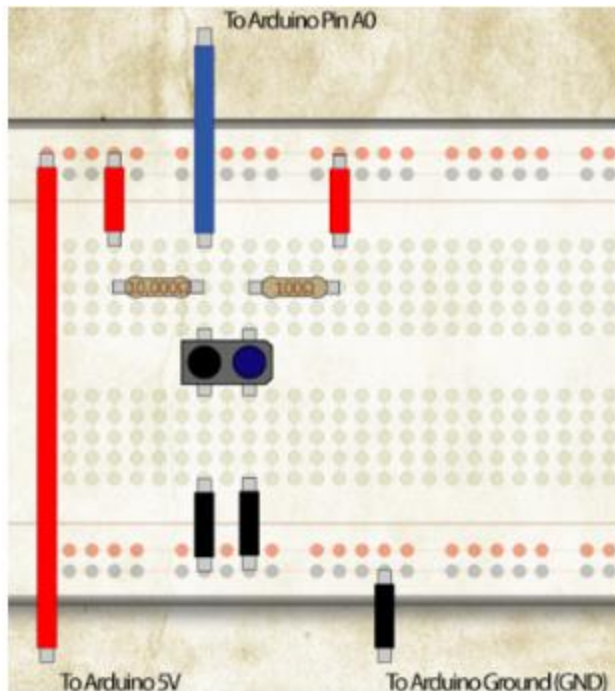


Image Taken From:
<http://blog.huntgang.com/2014/06/17/arduino-tcrt5000-build-ir-sensor/>

Permission Granted
 Courtesy of Dave Hunt

Figure 6.4.2: Illustration of how the TCRT5000L Infrared Sensor needs to be set up. NOTE: the data line is not connected to “Arduino Pin A0” in this project. That is simply how it was connected in the project from which this image was taken.

The diagram shows the lines to VCC in red, the ground lines in black, and the line to transmit data in blue. There needs to be a ten-thousand ohm resistor between the infrared receiver and its power source, while the infrared emitter needs a one-hundred ohm resistor between it and the power source. The pins on the opposite

side of the ones used to power the sensor are set to ground, and the data line for transmitting signals to the Shift Register are connected to the channel that has been set aside for that particular sensor. Once connected, information about what sensor has been activated can easily be received through the Shift Registers and the program running on the MCU can easily use this information as needed.

6.4.2 Software Details

The Impact Sensor Array is fairly simple to program and integrate into the table's systems. The Shift Registers are set to transmit a stream of ones whenever the sensors are inactive. When a sensor is activated, it will set the corresponding bit from the stream into a zero instead of a one. The program for the table will then use that information to trigger the appropriate functions. To implement this, all that really needs to be done is to create a variable that can hold the full bit-stream. Any time that the sensors cause the information to change, the variable can be read by the main program to trigger any corresponding changes in the table's display. At times it is necessary to create specific conditional statements in the code that will decide when the sensor information is accepted or not. There may be patterns implemented that leave some of the LEDs to display a constant color or pattern state that ignores input from the sensors while other continue to respond to them normally. Overall though, programming the sensors is just about making their information available to the LEDs when appropriate to do so.

There are specific tools in the code that are used for managing data from the sensors. One important note to make is the "BYTES_VAL_T" variable type that is used for storing the information taken from the sensors. There are two important variables that use this type: "pinValues", and "oldPinValues". The basic method of reading information from the sensors is dependent on comparing these two variables. Whenever information is taken in from the Shift Register, it is stored in the variable "pinValues". If this value is different from "oldPinValues" then it is immediately obvious that information from the sensors had been taken in and the main program needs to react to this change. Later, the value of "oldPinValues" will be set equivalent to "pinValues", so the next time that the state of the sensors changes it will be evident and the program can decide whether or not to respond to that change. In most cases, the state of the sensors will switch from none of them being triggered, to just one of them being triggered, to none of them being triggered again. These states are all sensed by the table program, but the table will only react as needed to for the sake of its game modes.

The specific function for taking information from the Shift Registers is called "read_shift-regs()" and has the return-type of "BYTES_VAL_T". This function shifts in the bits from the input channels of the Shift Registers and then stores them into the variable that has been made for the purpose of holding them. In this instance, that variable will be "pinValues". Later, to help keep track of the information when debugging code, this information is displayed to the Serial Monitor using another function called "display_pin_values()". This function displays the current state information of the infrared sensors as "HIGH" or "LOW" depending on whether the

sensors have been triggered or not, where activated sensors are displayed as “LOW” and inactive sensors are displayed as “HIGH”. This information is very important to access at will, so even if users will not be using the serial monitor themselves, it will remain as part of the code to allow for future modifications.

One of the specific tools in the code that is used for processing the information taken in from the sensors to determine which sensor was activated out of all of them. This specifically is the “pinCheck” function that takes in two “int” variables. One is the current value of “pinValues” in the form of an “int” variable. The other is a value that will be subtracted from the maximum possible value of “pinValues”. The resulting difference is then saved and compared to the current value of “pinValues”. If the current value is found equal to the calculated difference, then it means that a particular sensor was activated. This is because the value entered for the sake of being subtracted is always equal to an amount that gets subtracted from the maximum value of “pinValues” whenever a sensor is activated. The “pinCheck” function then returns true if the current value is found equal to the difference, or false if the value is not equal.

6.5 Cup Display System

This system is a simple display that bothers to detect whether or not the cups for the game are in place or not. It uses the same technology from both the LED Array and the Sensor Array, where it uses infrared sensors at specific points in the table to detect when cups are placed there, then react by lighting LEDs in the same location. They do not require much programming, unlike the table pixels which will need to be programmed with different effects. The sensors and lights will need to be arranged according to how the cups are supposed to be positioned according to the game rules. At most, we will program the sensors to detect when the cups are in place and light up to indicate that they are so. We might also program a function to change the color of the lights, but it shouldn't be one of the more labor intensive parts of our design.

6.6 Ball Cleaner

The ball cleaner consists of four different components total not including the microcontroller. The first component is the sensors used to detect the ping pong ball. The sensor used in this design is the Q12AB6FF50 made by Banner engineering. The reason we used this instead of the sensor used for the grid and cup sensors is because the infrared beam is over twice the length. The sensor also has screw holes making it much easier to mount. The sensor also has a coating with four color coated wires coming from it that is approximately three feet long allowing us to not have to solder them. The wires for this sensor being used include a brown wire connected to voltage supply of 10-30 volts. The blue wire connected to the sensor is connected to ground and the white wire is the data wire connected to the relay. The second component used is a motorized fan. We wanted enough air flow to be able to use pwm to lower and raise the speed of the motor to control CFM. We tried using several computer fans at first

but they were not strong enough so I used looked into Bilge Blowers. We found several bilge blowers at Seachoice and decided to use the 41851 Bilge Blower. We chose this one because of its size and voltage and amperage rating. At full power it can run up to 4.5 amps but using pwm we don't plan on using it to its full extent. The third component we used is an automotive relay used to handle the higher amperage and voltage instead of the microcontroller. It is easy to find relays at car shops such as Autozone or electronic stores like Skycraft. The relays we used had a maximum voltage of 14 volts and a maximum current of 1 amp. Since we lowered the power to the fan these relays worked perfectly for the design. The fourth component used is the PVC piping. The PVC pipe used is inch and a half in diameter allowing just a small fraction of space between the ping pong ball and its inside walls. The PVC consisted of 5 pieces allowing the ball to travel from one end to the other. To allow the entry opening to be as close to the side as possible we used a T shape PVC pipe allowing the fan to run along the side of the table instead of width ways. The T shaped PVC piece has the motor connected at one side end, the other side end is connected to a 90 degree angle, and the top of the T shape PVC piece leads to the top of the table allowing the ball to enter the tubing system. The 90 degree piece attached to the T shape the leads to a longer fitting that runs width wise along the table from one side to the other. At the end of this piece another 90 degree PVC piece is attached leading to the exit part of the tubing at the top of the table.

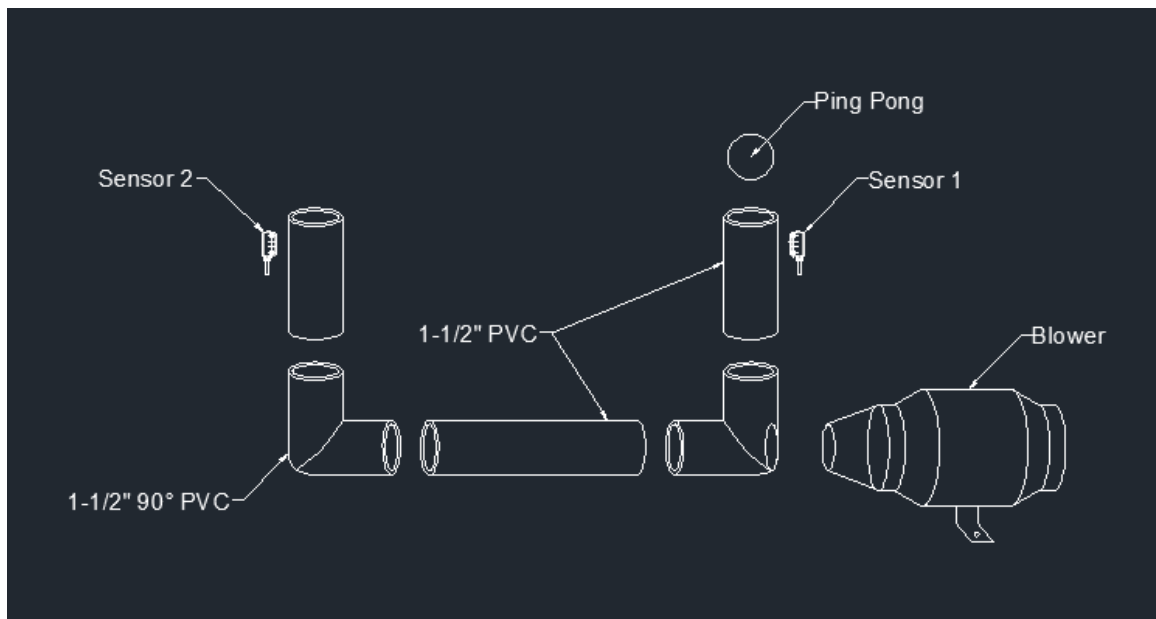


Figure 6.6.1: Basic tubing system allowing the ball to enter one end of the table and exit the other side excluding the T shaped PVC part.

The way the Ball cleaner works uses a system of outputs and inputs controlled by the microcontroller. Using the diagram below we can explain the way the ball

washer works. Sensor 1 is placed at the entrance of the tubing in the middle of the T PVC piece. The sensor is connected to ground and 12 volts and when the sensors infrared beam is broken or senses a change the white wire connected to the relay is grounded. Relay 1's coil is connected to 12 volts also and the other end of the coil is connected to the white wire from the sensor. Thus when the sensor grounds the white wire, voltage is allowed to flow which charges the coil in the relay. When the relays coil is charged, the contacts inside of it change. The relays have three contacts that include normally open (NO), normally closed (NC), and Common (COM). The relay's common is connected to normally closed when the coil is not charged and when the coil is charged the common switches to normally open. Thus when the sensor grounds the coil, the common switches to normally open. The common is connected to ground and normally closed is open so when the coil is not charged nothing will happen. The normally open contact is connected to a digital pin that is coded to act as a pulldown resistor. When the normally open contact is open the pin has 5 volts keeping the pin at high logic. When the common connects to the normally open contact the pin is grounded changes logic. The microcontroller has internal resistors used to pull voltage from the pin. Thus when the pin is grounded a voltage stops running through changing the pin logic state from High to Low hence the term pulldown resistor.

The motor also uses a relay since it also uses 12 volts and a higher amperage than what we want running through the microcontroller. The output pin is connected to a TIP120 transistor allowing pwm to control the flow of power from the fan. The base is connected to the digital pin, the emitter to ground, and collector to the negative coil terminal. When sensor 1 detects motion the output is triggered allowing 5 volts from the microcontroller digital pin to the transistor. When the transistor has voltage run to the base it allows voltage to flow from the collector to the emitter. Thus the coil is grounded through this process. The positive coil terminal is connected to 12 volts allowing the coil to charge when the output is triggered. Using the same structure as the sensor 1 relay we connect common to ground again and normally open to the motors negative terminal. The positive terminal of the motor connects to 12 volts so when the contact switches to normally open the motor is grounded allowing the flow of voltage turning the motor on. The second sensor is wired the same way as sensor 1 but is coded differently.

The coding used for this design uses one input for each sensor and one output for the motors. Thus we have a total of four inputs and two outputs. When sensor 1 has no detection it keeps the output at Low which doesn't allow 5 volts to the transistor. When sensor 1 changes from High to Low it also changes the output to High allowing 5 volts to the transistor turning the motor on. A delay of 250 ms is set after the sensor logic changes to stop the motor from turning on too quickly pushing the ball out the entrance. Once the motor is turned on another delay of 5 seconds is provided to allow the ping pong ball to travel to the exit end. Once it reaches the exit end the second sensor detects the ball and keeps the output

running at High logic keeping the motor on. Once the ball is removed from the second sensor the output is turned off.

```
sketch_nov30a $
int TIP120pin1 = 11;
int TIP120pin2 = 12;
void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(TIP120pin1, OUTPUT);
  pinMode(TIP120pin2, OUTPUT);
  // analogWrite(TIP120pin, 250);
}

void loop() {
  // put your main code here, to run repeatedly:
  int sensorVal1 = digitalRead(2);
  int sensorVal2 = digitalRead(3);
  int sensorVal3 = digitalRead(4);
  int sensorVal4 = digitalRead(5);

  Serial.println(sensorVal1);
  Serial.println(sensorVal2);
  Serial.println(sensorVal3);
  Serial.println(sensorVal4);

  if (sensorVal2 == HIGH) {
    digitalWrite(11, LOW);
  }
  else {
    delay (250);
    digitalWrite(11, HIGH);
    delay (1000*5);
  }
}

sketch_nov30a $
if (sensorVal1 == HIGH) {
  digitalWrite(11, LOW);
}
else {
  digitalWrite(11, HIGH);
}

if (sensorVal4 == HIGH) {
  digitalWrite(12, LOW);
}
else {
  delay (250);
  digitalWrite(12, HIGH);
  delay (1000*5);
}

if (sensorVal3 == HIGH) {
  digitalWrite(12, LOW);
}
else {
  digitalWrite(12, HIGH);
}
}
```

Figure 6.6.2: Code used to control the sensors and motor using the Arduino software.

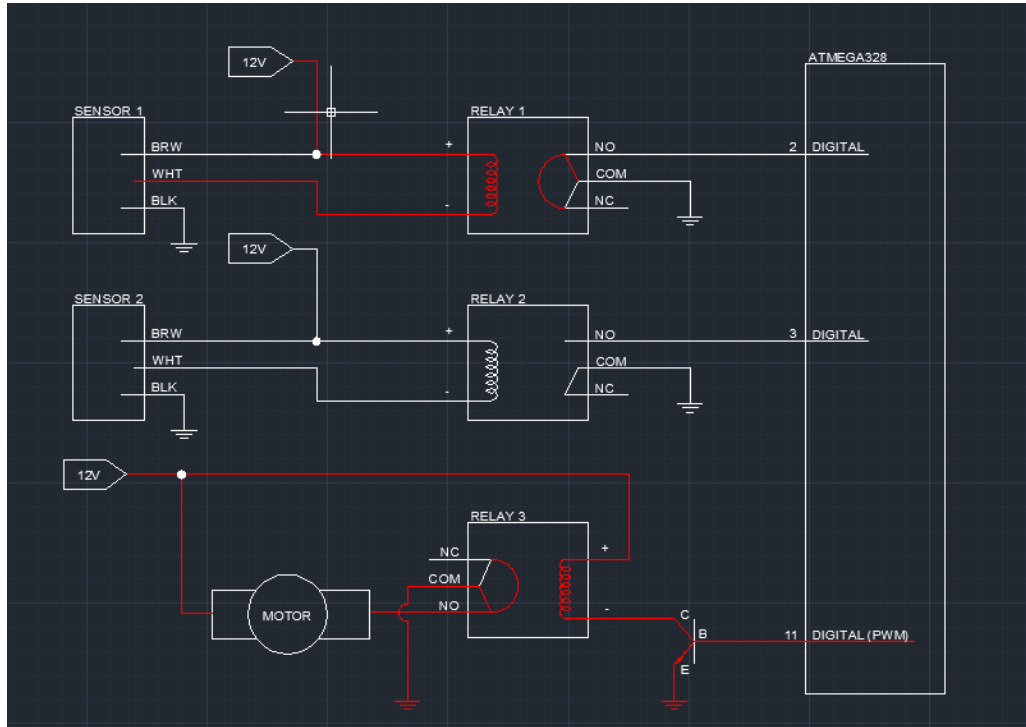


Figure 6.6.3: Schematic of the process of sensor 1 detecting and turning the motor on.

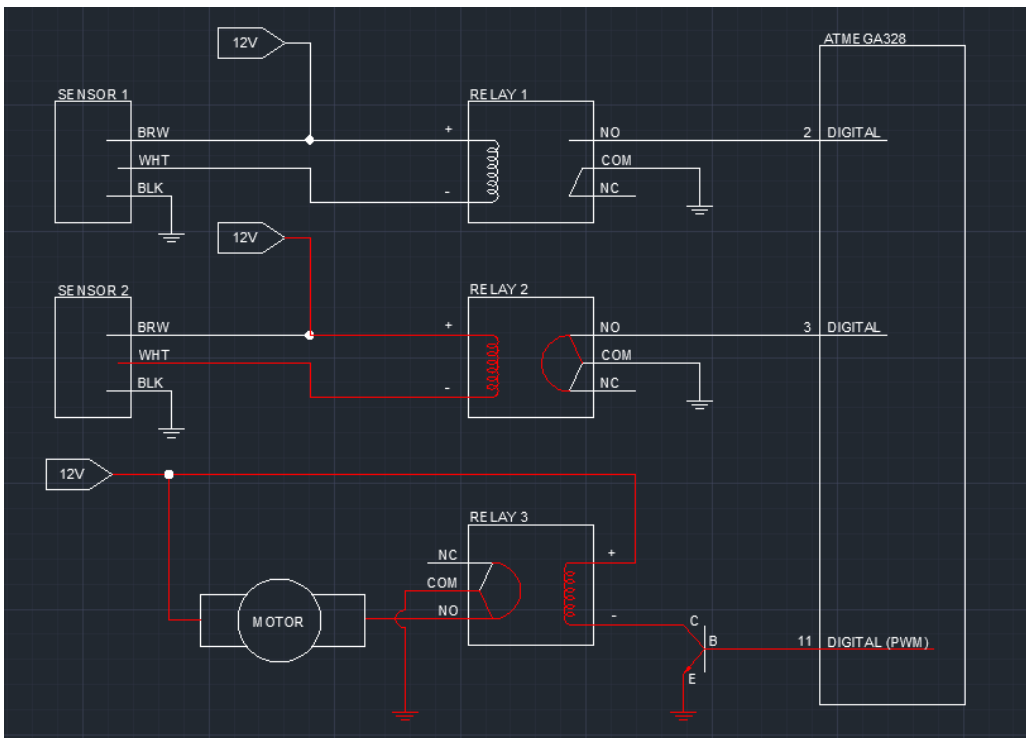


Figure 6.6.4: Schematic of sensor 2 detecting the ball and keeping the motor running after sensor 1 is off.

6.7 Power Supply

For our power supply requirements we needed a source of 12, 5, and 3.3 voltages. The components we are using use very small amperage and having a power source that can handle more than enough amps is always a safe bet. When looking at power supplies we were looking for resources that were already available. Because the power supply was our last priority and budget was running low we decided to use an older computers power supply. Computer power supplies have to supply various positive and negative voltages at various current ratings. The power supply we took from an old dell desktop is an ATX 600 W power supply. These are easy to use and have spreadsheets of the wiring and color coordination. To power the fan and sensors, 12 volts is needed. To power the microcontroller, LEDs, and other board components, 5 volts is required. The ATX power supply has these needed voltages and more. The ATX color coordination include; black wire = ground; red wire = 5 volts; orange wire = 3.3 volts. Since the ATX power supply is based on computer wiring a rig must be made to allow the power supply for normal control. The green wire leading to one connector must be connected to ground in order for the power supply to run. This wire lets the power supply know the computer is trying to turn on and thus must be rigged to turn on without a computer.

6.8 Smartphone App

For the smartphone application of the project, the group will be making a standard Android application using the Android API. To create this Android Application, the Android Studios IDE will be used for development.

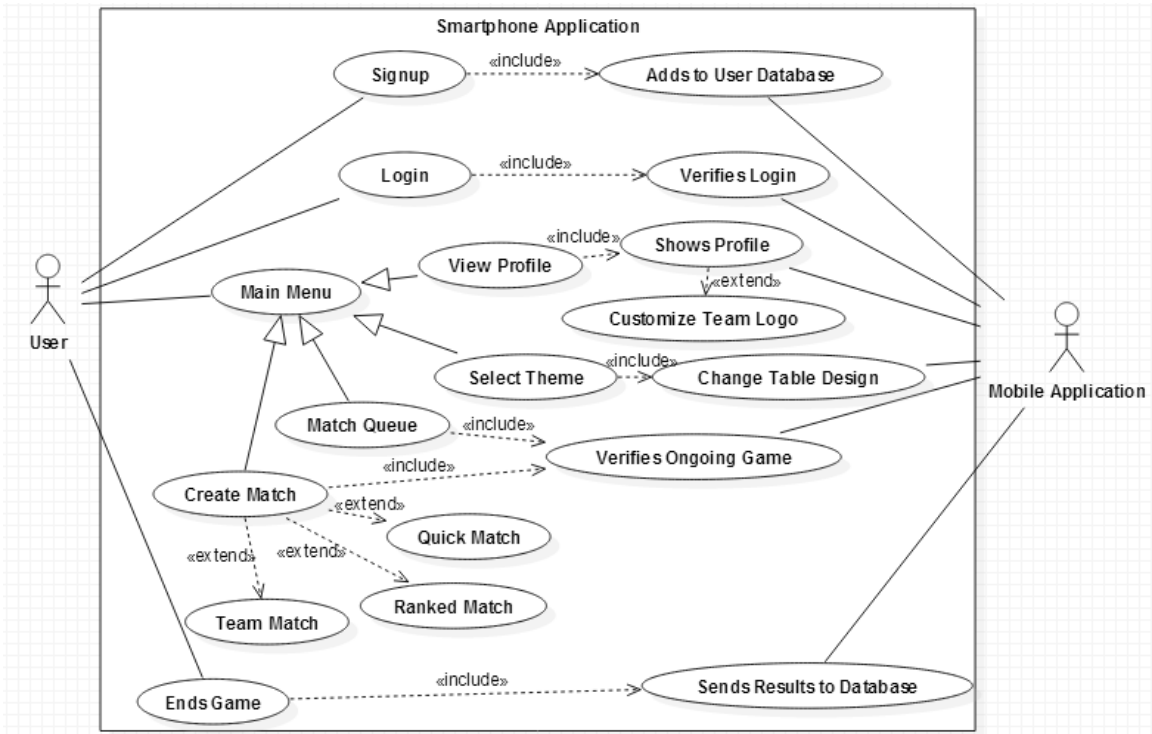


Figure 6.8.1 Use Case Diagram Shows the user interaction with the mobile application

This smartphone application will be the major component of the user interface features for the table. The features the application will have includes a sign up and login interface that will save the user information to a MySQL database. The app will then have a main menu screen. From this screen the user can see options such as viewing their own profile, queuing for a match, create a match, and create a team. In the profile screen the user will be able to view their rank, level and match history all of which are saved into a database. The user will also see a list of teams that they are currently apart of. For the queuing of the match on the smartphone, the user will be put into a queue that is in order of the time accessed to get into a match. Once a user queues for a match, the app will show who is currently playing on the table and the match score. This screen will also show the order of the people who queued in to play a match in a table below the current game statistics.

The other option from the main menu screen would be to create a match. The options for the match creation would be to create a team match, quick match, or ranked match. A quick match and ranked match only differs is how the players' statistics are affected. For a quick match both users would get points for their levels, with the winner getting more points than the loser. While ranked matches will affect user's rank, with the loser losing some points and the winner acquiring points to reach the next rank. The team match option allows users' to team up as teams of two. This type of match will just affect the team statistics such as team level and rank. Both of these statistics are affected in the same way the statistics for the quick and ranked match are in that level will always rise, more if you win

and less if you lose, and rank can go up and down depending on if the team wins or loses.

The final aspect of the application would be the team creation component. In this section, the users are given a choice to either join an already existing team or create an entirely new team. If the User wants to join an existing team, the user will be given a search bar to search for the team name. That search would look through a database that has all the teams in it. If the team is found, a request is sent to the team leader to join, but if the team is not found then the user is told that the group does not exist. When a user wants to create a team, the user will be given an inquiry as to what the team name is and which users do they want to invite to the team. Once that is completed, the user will be given a smaller version of a small section of the LED table display. With this section the user would be able to create a team logo.

6.9 Desktop App

The desktop application will create consists of a databases and applet. A MySQL database will store various information such as the match history, and the leaderboard. Meanwhile, the app will be responsible for queuing in new players and tracking in-game scores.

6.9.1 Web Page and Database

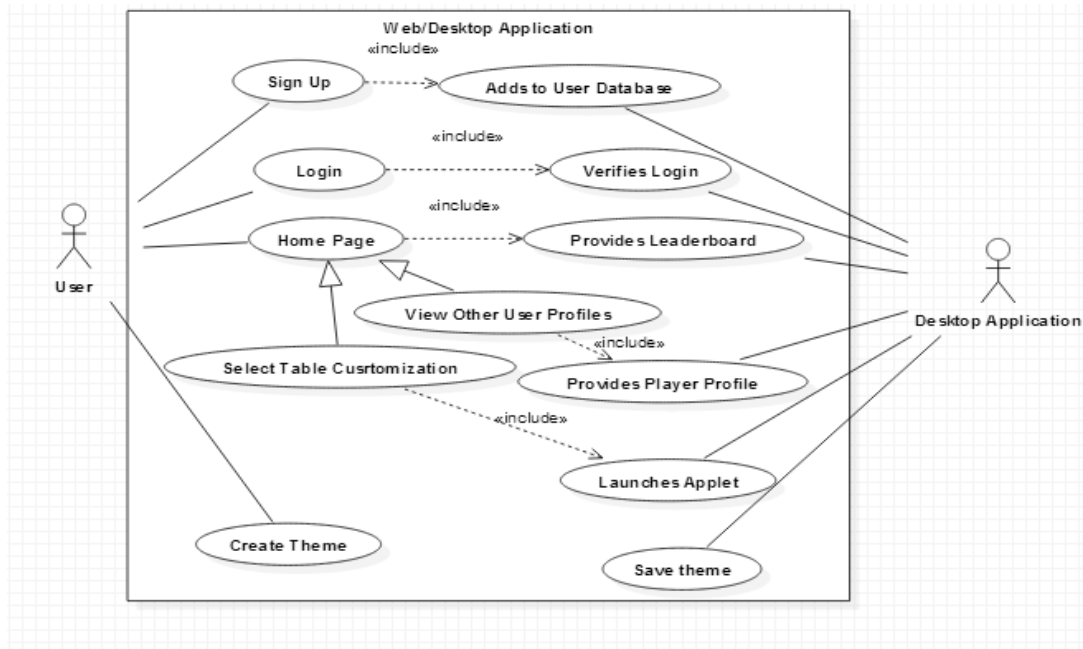


Figure 6.9.1: Use Case Diagram Shows the user interaction with the web application.

The project's components will consist of resources like a score-tracking applet, a player's profile, and a leader board. On the applet there will be a user login bar.

The player profile page will show various info about the player. Some of the information displayed would be the player name, rank and level. Player level does not have a set level cap and player rank can be anywhere from zero to ten. Player level is determined by whether the user wins or loses the game. If the user wins the game, they will get 100 points while the loser will only get 50. In order to level up, the user would only need 100 points for each level. To determine the rank, an equation, shown in Figure 6.9.2a, was created by the group to handle it.

	Win(Points Earned)	Lose(Points Lost)
Ranking	$50 + 0.10(\text{loser points})$	$-30 - 0.10(3000 - \text{winner Points})$

Table 6.9.2a: shows the equation used to calculate the rankings

Points For Rank	Rank Number
2701 - 3000	10
2401 - 2700	9
2101 - 2400	8
1801 - 2100	7
1501 - 1800	6
1201 - 1500	5
901 - 1200	4
601 - 900	3
301 - 600	2
0 - 300	1

Table 6.9.2b : Shows the level breakdown of the rankings.

Beside the rank and level, the profile will also show the user's name, nickname, which teams the player is on, and the player's match history. The list of the teams on the user's profile page will be handles by a database. On this database, the

different columns will represent the team name, team member count, the team level, and the team rank while each row will represent a different team. The match history will also be represented by a database. This database's different columns will represent the different information such as the date, result, match length, rank, resulting match points, who the opponent was, and what team if any were you on.

6.9.2 Applet

This applet is designed to operate on a specific tablet PC that will be used to accompany the table. It will feature software that can be used for the purpose of creating a queue of players. The user operating the app could be thought of as a referee with the responsibility of letting in new users after each game was complete. They would also have the responsibility of updating the scores during the game to track who won and lost. Thus, the users operating the app would be integrated into the game with the use of technology to make the game a smarter and more intelligently built experience.

6.10 Table Construction

Constructing the table required three levels of building. The first layer is the top layer consisting of the roof of the table and the glass and acrylic platforms placed over the sensors. We first started with a cut piece of plywood measured at 8' by 2.25'. A rectangular piece was cut out of the exact middle of the top allowing the glass to be placed. This cutout piece measured at 36" by 24". Grooves were placed around this cutout used to hold the glass even with the top of the table. Lastly for the top piece two triangles were cut out for the cup sensors 5" away from the ends of the table. The triangles are measured to add an inch around the base of the beer pong cups. Over these triangles are acrylic triangles an inch outside of the cutout.

The second layer consists of where the components are located inside the table. For this we needed the bases for the LED/sensor modules. This includes the cutout pieces of both triangles and grid cutout. These three pieces are lifted using basic cuts of wood to raise the modules just before the acrylic and glass. Along with the grid base an interlocking grid wall was made. 8 pieces of precut 1-1/2" were run the length way of the grid and 10 pieces along the width. These pieces were run 3" apart length wise and 2-1/2" width wise creating a grid. To hold these in place grooves were cutout to interlock each piece to each other making it sturdy. The last part of the second layer is the support. This is created using 2x4 wood and using a nailing gun to hold them together. An outside frame is created first and then two width run pieces on both sides of the grid to support the top. This support is glued using wood glue to the bottom of the base which is the third layer. An "L" bracket is screwed on each corner of the frame over lapping to hold the top in place.

The third layer is the bottom which has the frame support glued onto it. The base is cut to the same size as the top and has one hole between the cups and grid allowing power to be run inside of the table.

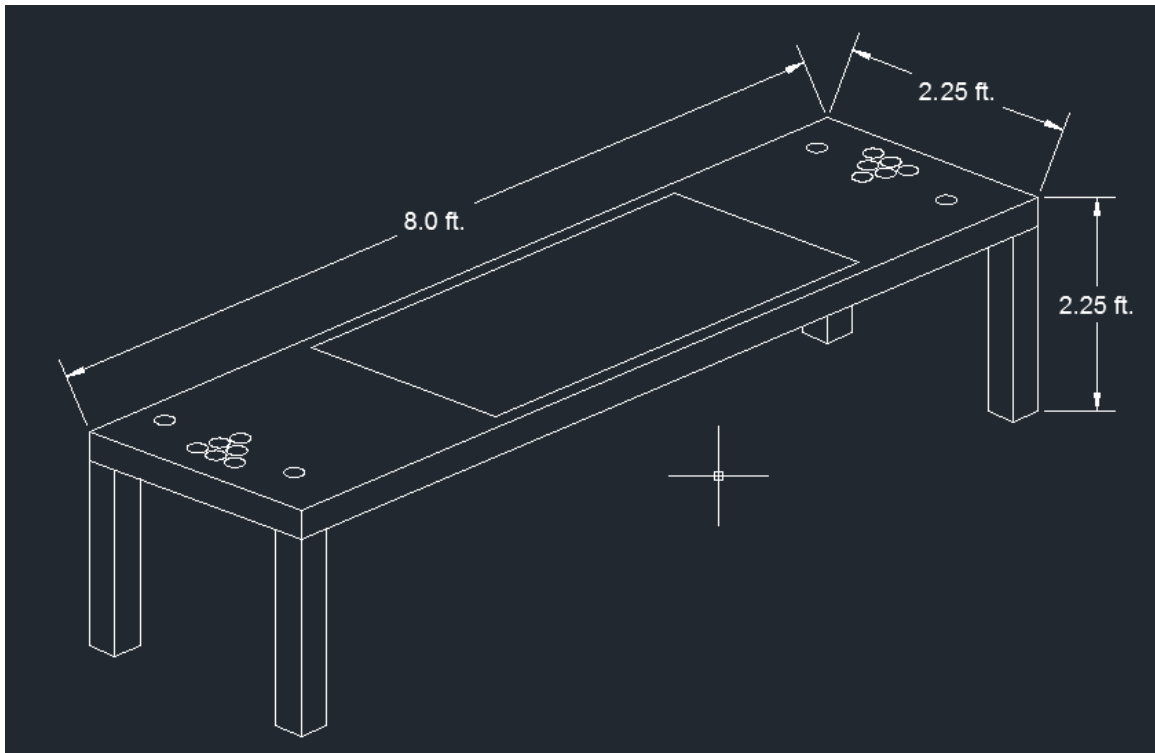


Figure 6.10.1: Table size and layout.

7.0 Prototype Testing

7.1 Test Environment

Testing for all components will be done in a dry, well-lit, enclosed environment where weather conditions, like rain or change in temperature, cannot affect the testing results and where interference from more random variables in testing can be avoided. Not all tests may not be performed in the same place as some of our components are being constructed exclusively by specific members and do not require assistance from other members. Care needs to be taken when testing the structure of the table for potential leaking points to make sure that no liquids come into contact with any circuitry we are building. It would also be best that the tests are performed at approximately room temperature, unless it is specifically required that they are not, and that testing be done when there is no threat of strong storms in the area to knock out power or surge our electrical equipment. Not only could extreme weather conditions damage our project, they could prove a threat to our

team members as well. As long as these conditions are met, we should be able to test our prototypes effectively and safely.

7.2 Test Components

7.2.1 MCU

Testing the MCU will mostly be accomplished by making sure that our main program for allowing it to interact with the rest of the table works properly. Any tests specifically focused on orchestrating the functions of the table, such as triggering the lights and setting up customization functions, will require the use of the TIVA C Experiment Board. The MCU that we intend to use in our final design is the same as the one in the TIVA C Experiment Board, and so as long as our programs can be run from the TIVA C board, they can also be run on our final design's circuit board.

Program testing will be done incrementally by first testing to see if the LED lights can be triggered properly, and if signals can be received from the infrared sensors. We may use Code Composer Studio, Notepad ++, and/or Eclipse to help us write the necessary code for the table functions, and a small program like either Hyperterminal or PuTTY may be needed for entering and receiving signals. Other than that the only components we need to test the table's functions are a USB cable to connect to the TIVA C Experiment Board so that new programs can be uploaded to it. After confirming that it can interact with a small test group of these parts, we will gradually connect more sensors, LEDs, LED drivers, and shift registers to get the full grid of peripherals working properly. This will inherently require more complex coding, and as long as the MCU is able to process and the data correctly and provide the desired results, we can say that it is functioning the way it is supposed to.

7.2.2 LED Array

Testing of the LED Array will be done incrementally and goes along with the process of testing the MCU. Early program testing requires the use of enough LEDs to fill up at least one LED driver and possibly connect to a shift register. Some degree of testing is done with the infrared sensors along with way to make sure that they are able to communicate their signals to the corresponding LEDs. A breadboard is required to connect these components temporarily while a complete program is incrementally built a complete program for the table to use. Gradually, we will make use of more LEDs, LED drivers, shift registers, and infrared sensors for testing once the initial testing is completed so that we can insure the parts continue to work as desired.

7.2.3 Impact Sensor Array

Testing of the Impact Sensor Array will be done incrementally and goes along with the process of testing the MCU. Sensors will be added gradually with to the Shift Registers as needed in order to test their ability to send signals to the MCU for the LEDs to respond to. Provided that the Sensors' input can be detected by the MCU and used to trigger the LEDs in a desired fashion, they will be considered to be functioning properly. Each sensor will be tested individually, and as part of a working system throughout our experiments, so by the time we are done we should easily have tested the ability of our sensors to work together properly.

7.2.4 Cup Display System

Testing of the Cup Display System will not differ too heavily from testing the LED Array. We will arrange a battery of tests to run through every possible combination of the arrangement of cups on the table and see if the Display System responds appropriately. When a cup is not present, the corresponding light should either not light up, or change to a different color. Each of the lights should be able to follow this rule, regardless of how the other cups are arranged, unless some manner of custom programming has been put in to make them do otherwise. As long as they can follow these basic tests, the Cup Display System is functioning properly.

7.2.5 Ball Cleaner

When testing the ball washer we ran into several problems. The first was the motor would turn on too fast and push the ball back out the entrance thus a delay was added. To make sure the ball stayed afloat the exit end the sensor had to be placed at an angle that the ball floated at. Distances and holes sizes were continuously changed to allow the sensors to work and find the right spot to mount them.

7.2.6 Power Supply

In order to test different parts of the circuitry during construction and insure that the correct amount of voltage is moving throughout the circuit, a few standard tools for measuring electronics will be needed. These include an oscilloscope and a multimeter, but may also require the use of resistors and a few extra wires in order to simulate the effect of other components on a circuit before they are actually attached.

7.2.7 Smartphone App

The Smartphone App would be tested using the Android Virtual Device on a PC, a smartphone (HTC One M7), and a table (Samsung Galaxy Note 3). The initial testing for the smartphone app will all be done on the Android Studios IDE. Android Studios can simulate the app on several different phones using their

virtual device counterpart. Just to test the framework all the designing would be done on the virtual device first.

Everything that will be tested on the virtual machine will also be tested on the smartphone and tablet. Each feature that will be incorporated into the app will be tested on each platform before going on to the next feature and or page. The project group will start the testing with the sign up framework on each device. For this the group will look to make sure the positioning of each component on the application will be positioned correctly on the page. The next part the group will focus on is the functionality of the page. At this point the project group will be creating our own accounts and other dummy accounts to check if the signup page for the application is taking in the information the user enters and storing it to the database. After checking the sign up page the group would then focus on logging in to the application. For this part the project group needs to have the app access the information inside the database. The project group will attempt to access the accounts that were previously set up to test the sign up feature. For the sake of testing, the login page would reload with the word “success” on the top of it, this will be used to indicate if the database was successfully accessed. If access to the database was not successful then the app will simply reopen the login page saying that the login was unsuccessful.

The project group will then test whether or not the database the group used for the signup can be showed on the application page. This will not be part of the final app but this will be used just to test the applications ability to display a database table in the desired areas of the application. Once the group has successfully incorporated the database table onto an application page, the group will start adding more dummy accounts to the database. This will be used to check whether or not the application can update the database to show the newly added information on the application. As stated earlier the signup database will only be used for testing purposes.

Once the database features on the application is tested, the project group will then just set up the framework for the application. This means the group will just set up the layout for the different pages of the application. For example, the project group will set up the application’s main menu screen. Just to make sure the layout is correct, the group will simulate it on each available platform to us before moving on to the next pages of the application. This way of setting up pages will be done to each page in the application. For pages that should show the database information on it, the signup database will be used as a placeholder. These components will be switched to the proper databases once the project group reaches to a point in the project where the group can receive information from the table.

7.2.8 Desktop App

The project group will test the different components of the desktop application on different computers just to make sure they all work and appear properly no matter where the project group tries to gain access to it. The first thing to make

sure of is just to check that all the layouts of each pages in the desktop app shows correctly on each different computer. This will include making sure that the different databases tables will show up on the correct pages. After the framework is checked out, the project group will continue with checking the functionality of the features on the desktop application.

After all the framework and functions of the website are checked, the project group would then proceed to work on the applet that will be accessed through a link on the website. The applet will be used to set themes for the table. So while the LED table is not at the point of communicating to the application, the project group will just test the applet too see if components work in the way that the project group desires. To check this the project group will create various themes using the applet. By creating these the themes the project group will be checking the component used for creating the different animations. This component is a graphic of the LED table. When creating the animations, the project group will test this component out by making sure each individual LED square can change to the desired color that the applet user specifies. Once the LED table is communicating to the table, the project group will see whether or not the table can correctly display the themes that were used in the initial testing of the applet.

7.2.9 Databases

The project group will go through the various features that make use of a database in order to test them properly. The first function to be checked will be the login feature of the website. To test this the project group will just be making dummy accounts. These dummy accounts will be used to check if the user information is added properly to the user database. Once this is checked out by the project group, we will also need to check if the other databases accurately adds the information for these accounts. To check that these databases will update and display properly, the project group will manually add in information at first, just while the LED table is still not at the phase of interacting with the application. The added information will be used to check if the database tables for the user's stats and game history is updating and displaying correctly. Once the table does reach a point where the information of the match can be sent to the application, the project group will do trial games just to check that the information is being properly sent and updating on the application.

Another database that the project group would need to test is working properly is the database that holds the different custom teams. This database will be tested by using the previously created themes that the project group made to test the applet. The project group will check this database by just making sure all the previously created databases by the different dummy users are shown on each account properly and the project group will make more themes to make sure that the database is accurately updating to incorporate the newly constructed themes by the user.

8.0 Administrative Content

8.1 Budget

All of the components necessary for building this project were either ordered online or obtained from a local retailer so that they could be acquired in a timely manner. The majority of the components are not particularly expensive, but the cost of the PCB and the materials to build the table were somewhat more costly than any of the other materials. The number of Shift Registers and LED drivers that were required did add on to the cost as well, but that was necessary and well worth it to meet the design requirements. The following is a table of the cost of all the components used in the design of the project:

Parts	Amount	Price	Total
Bluetooth	1	\$3.90	\$3.90
TLC5940 LED Driver	17	\$6.98	\$118.66
74HC165 Shift Register	35	\$1.88	\$65.80
Common Anode LED	80	\$0.07	\$5.60
Infrared Sensor	80	\$0.34	\$27.0
4 Color RGB extension Cable	4	\$11.00	\$40
Protoboard	10	\$2.00	\$20.00
Resistors			\$30
2-pin connectors	80	\$0.44	\$35.20
4-pin connectors	80	\$0.40	\$32
PCB			\$130
Table materials			\$100
Plexiglass			\$40
Bilge Blower	1	\$20.89	\$20.89
Power Supply			\$35
Proximity Sensor	2	\$76	\$152
DPDT Relay	1	\$3.95	\$3.95
SPDT Relay	2	\$3.50	\$7
Grand Total			\$1033.41

Table 8.1.1

More or less, the project's supplies totaled to about \$1000 dollars. This is significantly noteworthy because the research of other similar designs to this one revealed that the cheapest one was about \$2000. In that regard, this project managed to outclass the other existing designs.

The cost of materials was mostly handled by a benefactor found by one of the project members that has chosen to remain anonymous for professional reasons. Some extra costs were covered by the remaining members as well, but the majority of supplies were handled by this benefactor and the members of this project are very grateful for their contribution.

8.2 Labor Division

Labor was divided into about 4 different sections: table circuitry, table construction, table programming, and app programming. Edgar Alastre was given the responsibility of designing the circuitry that would be fit into the table for controlling the LEDs and infrared sensors. Jonathan Chang was made responsible for building an app that could track scores and queue new players as well as making it compatible with a database to store information about player profiles. Ashish Naik was tasked with learning how to program the hardware used for the table and programming the effects that the table would use in its displays. Finally, Colton Myers constructed the physical table as well as built and wired the Ball Cleaner to work properly.

At times some of the labor was shared amongst all of the members. Ashish Naik and Edgar Alastre worked together to understand the hardware used in the table as well as how to program it. Jonathan Chang was able to help program the table at several key points in the development stages, while all group members helped Colton Myers with constructing the table in some way. To complete the project it time, all members needed to pay attention to each other's progress and assist when necessary.

8.3 Milestones

October 11, 2015 First successful communication with single TLC5940

November 2, 2015 Construction of 3x3 LED and Sensor grid

November 4, 2015 3x3 LED and Sensor grid featured for Midterm Demonstration

December 2, 2015 Professional Presentation to Faculty

8.4 Group Member Information



Phone: (407) 223-9866

ealastremier@knights.ucf.edu

Edgar Alastre is a 25-year old international student from Venezuela pursuing a Bachelor's of Science in Electrical Engineering at the University of Central

Florida. He has a keen interest in computer hardware such as CPUs and GPUs and will seek work experience on these fields.



Phone: (954) 591-3769

jonathanchang@knights.ucf.edu

Jonathan Chang is a 22 year old graduating senior pursuing a Bachelor's of Science in Computer Engineering at the University of Central Florida. Has the ability to work with a variety of computer programming languages such as, but not limited to, Java and C. Currently searching for a job in fields related to software design.



Phone: (407) 484-7567

cmsport03@knights.ucf.edu

Colton Myers is a 24-year old student pursuing a Bachelor's of Science in Electrical Engineering at University of Central Florida. His interests are working in military simulation, specifically on military aircraft weapon systems at Lockheed.



Phone: (239) 940-1937

osmostrix@knights.ucf.edu

Ashish Naik is a 23-year old student raised here in the United States. He is attempting to obtain a Bachelor's of Science in Computer Engineering at the University of Central Florida. Work specialties are coding in C, Java, and MSP430 Assembly Code. Will be pursuing a job related to Software Design after graduation.

9.0 Conclusion

This project has taught its team members a great deal throughout its course. All of them now have a stronger understanding about the process of building an actual device using the skills they've learned and the difficulties one encounters in doing so. They also understand more about working together with other engineers of different backgrounds and mindsets. This will undoubtedly help them in their future employment opportunities to function in a realistic work environment.

Appendices

A. Bibliography

Relevant Technologies:

A Brief Introduction to the Serial Peripheral Interface (SPI). (2015). Retrieved August 5, 2015.

A Look at the Basics of Bluetooth Technology. (2015). Retrieved August 5, 2015.
DeMilo, C., & McDaniel, D. (2013). Truly round LED sources deliver optimal performance in many SSL applications (MAGAZINE). Retrieved August 3, 2015.

Harris, Tom, and Wesley Fenlon. "How Light Emitting Diodes Work" 31 January 2002. HowStuffWorks.com.
<<http://electronics.howstuffworks.com/led.htm>> 02 August 2015.

Holmes, D. G., & Lipo, T. A. (2003). *Pulse width modulation for power converters. [electronic resource] : principles and practice*. Hoboken, NJ : John Wiley, 2003.

Marian, P. (2010). Fairy Flashing LED Lights. Retrieved August 5, 2015, from <http://www.electroschematics.com/766/flashing-led-lights/>

Paul, R. P., Rathod, G. B., Trivedi, V. R., & Thakkar, P. V. (2013). Persistence of Vision Control Using Arduino. *International Journal Of Intelligent Systems & Applications*, 6(1), 102. doi:10.5815/ijisa.2014.01.11

Reiss, A. (2011, October 21). The Difference Between Single- and Tri-colored LEDs. Retrieved August 3, 2015.

Sellers, F. (2011, November 8). Back to LED Basics. Retrieved August 3, 2015.

Serial Communication using UART. (n.d).

SPI Interface in embedded systems. (2006). Retrieved August 5, 2015.

Spookyblue.com » How do flashing LEDs work? (2009, August 21). Retrieved August 5, 2015, from <http://spookyblue.com/spookyblog/how-do-flashing-leds-work>

Existing Similar Projects:

100Pixel RGB LED table - Interactive (touch). (2012, April 29). Retrieved July 24, 2015, from <http://www.it-gecko.de/100pixel-rgb-led-tisch-interaktiv-touch.html>

Ausley, L., Moyerman, J., & Smith, A. (2013). *Senior Design 1 Project Documentation: 3D LED Cube*. Orlando, FL.

Braun, C., Hogan, R., Jackson, J., & Thomson, D. (n.d.). The DCPPTSS. Retrieved July 16, 2015, from http://www.eecs.ucf.edu/seniordesign/fa2011sp2012/g06/DCPPTSS/THE_DCPPTSS.html

Darrah, K. (2012, September 16). How to Control a Ton of RGB LEDs with Arduino & TLC5940. Retrieved July 24, 2015.

Monn, S., Dworkin, M., & Alami, O. (2013). *Senior Design I Document: Game Qube*. Orlando, FL.

Phillips, Z. F., D'Ambrosio, M. V., Tian, L., Rulison, J. J., Patel, H. S., Sadras, N., & ... Waller, L. (2015). Multi-Contrast Imaging and Digital Refocusing on a

Mobile Microscope with a Domed LED Array. *Plos ONE*, 10(5), 1-13.
doi:10.1371/journal.pone.0124938

Components:

35 Model Ultimate Building Set 744476124180. (n.d.). Retrieved November 15, 2015, from <http://www.knex.com/shop/17783/35-model-ultimate-building-set/>

74HC165; 74HCT165 8-bit parallel-in/serial out shift register. (2008, March 14). Retrieved September 17, 2015, from http://www.nxp.com/documents/data_sheet/74HC_HCT165.pdf

8 Bit Parallel Load Shift Register DIP 16. (n.d.). Retrieved September 9, 2015, from http://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&storeId=10001&productId=45495&catalogId=10001&CID=GOOG&gclid=Cj0KEQjwsb-vBRCLj7TvqpGx_MoBEiQALgFGngQgH4QySof2xGHj4Kctxl8AT6MN_xUYixeqoVBnnnYaAmAQ8P8HAQ

AT91SAM ARM-based Flash MCU SAM7S512 SAM7S256 SAM7S128 SAM7S64 SAM7S321 SAM7S32 SAM7S161 SAM7S16. (2012). San Jose, CA: Atmel Corporation.

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V 8-bit Atmel Microcontroller with 16/ 32/64KB In-System Programmable Flash SUMMARY. (2014). San Jose, CA: Atmel Corporation.

Ball Washers. (n.d.). Retrieved November 6, 2015, from <https://www.standardgolf.com/ball-washers.html>

Capacitive sensor, Theory, application and design. (2012, April 7). Retrieved September 28, 2015, from <https://www.youtube.com/watch?v=Qltuf6lNvml>

Home Plastic Injection Molding with an Epoxy Mold. (n.d.). Retrieved November 15, 2015, from <http://www.instructables.com/id/Home-Plastic-Injection-Molding-with-an-Epoxy-Mold/>

Lancaster Table & Seating 30" x 96" Heavy Duty White Granite Plastic Folding Table. (n.d.). Retrieved November 15, 2015, from http://www.webstaurantstore.com/lancaster-table-seating-30-x-96-heavy-duty-white-granite-plastic-folding-table/384YCZ9630.html?utm_source=Google&utm_medium=cpc&utm_campaign=GoogleShopping&gclid=CjwKEAiA1JuyBRCogJLz4J71kj0SJAds_d6QRVG1DqCPYpFN8Vh7vy5Onxn4kK

- Metal Sheets & Rods. (n.d.). Retrieved November 15, 2015, from <http://www.homedepot.com/b/Tools-Hardware-Hardware-Metal-Sheets-Rods/N-5yc1vZc2b9/Ntk-Extended/Ntt-metal?Ntx=mode-matchpartialmax&NCNI-5>
- Piezo Vibration Sensor - Small Vertical. (n.d.). Retrieved July 16, 2015, from <http://store.cutedigi.com/piezo-vibration-sensor-small-vertical/>
- Programming the Ev3 Infrared Sensor. (2015, June 21). Retrieved September 11, 2015, from <https://www.youtube.com/watch?v=HfOxl4kfjok>
- Reflective Optical Sensor with Transistor Output. (2012, October 2). Retrieved December 6, 2015, from <http://www.vishay.com/docs/83760/tcrt5000.pdf>
- SAM3X / SAM3A Series Atmel | SMART ARM-based MCU DATASHEET.* (2015). San Jose, CA: Atmel Corporation.
- Shift Register 8-Bit - 74HC595. (n.d.). Retrieved September 9, 2015, from <https://www.sparkfun.com/products/733>
- Steel Assembly Table - 72 x 36" (n.d.). Retrieved November 15, 2015, from http://www.uline.com/Product/Detail/H-3630/Packing-Tables/Steel-Assembly-Table-72-x-36?pricode=WY703&gadtype=pla&id=H-3630&gclid=CjwKEAiA1JuyBRCogJLz4J71kj0SJADsd6QRQm3Lw2LsLg03UPuZZkHLneKVqgnt9Q1Pv_-ESegY6xoCUwLw_wcB&gclsrc=aw.ds
- Tiva™ TM4C123GH6PM Microcontroller.* (2007). Austin, TX: Texas Instruments Incorporated.
- TLC5940 16-Channel LED Driver With DOT Correction and Grayscale PWM Control. (2015, November 1). Retrieved December 6, 2015, from <http://www.ti.com/lit/ds/symlink/tlc5940.pdf>
- Weight sensors: How do they work? (2010, March 17). Retrieved September 30, 2015, from <https://www.youtube.com/watch?v=zAddvPHfKw>
- Research:**
- Copyrights. (2015). Retrieved December 10, 2015, from <http://www.ti.com/corp/docs/legal/copyright.shtml>
- Arduino-due-tlc5940 - TLC5940 Library for Arduino Due. (n.d.). Retrieved December 11, 2015, from <https://code.google.com/p/arduino-due-tlc5940/>
- Martin, R. (n.d.). UML Tutorial: Part 1 -- Class Diagrams. Retrieved September 15, 2015, from <http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf>

- Rules - Drinking Beirut (Beer Pong). (n.d.). Retrieved July 24, 2015, from <http://www.beirut-guide.com/rules/basic.php>
- Rules-Official Tournament Rules | BPONG. (n.d.). Retrieved July 24, 2015 from <http://www.bpong.com/tournament-rules/>
- Rules-Typical House Rules | BPONG. (n.d.). Retrieved July 24, 2015, from <http://www.bpong.com/beer-pong-rules/>
- ShiftRegSN74HC165N. (n.d.). Retrieved December 7, 2015, from <http://playground.arduino.cc/Code/ShiftRegSN74HC165N>
- TLC5940 driver for Tiva. (2014, August 1). Retrieved October 3, 2015, from <http://forum.43oh.com/topic/7599-tlc5940-driver-for-tiva/>
- TLC5940 Programing Flow Chart v0.1. (2005, September 29). Retrieved October 3, 2015, from <http://www.ti.com/lit/sw/slvc106/slvc106.pdf>
- Tutorial – Arduino and the TLC5940 PWM LED Driver IC. (2013, October 21). Retrieved October 3, 2015, from <http://tronixstuff.com/2013/10/21/tutorial-arduino-tlc5940-led-driver-ic/>

Standards:

- D2395 – 14 Standard Test Methods for Density and Specific Gravity (Relative Density) of Wood and Wood-Based Materials. (2015). In *Book of Standards* (Vol. 04.10). PA: Subcommittee D07.01 on Fundamental Test Methods and Properties.
- D2633 - 13a Standard Test Methods for Thermoplastic Insulations and Jackets for Wire and Cable. (2015). In *Book of Standards* (Vol. 10.02). PA: Subcommittee D09.18 on Solid Insulations, Non-Metallic Shieldings and Coverings for Electrical and Telecommunication Wires and Cables.
- D4442-07 Standard Test Methods for Direct Moisture Content Measurement of Wood and Wood-Base Materials. (2015). In *Book of Standards* (Vol. 04.10). PA: Subcommittee D07.01 on Fundamental Test Methods and Properties.
- E2126-11 Standard Test Methods for Cyclic (Reversed) Load Test for Shear Resistance of Vertical Elements of the Later Force Resisting System for Buildings. (2015). In *Book of Standards* (Vol. 04.12). PA: Subcommittee E06.11 on Horizontal and Vertical Structures/Structural Performance of Completed Structures.
- E564-06(2012) Standard Practice for Static Load Test for Shear Resistance of Framed Walls for Buildings. (2015). In *Book of Standards* (Vol. 04.11). PA: Subcommittee E06.11 on Horizontal and Vertical Structures/ Structural Performance of Completed Structures.

B. Copyright Permissions

Attempt to request permission to use an image by IT-Gecko:



Ashish Naik

To: webmaster@it-gecko.de; ...



Reply all

Fri 12/4/2015 10:31 PM

To the Engineer responsible for building the 100Pixel RGB LED Table,

I am a student at the University of Central Florida who is currently working on finishing a project that makes use of much of the same technology as the table you built. We wanted to explain how your project helped inspire us to build ours and would like permission to use an image of your table to show what it is you built exactly. Please let me know if we have your permission to use one of the pictures of your table posted here:

<http://www.it-gecko.de/100pixel-rgb-led-tisch-interaktiv-touch.html>



100Pixel RGB-LED Tisch - Interaktiv (Touch) - IT-Gecko

100 Pixel großer elektronischer RGB LED Tisch mit einer interaktiven Plexiglas Oberfläche (Touch-Sensor) und Bluetooth Anbindung.

[Read more...](#)

Thank you for sharing the information of how to use LED Drivers to create such a project. It is easily possible that our own project would never have made any progress were it not because of the knowledge that you have passed on to us.

Permission to use image from video by Kevin Darrah:



Kevin Darrah <kevin.darrah@usafirmware.com>

To: Ashish Naik; ...



Reply all

Sat 12/5/2015 7:39 PM

You replied on 12/10/2015 6:51 AM.

Action Items

Hello,

I just received this message - yes you may use any content you'd like from my videos. I'd like to see the final project if possible :)

Good luck!

Best Regards,

Permission to use diagram of Infrared Sensor setup circuit:



Dave Hunt <dave@huntgang.com>

To: Ashish Naik;



Reply all |

Mon 12/7/2015 10:09 PM

I have no problems with you using the images on my blog for educational purposes.

Permission to use Image from LED Dome Array:

Computational CellScope. A. Device observing a sample using a Nexus 4 smartphone. B. Optical schematic of the CellScope device with our custom-made domed LED illuminator. C. CAD assembly of the dome. D. Assembled dome and control circuitry.

Copyright: © 2015 Phillips et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. (Phillips)

Permission to use image of Game Qube:



Stephen Monn <kevinsbro@gmail.com>

To: Ashish Naik; Cc: Omar Alami <OmarAlami24@gmail.com>; Matthew Dworkin <mattd...



Reply all |

Sat 12/5/2015 11:33 AM

You replied on 12/5/2015 11:37 AM.



Action Items






Yeah, go ahead! You're making an LED cube? There's definitely room to improve upon the original design. Feel free to ask for advice.

Stephen Monn

Permission to use an image about the SN74HC165:

PR P R <pr@nxp.com>
To: Ashish Naik; ...

  Reply all | 
Thu 12/10/2015 10:17 AM




Agreed.

Best, NXP

Attempt to contact StellaScapes for permission using an image of the 3D RGB LED Cube that they funded:



Ashish Naik
To: sales@stellascapes.com; ...

  Reply all | 
Sat 12/5/2015 11:18 AM

To whom it may concern,

I am a student at the University of Central Florida that is working on a project for graduation. Our project was partially inspired by the 3D RGB LED Cube that your company sponsored. My group would like to request the use of an image of that cube to help explain what it was and how our project relates to it. We appreciate you making the effort to fund such a project as we were able to use it to help provide information about how to build our own. Please let me know if we have your permission to do so.

Permission to use images from documentation created by Texas Instruments:

GEN, Email Technical Support, www.ti.com, SLVS515D

SU

support@ti.com

Reply all |

To:

Ashish Naik;

...

...

Mon 12/7/2015 12:57 PM

Hello Ashish,

Copyright

<http://www.ti.com/corp/docs/legal/copyright.shtml>

Larry Bassuk 972-917-5458

Shannon Gragsone 972-917-5460

If you have any questions or concerns, please contact us.

Regards,

Lisa Barrett
TI Customer Support
Americas Customer Support Center
512-434-1560



<http://e2e.ti.com/>

http://www-k.ext.ti.com/sc/technical_support/pic/americas.htm

TI makes no warranties and assumes no liability for applications assistance or customer product design. You are fully responsible for all design decisions and engineering with regard to your products, including decisions relating to application of TI products. By providing technical information, TI does not intend to offer or provide engineering services or advice concerning your designs.

Please do not delete the below Thread ID when replying to this email, doing so will delay our response to your inquiry

[THREAD ID:1-WUAVIT]

Excerpt from Texas Instruments Copyrights page:

TI further grants permission to non-profit, educational institutions (specifically K-12, universities and community colleges) to download, reproduce, display and distribute the information on these pages solely for use in the classroom. This permission is conditioned on not modifying the information, retaining all copyright notices and including on all reproduced information the following credit line: "Courtesy of Texas Instruments". Please send us a note describing your use of this information under the permission granted in this paragraph. (Copyrights)

Excerpt from Vishay Copyrights page:

Vishay Intertechnology, Inc. permits browsing of the Vishay Intertechnology, Inc. web site and the necessary caching of the Web Content onto your computer. Vishay Intertechnology, Inc. permits the download, reproduction, display and internal distribution of the Web Content posted on this site solely for informational and non-commercial (the use of the Web Content in purchasing and design applications are not considered commercial uses) or personal use, provided that the Web Content is not modified and provided further that all copyright and

proprietary notices remain as they appear within such Web Content. Vishay Intertechnology, Inc. further grants to educational institutions (specifically K-12, universities and community colleges) permission to download, reproduce, display and distribute the Web Content posted on this site solely for use in the classroom, provided that such institutions identify Vishay Intertechnology, Inc. as the source of the Web Content and include the following credit line: "Courtesy of Vishay Intertechnology".

Excerpt from TE Connectivity page:

The Content of this Website is protected by Copyright Laws of the U.S. and other countries and international treaty provisions and is owned by, or licensed to TE group of companies. You may download Content only for your personal use for non-commercial purposes, provided you also retain all copyright and other proprietary notices, but no modification or further reproduction of the Content is permitted. Website Content may not be distributed, reproduced, publicly displayed, downloaded, modified, reused, re-posted, or otherwise used except as provided herein without the express prior written permission of TE. All other rights reserved. See Copyright Policy/DMCA on this Website for additional information.

Excerpt from SHARP page:

All text, photographs, illustrations, videos, music, software, etc. (hereinafter referred to as "content") on this website are protected by copyright owned by Sharp Corporation and its affiliates (hereinafter referred to as "SHARP") or by a third party. Content may be downloaded or otherwise reproduced only for the purposes of personal use, or use within a household, or other limited uses equivalent thereto. Furthermore, when content is accompanied by a SHARP or third-party copyright notice, the content must be reproduced with this notice appended intact thereto. For reproduction other than for the purposes mentioned above, when the copyright holder has indicated separate terms and conditions for use of the specific content, the content may be used in accordance with those terms and conditions. Usage may be refused in the event likenesses and/or works of third parties, etc., are incorporated in the content or if SHARP deems the usage to be inappropriate.

C. Software

Table Program Run On MCU:

```
// Ashish Naik  
// December 4, 2015  
// Due_pattern_13
```

// This code is used for programming the microcontroller for a project operating about 80 LEDs and infrared sensors.

/// Pin Assignments:

/// - SPI CLK = SDA1, A.17, peripheral B -> TLC5940 SCLK (pin 25)

/// - SPI MOSI = TX1 (pin 18), A.11. peripheral A -> TLC5940 SIN (pin 26)

/// - SPI MISO = RX1 (pin 17), A.10, peripheral A -> TLC5940 SOUT (pin 17)
[opt]

/// - pin 6 -> TLC5940 VPRG (pin 27)

/// - pin 5 -> TLC5940 XLAT (pin 24)

/// - pin 4 -> TLC5940 BLANK (pin 23)

/// - pin 3 -> TLC5940 GSCLK (pin 18)

/// - pin 2 -> TLC5940 XERR (pin 16) [optional]

//

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include "Arduino.h"

#include "due_tlc5940.h" // This library was found online. The bibliography
//references the source

#define NUMBER_OF_SHIFT_CHIPS 8

/* Width of data (how many ext lines).

*/

#define DATA_WIDTH NUMBER_OF_SHIFT_CHIPS * 8

```

/* Width of pulse to trigger the shift register to read and latch.
*/
#define PULSE_WIDTH_USEC 5

/* Optional delay between shift register reads.
*/
#define POLL_DELAY_MSEC 1

/* You will need to change the "int" to "long" if the
 * NUMBER_OF_SHIFT_CHIPS is higher than 2.
*/
#define BYTES_VAL_T unsigned int

int loadPin      = 8; // Connects to Parallel load pin the 165
int clockEnablePin = 9; // Connects to Clock Enable pin the 165
int dataPin      = 11; // Connects to the Q7 pin the 165
int clockPin     = 12; // Connects to the Clock pin the 165

// Color definitions for use in ledNode struct
typedef enum COLOR {

    BLUE,

    GREEN,

    RED,

    YELLOW,

    PINK,

```



```

}LedNode;

// Set the initial values of a node
// NOTE: Each node will correlate to a single LED, and therefore 3 channels that
lead to the LED.
// Channel numbers are equal to idNumber x 3 = channel_1.
// channel_2 = channel_1+1.
// channel_3 = channel_1+2.
struct LedNode *initializeN(LedNode *subject, int id) {

    subject = (LedNode *)malloc(sizeof(LedNode));
    subject->idNumber = id;
    subject->current = BLACK;
    subject->blinking = 0;
    subject->off = 1;
    subject->north = NULL;
    subject->east = NULL;
    subject->south = NULL;
    subject->west = NULL;
    return subject;
}

// This checks if channel numbers need to be adjusted based on number of
TLCs.
// We never use channel 15 of any TLC, so we need to adjust our numbers
accordingly.

```

```

int channelCheck(int n) {
    int i;
    int variable;

    for (i = 1; i < (NUM_TLCS+1); i++) {
        variable = ((i*16)-1);
        if (n >= variable) {
            n++;
        }
    }

    return n;
}

```

// Small recursive function for calculating exponential power equivalent

```

int power (int x, int y) {

    if (y > 0) {
        x = x*(power(x, y-1));
    }
    else {
        return 1;
    }

    return x;
}

```

```

// The following functions are used to create the Ripple effect across the board
// Rip is where the effect starts
// The effect spans two perimeters
// ADJUST TO GRID SIZE
void Rip(struct LedNode *subject, int color) {
    activateColor(subject, color);
    delay(500);
    deactivateColor(subject);

    // Use this nodes to travel along the reference pointers
    struct LedNode *north, *ne, *east, *se, *south, *sw, *west, *nw;

    // Activate the adjacent nodes to create the ripple effect
    if (subject->north != NULL) {
        north = subject->north;
        activateColor(north, color);

        if (north->east != NULL) {
            ne = north->east;
            activateColor(ne, color);
        }

        if (north->west != NULL) {
            nw = north->west;
            activateColor(nw, color);
        }
    }
}

```

```
}

if (subject->east != NULL) {
    east = subject->east;
    activateColor(east, color);
}

if (subject->south != NULL) {
    south = subject->south;
    activateColor(south, color);

    if (south->east != NULL) {
        se = south->east;
        activateColor(se, color);
    }

    if (south->west != NULL) {
        sw = south->west;
        activateColor(sw, color);
    }
}

if (subject->west != NULL) {
    west = subject->west;
    activateColor(west, color);
}
```

```
delay(500);

deactivateBorder(subject);

// Activate the next set of adjacent nodes to make the effect look larger
if (nw != NULL) {
    if (nw->west != NULL) {
        nw = nw->west;

        if (nw->north != NULL) {
            nw = nw->north;
            activateColor(nw, color);
        }
    }
}

if (north != NULL) {
    if (north->north != NULL) {
        north = north->north;
        activateColor(north, color);
    }
}

if (ne != NULL) {
    if (ne->east != NULL) {
        ne = ne->east;
```

```
    if (ne->north != NULL) {
        ne = ne->north;
        activateColor(ne, color);
    }
}
```

```
if (east != NULL) {
    if (east->east != NULL) {
        east = east->east;
        activateColor(east, color);
    }
}
```

```
if (se != NULL) {
    if (se->east != NULL) {
        se = se->east;

        if (se->south != NULL) {
            se = se->south;
            activateColor(se, color);
        }
    }
}
```



```
if (south != NULL) {  
    if (south->south != NULL) {  
        south = south->south;  
        activateColor(south, color);  
    }  
}
```

```
if (sw != NULL) {  
    if (sw->west != NULL) {  
        sw = sw->west;  
  
        if (sw->south != NULL) {  
            sw = sw->south;  
            activateColor(sw, color);  
        }  
    }  
}
```

```
if (west != NULL) {  
    if (west->west != NULL) {  
        west = west->west;  
        activateColor(west, color);  
    }  
}
```

```
}
```

```
// Simple function that flashes an chosen LED on and off in the same function.
```

```
// Delays are currently a value set before program is run.
```

```
// We can modify the delays to be dependent on what program is run later.
```

```
void NodeFire(struct LedNode *subject) {
```

```
    int relative = (subject->idNumber)*3;
```

```
    if ((subject->current) == BLUE) {
```

```
        activateColor(subject, 0);
```

```
        delay(10);
```

```
        deactivateColor(subject);
```

```
    }
```

```
    if ((subject->current) == GREEN) {
```

```
        activateColor(subject, 1);
```

```
        delay(10);
```

```
        deactivateColor(subject);
```

```
    }
```

```
    if ((subject->current) == RED) {
```

```
        activateColor(subject, 2);
```

```
    delay(10);

    deactivateColor(subject);
}

if ((subject->current) == YELLOW) {
    activateColor(subject, 3);
    delay(10);

    deactivateColor(subject);
}

if ((subject->current) == PINK) {
    activateColor(subject, 4);
    delay(10);

    deactivateColor(subject);
}

if ((subject->current) == PURPLE) {
    activateColor(subject, 5);
    delay(10);

    deactivateColor(subject);
}

if ((subject->current) == ORANGE) {
```

```

        activateColor(subject, 6);
        delay(10);

        deactivateColor(subject);
    }

    if ((subject->current) == WHITE) {
        activateColor(subject, 7);
        delay(10);

        deactivateColor(subject);
    }

    if ((subject->current) == BLACK) {
        activateColor(subject, 8);
        delay(10);

        deactivateColor(subject);
    }
}

// Basic switch function. LED off becomes on. LED on becomes off.
// The "hold" integer below is used to indicate when this function can be allowed
// to run in the program.
// This keeps the LED from changing prematurely.
// Can use an array of such integers to manage a matrix of LEDs
static int hold = 0;

```

```
static COLOR pocket;
void SwitchUp(struct LedNode *subject) {

    if ((subject->off)== 1) {
        if ((subject->current) == BLUE) {
            activateColor(subject, 0);
        }

        if ((subject->current) == GREEN) {
            activateColor(subject, 1);
        }

        if ((subject->current) == RED) {
            activateColor(subject, 2);
        }

        if ((subject->current) == YELLOW) {
            activateColor(subject, 3);
        }

        if ((subject->current) == PINK) {
            activateColor(subject, 4);
        }

        if ((subject->current) == PURPLE) {
            activateColor(subject, 5);
        }
    }
}
```

```

    if ((subject->current) == ORANGE) {
        activateColor(subject, 6);
    }

    if ((subject->current) == WHITE) {
        activateColor(subject, 7);
    }

    if ((subject->current) == BLACK) {
        activateColor(subject, 8);
    }

    subject->off = 0;

}
else {

    deactivateColor(subject);

}

}

// Function that allows a fade from any one color to another
void FadeColor(struct LedNode *subject, int color) {

```

```

int relative = channelCheck(((subject->idNumber)*3));
int current1, current2, current3;
int standard1, standard2, standard3;

standard1 = B[color][subject->idNumber];

standard2 = G[color][subject->idNumber];

standard3 = R[color][subject->idNumber];

int i = 0;

for (i = 0; i < 9; i++) {
    if ((subject->current) == library[i]) {
        current1 = B[i][subject->idNumber];
        current2 = G[i][subject->idNumber];
        current3 = R[i][subject->idNumber];
    }
}

int halt = 0;

while ((current1 != standard1) || (current2 != standard2) || (current3 !=
standard3)) {

    if (current1 != standard1) {
        if (current1 < standard1) {

```

```
        current1++;
        setGSData(relative, current1);

    }
    else if (current1 > standard1) {
        current1--;
        setGSData(relative, current1);
    }
}
```

```
if (current2 != standard2) {
    if (current2 < standard2) {
        current2++;
        setGSData(relative+1, current2);

    }
    else if (current2 > standard2) {
        current2--;
        setGSData(relative+1, current2);
    }
}
```

```
if (current3 != standard3) {
    if (current3 < standard3) {
        current3++;
```



```

        setGSData(relative+2, current3);

    }
    else if (current3 > standard3) {
        current3--;
        setGSData(relative+2, current3);
    }
}

delay(10);
sendGSData();
}

subject->current = library[color];

if (subject->current == BLACK) {
    subject->off = 1;
}

Serial.print("Fade Complete \n");

}

// Function for determining whether a specific pin was or was not activated
boolean pinCheck(int check, int pinValues) {
    int sum = ((I_num-1) - check);

```

```

    if (sum == pinValues) {
        return true;
    }

    return false;
}

// Deactivates LEDs adjacent to a single LED in particular
void deactivateBorder(struct LedNode *subject) {
    struct LedNode *jump1 = subject->north;
    struct LedNode *jump2 = subject->south;

    deactivateColor(subject->north);
    deactivateColor(jump1->east);
    deactivateColor(subject->east);
    deactivateColor(jump2->east);
    deactivateColor(subject->south);
    deactivateColor(jump2->west);
    deactivateColor(subject->west);
    deactivateColor(jump1->west);
}

// This deactivates any selected LED
void deactivateColor(struct LedNode *subject) {
    int relative = channelCheck(((subject->idNumber)*3));

    setGSData((relative), 0);
}

```

```

    setGSDData((relative+1), 0);
    setGSDData((relative+2), 0);
    sendGSDData();
    subject->off = 1;

}

// This function activates a particular LED to the indicated color
void activateColor(struct LedNode *subject, int color) {

    if (subject != NULL) {
        int relative = channelCheck(((subject->idNumber)*3));

        if (color == 0) {
            setGSDData((relative), B[color][subject->idNumber]);
            setGSDData((relative+1), G[color][subject->idNumber]);
            setGSDData((relative+2), R[color][subject->idNumber]);
            sendGSDData();

        }

        if (color == 1) {
            setGSDData((relative), B[color][subject->idNumber]);
            setGSDData((relative+1), G[color][subject->idNumber]);
            setGSDData((relative+2), R[color][subject->idNumber]);
            sendGSDData();
        }
    }
}

```

```
}
```

```
if (color == 2) {
```

```
    setGSData((relative), B[color][subject->idNumber]);
```

```
    setGSData((relative+1), G[color][subject->idNumber]);
```

```
    setGSData((relative+2), R[color][subject->idNumber]);
```

```
    sendGSData();
```

```
}
```

```
if (color == 3) {
```

```
    setGSData((relative), B[color][subject->idNumber]);
```

```
    setGSData((relative+1), G[color][subject->idNumber]);
```

```
    setGSData((relative+2), R[color][subject->idNumber]);
```

```
    sendGSData();
```

```
}
```

```
if (color == 4) {
```

```
    setGSData((relative), B[color][subject->idNumber]);
```

```
    setGSData((relative+1), G[color][subject->idNumber]);
```

```
    setGSData((relative+2), R[color][subject->idNumber]);
```

```
    sendGSData();
```

```
}
```

```
if (color == 5) {
```

```
    setGSData((relative), B[color][subject->idNumber]);
```

```
    setGSData((relative+1), G[color][subject->idNumber]);
    setGSData((relative+2), R[color][subject->idNumber]);
    sendGSData();
}

if (color == 6) {
    setGSData((relative), B[color][subject->idNumber]);
    setGSData((relative+1), G[color][subject->idNumber]);
    setGSData((relative+2), R[color][subject->idNumber]);
    sendGSData();
}

if (color == 7) {
    setGSData((relative), B[color][subject->idNumber]);
    setGSData((relative+1), G[color][subject->idNumber]);
    setGSData((relative+2), R[color][subject->idNumber]);
    sendGSData();
}

if (color == 8) {
    setGSData((relative), B[color][subject->idNumber]);
    setGSData((relative+1), G[color][subject->idNumber]);
    setGSData((relative+2), R[color][subject->idNumber]);
    sendGSData();
}
```

```
    subject->off = 0;
    subject->current = library[color];
}
}
```

```
BYTES_VAL_T pinValues;
```

```
BYTES_VAL_T oldPinValues;
```

```
// This function was built by Jonathan Chang
```

```
/* This function is essentially a "shift-in" routine reading the
```

```
* serial Data from the shift register chips and representing
```

```
* the state of those pins in an unsigned integer (or long).
```

```
*/
```

```
BYTES_VAL_T read_shift_regs()
```

```
{
```

```
    long bitVal;
```

```
    BYTES_VAL_T bytesVal = 0;
```

```
    /* Trigger a parallel Load to latch the state of the data lines,
```

```
    */
```

```
    digitalWrite(clockEnablePin, HIGH);
```

```
    digitalWrite(ploadPin, LOW);
```

```
    delayMicroseconds(PULSE_WIDTH_USEC);
```

```
    digitalWrite(ploadPin, HIGH);
```

```
    digitalWrite(clockEnablePin, LOW);
```

```

/* Loop to read each bit value from the serial out line
 * of the SN74HC165N.
 */
for(int i = 0; i < DATA_WIDTH; i++)
{
    bitVal = digitalRead(dataPin);

    /* Set the corresponding bit in bytesVal.
    */
    bytesVal |= (bitVal << ((DATA_WIDTH-1) - i));

    /* Pulse the Clock (rising edge shifts the next bit).
    */
    digitalWrite(clockPin, HIGH);
    delayMicroseconds(PULSE_WIDTH_USEC);
    digitalWrite(clockPin, LOW);
}

return(bytesVal);
}

// This function was built by Jonathan Chang
/* Dump the list of zones along with their current status.
 */
void display_pin_values()
{
    Serial.print("Pin States:\r\n");

```

```

for(int i = 0; i < DATA_WIDTH; i++)
{
    Serial.print(" Pin-");
    Serial.print(i);
    Serial.print(": ");

    if((pinValues >> i) & 1)
        Serial.print("HIGH");
    else
        Serial.print("LOW");

    Serial.print((pinValues >> i));

    Serial.print("\r\n");
}

Serial.print("\r\n");
}

////////////////////////////////////
// The setup() method runs once, when the sketch starts

void setup ()
{
    // This function is used to start the TLC5940 chips
    initTLC5940();
}

```



```

// This starts the feed of serial data
Serial.begin(9600);

// Intialize pins for communication with the Shift Registers
pinMode(ploadPin, OUTPUT);
pinMode(clockEnablePin, OUTPUT);
pinMode(clockPin, OUTPUT);
pinMode(dataPin, INPUT);

digitalWrite(clockPin, LOW);
digitalWrite(ploadPin, HIGH);

// Read in and display the pin states at startup.

pinValues = read_shift_regs();
display_pin_values();
oldPinValues = pinValues;

return;
}

/////////////////////////////////////////////////////////////////
/// The loop() method runs over and over again, as long as the Arduino has
power

```

```

void loop()
{

// Initialize the double pointer that will be used to hold the array of nodes
int k = 80;

struct LedNode **exstar = (struct LedNode **)malloc((k*sizeof(struct LedNode
*)));

// This array is used to help calculate change in serial data from each individual
sensor.

// Holds the difference resulting from each sensor's input. Changes according to
number of LEDs in grid

int pinExclusion[80]; // IMPORTANT: Always specify the size of this array with a
number input directly. NO VARIABLES

// Failure to do so will cause unpredicable anomalies in color
output

// Initialize each individual LED node by allocating memory to them and setting
their properties.

// Calculate the value of each pinExclusion value and store in the array
for (k = 0; k < 80; k++) {
    exstar[k] = initializeN(exstar[k], k);
    pinExclusion[k] = power(2,k);
}

// This "for" loop connects the different nodes to any adjacent nodes for the
sake of making patterns easier.

// ADJUST TO GRID SIZE
for (k = 0; k < 80; k++) {

```

```

if (((k%8) < 7)) {
    exstar[k]->west = exstar[k+1];
}

if (k < 70) {
    exstar[k]->north = exstar[k+8];
}

if (k > 8) {
    exstar[k]->south = exstar[k-8];
}

if (k > 0) {
    if (((k%8) != 0)) {
        exstar[k]->east = exstar[k-1];
    }
}

}

int i = 0; // Counts number of turns game runs
int j = 0; // extra variable for double for-loop structures

// Code written from here to the end of the loop varies depending on effects
desired for display

```

```
// This activates all programmed colors on each LED in order of how they are
wired into the grid
```

```
for (i = 0 ; i < 9; i++) {
  for (j = 0; j < 8 ; j++) {
    activateColor(exstar[i],j);
    delay(500);
    deactivateColor(exstar[i]);
  }
}
```

```
// An example of the Rip class to show the ripple effect it creates
```

```
Rip(exstar[4], 2);
```

```
delay(100);
```

```
// Get the grid to display a single uniform color
```

```
for (i = 0; i < 9; i++) {
  activateColor(exstar[i], 0);
}
```

```
// NOTE: Array "library" holds COLOR values from 0-8.
```

```
// Order: Blue, Green, Red, Yellow, Pink, Purple, Orange, White, Black
```

```
// NOTE: Small changes in brightness seem to occur with Yellow and Pink
```

```
i = 0;
```

```
// For the duration of 6 pin reads, either activate or deactivate LEDs o the grid.
```

```
while (i < 12) {
```

```

/* Read the state of all zones.
*/
    pinValues = read_shift_regs();

/* If there was a change in state, display which ones changed.
*/
    if(pinValues != oldPinValues)
    {

        display_pin_values();
        oldPinValues = pinValues;

        // Use "pinCheck" when the sensor read changes.
        // Determine which sensor was triggered
        // Switch that LED

        if (pinCheck(pinExclusion[0], pinValues)) {
            SwitchUp(exstar[0]);
        }

        if (pinCheck(pinExclusion[1], pinValues)) {
            SwitchUp(exstar[1]);
        }

        if (pinCheck(pinExclusion[2], pinValues)) {
            SwitchUp(exstar[3]);
        }
    }

```

```
if (pinCheck(pinExclusion[3], pinValues)) {  
    SwitchUp(exstar[2]);  
}
```

```
if (pinCheck(pinExclusion[4], pinValues)) {  
    SwitchUp(exstar[4]);  
}
```

```
if (pinCheck(pinExclusion[5], pinValues)) {  
    SwitchUp(exstar[5]);  
}
```

```
if (pinCheck(pinExclusion[6], pinValues)) {  
    SwitchUp(exstar[6]);  
}
```

```
if (pinCheck(pinExclusion[7], pinValues)) {  
    SwitchUp(exstar[7]);  
}
```

```
if (pinCheck(pinExclusion[8], pinValues)) {  
    SwitchUp(exstar[8]);  
}
```

```
        i++;
    }
}
Serial.print("Value Reset \n");// This just tells us the loop ended
```

```
// Free the allocated memory for each node
for (k = 0; k<9; k++) {
    free(exstar[k]);
}
}
```

```
// Free the memory for the double pointer.
free(exstar);
```

// If you wish to restart from the beginning of this loop, activate a sensor, otherwise, simply turn off the setup

```
i = 1;
Serial.print("Restart?\n");
while (i > 0 ){
    pinValues = read_shift_regs();
    if(pinValues != oldPinValues)
    {
```

```
        oldPinValues = pinValues;
```

// The "hold" variable acts as a semaphore and excludes unnecessary sensor input.

```
    i--;
```

```
  }
```

```
}
```

```
  delay(POLL_DELAY_MSEC);
```

```
  //Lamp Test
```

```
}
```

```
// This function was built by Jonathan Chang
```

```
// A debugging function used for testing when LEDs are connected properly.
```

```
void LampTest()
```

```
{
```

```
  int cnt = 0;
```

```
  //Lamp Test
```

```
  for(int i = 0; i < 48; i++)
```

```
  {
```

```
    setGSData(i, 4095);
```

```
    delay(10);
```

```
    sendGSData();
```

```
    setGSData(i, 0);
```



```
delay(10);
sendGSData();
cnt++;

if(cnt == 15)
{
  cnt = 0;
  i++;
}
}

////////////////////////////////////
/// End of Code
```