Smart Sports Scoreboard

ECE 445 Design Document

Michael Manning, Max Mitchell, and Matthew Rosenbaum
Group 49
TA: Daniel Vargas
2/26/2021

1 Introduction

1.1 Objective

Smartphones have become a central part of our society in America over the past decade. As of last year, 96% of Americans owned a cell phone, and 81% of the population owned a smartphone [1]. As a result, we have become nearly dependent on using smartphones to obtain information, stay connected to social circles, and communicate. As of 2019, people have been found to check their phones an average of 96 times a day, a number that has increased by 20% over two years prior [2]. Among these occurrences, sports apps are among the most common - a Capgemini study found that 69% of sports fans utilize smartphone technology to enhance their sports following [3]. Although, this great ability has brought with it a few negatives, specifically with the exposure to blue light. Studies have found that an excessive amount of blue light exposure can lead to daytime fatigue and a disruption of one's circadian rhythms. Furthermore, blue light can cause damage to the retina, leading to macular degeneration. The same studies found that children and adolescents are most susceptible to this damage, since their eyes are not as developed [4].

Our team strives to provide fans with a fast and real-time sports experience without the use of a device that emits blue light, such as a phone or television. In order to do this, we will develop a wall-mounted LED display, interfaced with Bluetooth and Wi-Fi, that allows users to specify their favorite sports teams and display information about the team's ongoing games. The device will use a Wi-Fi chip to connect to the home's internet and scrape the score and game information from the web. There will also be a Bluetooth application that allows users to easily choose their favorite teams as soon as the scoreboard system is taken out of the box.

1.2 Background

Many media companies, such as ESPN, Yahoo!, and FOX, have developed their own sports apps within the last 10 years. These apps allow the user to check the scores or even stream the game directly within the given app. As stated according to [3], 69% of sports followers utilize these platforms to follow their teams. The ESPN app itself has 70 million downloads and 2 million daily active users [7]. Although, outside of apps, websites, and television, there are no viable products on the market that enable a user to follow their teams in a unique way.

With the ability to scrape the web or utilize an API to grab sports scores, there is an excellent opportunity to give fans a new method for following their favorite teams and scores. We plan to test our system modularly, checking that information from an actual sports game is being delivered. Accomplishing these feats will give every sports fan a unique and high-quality fandom experience without needing to pick up their phone.

1.3 Physical Design

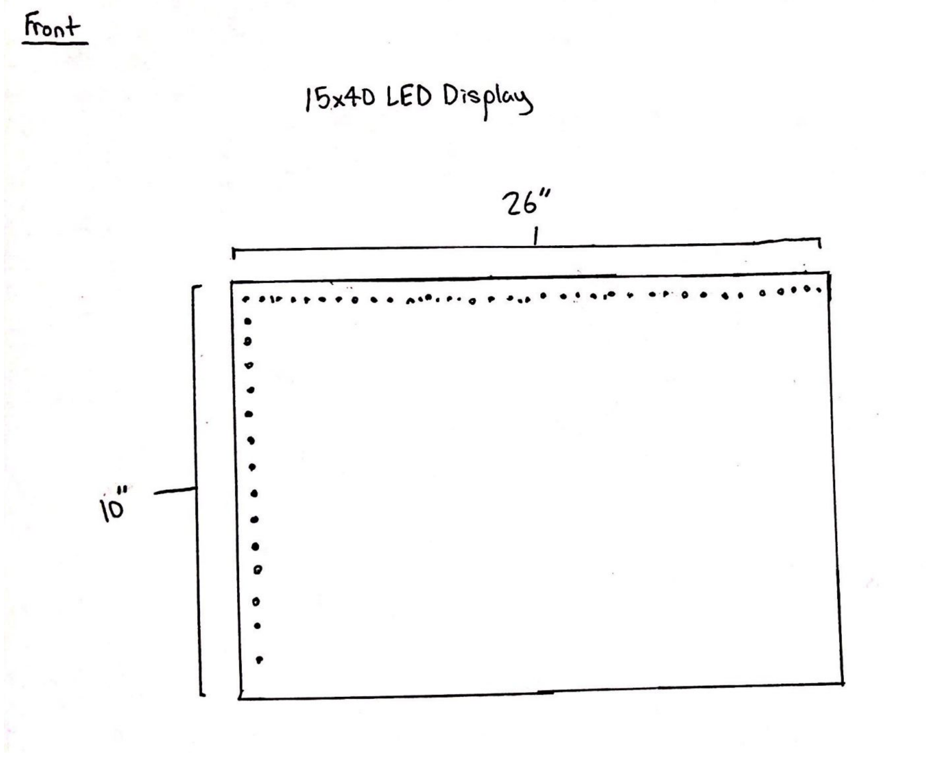


Fig. 1. Physical Design of Front of Scoreboard

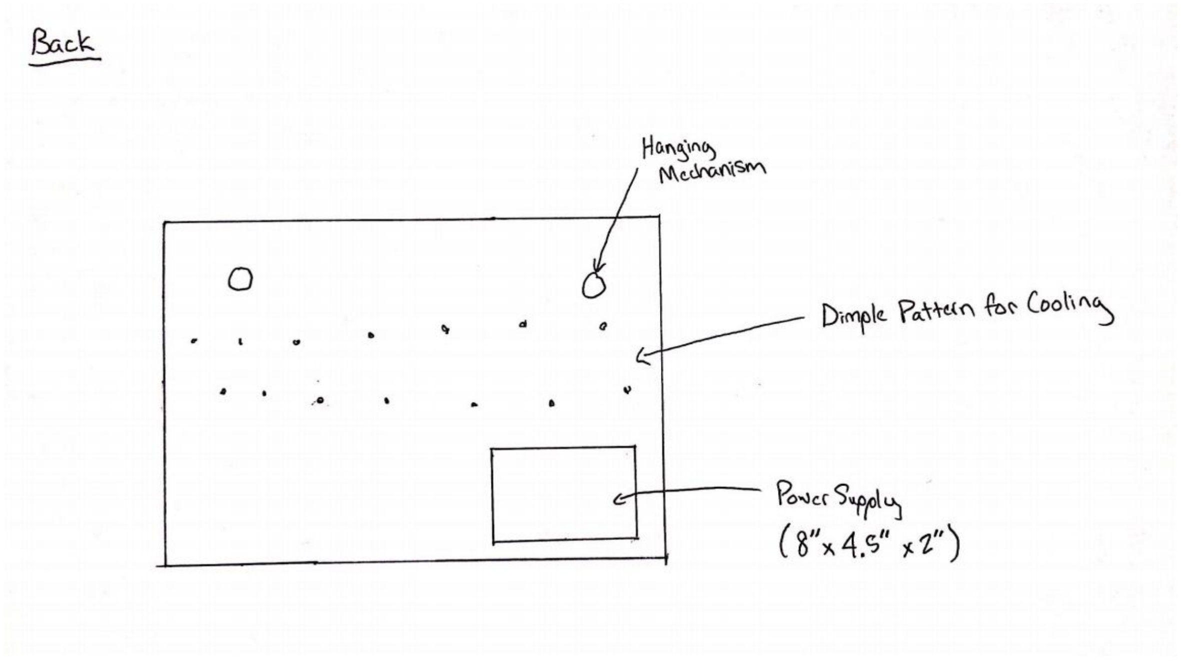


Fig. 2. Physical Design of Back of Scoreboard

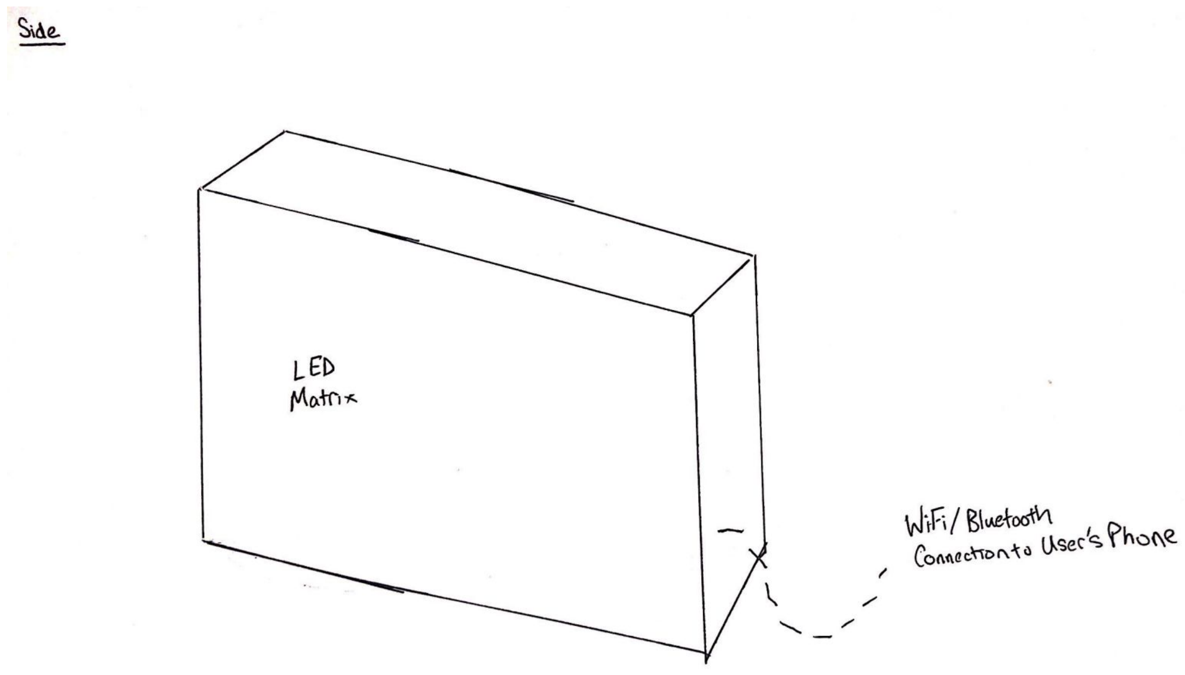


Fig. 3. Physical Design of Side of Scoreboard

After talking with the machine shop, the physical design of our scoreboard is going to be constructed of wood. Our LED Display is going to have dimensions of 26”x10” and there will be a 1” wood barrier on each side of the scoreboard to allow for access to the LED’s in order to replace or perform work on them. The power supply we plan on using for the device has dimensions of 8”x4.5”x2” and is going to be connected through the back of the device. All components within the device are going to be accessible through the back. The device is going to be hung with angle brackets, or another mechanism that may work more properly. With the current design we will have a workspace inside of the scoreboard with dimensions 3 ¾”x10”x26”. Within these dimensions we will have to store the microphone, the PCB, and other components. The machine shop stated that they need to visualize our parts before they can come up with a definitive drawing of how the device will look.

1.4 User Interface Workflow

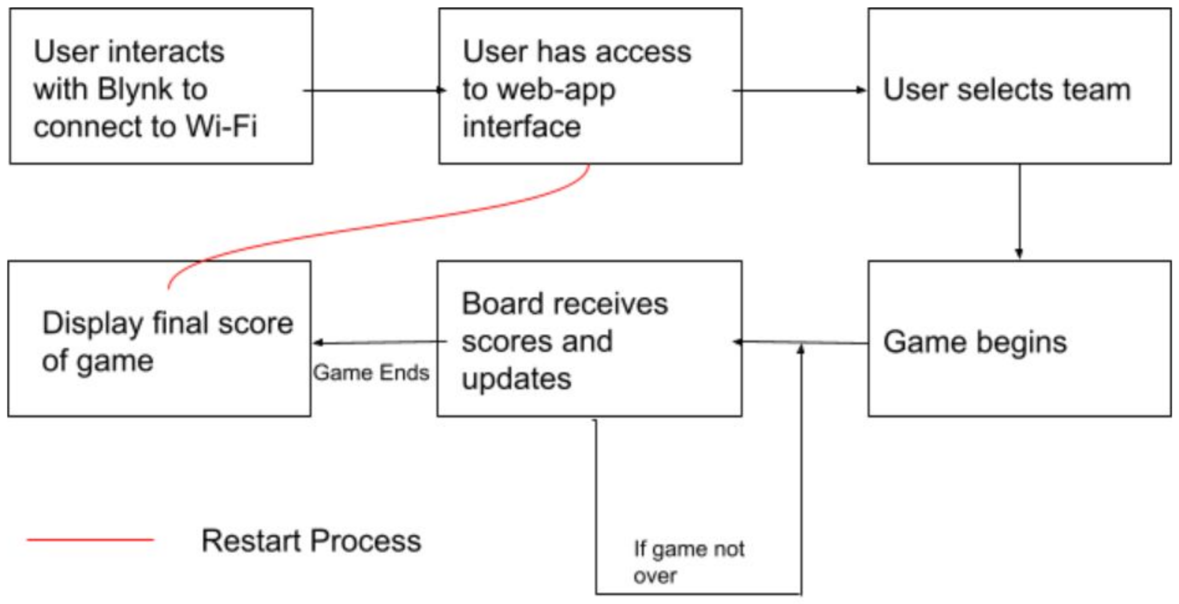

User interacts with Blynk to connect to Wi-Fi
User has access to web-app interface
User selects team
Display final score of game
Game Ends
Board receives scores and updates
Game begins
Restart Process
If game not over

Fig. 4. User Interface Workflow Diagram

Above is a diagram for the user experience. This is the ideal workflow for the user and how the device will operate starting from the first time they interact with it until the restart of the process.

1.5 High-Level Requirements

- The scoreboard operates at a latency no greater than 30s [11] with a 2.4 GHz band, and it displays the correct score of the game at run-time.
- The scoreboard system will be able to operate up to 15 amps to fit standard electrical outlets in the U.S. The Bluetooth LE protocol must transmit data at a speed of at least 1 Mb/s.
- The scoreboard system must have a UI that is easy to navigate for the user. It must allow the user to select their favorite team and customize LED interfaces using team themes.

2 Design

In our design, the ESP32 acts as the brains of the design, enabling a web interface through Wi-Fi, a Bluetooth connection to send over Wi-Fi password information, enough processing power to drive our LEDs and react to noise from the microphone in real time, and a small form factor to fit within our enclosure. Our power system is sufficient enough to power all of our LEDs and the sensors. Our enclosure, while not displayed in the block diagram, is another crucial component. It will look clean while also being light enough to be mounted on a wall.

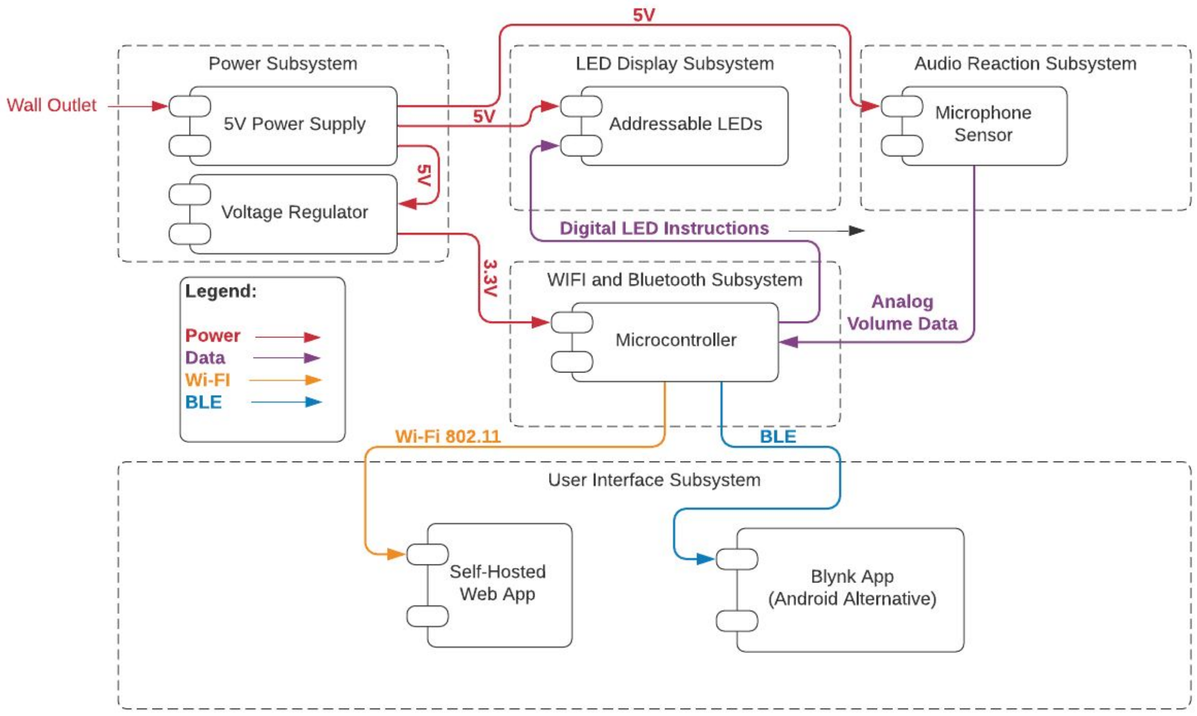


Fig. 5. Block Diagram

2.1 Power Subsystem

A power supply is required to power the LEDs, the ESP32 chip, and the microphone sensor. Both the WS2812B and ESP32 can be powered with 5V, and we will have 600 LEDs with a maximum power draw of 60mA per LED, for a total power draw of 36A. This assumes that all of our LEDs are on at max intensity, which would never be the case in our project. The ESP32 and microphone draw a negligible amount in comparison. Therefore a 15A power supply will be suitable, with the note that only 40% of our LEDs can be on at maximum power at any given time. This subsystem must interface with the LED display subsystem, audio reaction subsystem, and Wi-Fi and Bluetooth subsystem, providing constant power at 5V +/- 0.1V. If the power supply were to malfunction, the entire device would fail.

2.1.1 5V Power Supply

We will be using a power supply with a built-in AC to DC converter that plugs into a standard US wall outlet. This will allow for easy set-up for the users, who will plug the device into the wall. It will interface with the LEDs, a microphone sensor, and ESP32 to provide power.

Requirements	**Verification**
1. Supply voltage output of 5V +/- 2% power.	1. a. Probe both terminals using an oscilloscope. b. Ensure output voltage is within 2% of 5V.
2. Can support load between .036A to 15A.	2. a. Connect output of 5V power supply to VDD of constant-current test circuit from Figure 6. b. Change R_s from Figure 6 to deliver a max of 15A. Measure using a multimeter. c. Measure output voltage with an oscilloscope, ensure output voltage is within 2% of 5V.

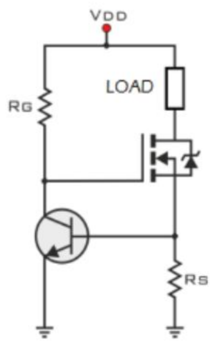


Fig. 6. Circuit Schematic for Constant Current Test Circuit where I_L = .07/R_3

2.1.2 Voltage Regulator

We will be using a linear voltage regulator to supply 3.3V to the ESP32 from the 5V AC to DC power supply. We will use the TI LM-1117 regulator to step down the voltage, ensuring that 3.3V is provided to the ESP32 while 5V is provided to the rest of the system components. The addressable LEDs and microphone sensor both require 5V.

Requirements	**Verification**
1. Supply 3.3V to ESP32 +/- 8.33% from a 5V source. Maximum voltage for ESP32 is 3.6V.	1. a. Probe both terminals using an oscilloscope. b. Ensure output voltage is within 8.33% of 3.3V.
2. Operates at a current that does not exceed 800mA.	2. a. Connect V_{OUT} from Figure 7 to VDD of constant-current test circuit from Figure 6. b. Change R_s from Figure 6 to deliver a max of 15A. Measure using a multimeter. c. Measure output voltage with an oscilloscope, ensure output voltage is within 8.33% of 3.3V.
3. Can maintain a thermal stability no greater than 125℃.	3. a. During verifications 2(a,b,c), use an IR thermometer to ensure that chip never reaches a temperature above 125℃.

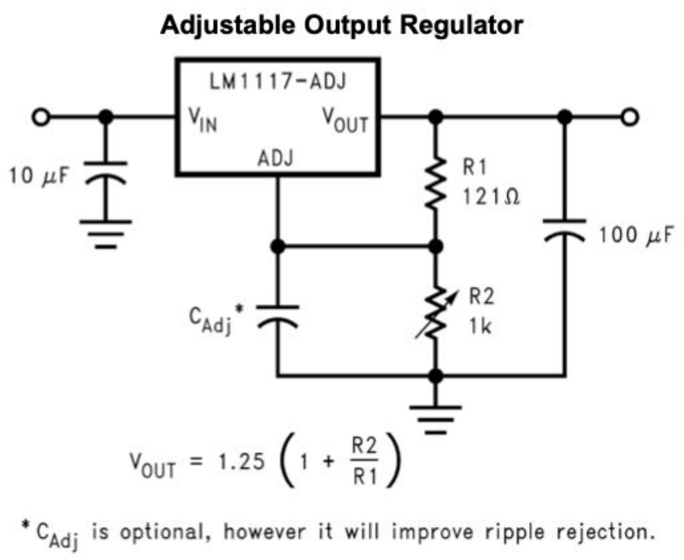

Adjustable Output Regulator
LM1117-ADJ
V_IN
V_OUT
ADJ
10 µF
R1
121Ω
100 µF
C_Adj*
R2
1k
V_OUT = 1.25 (1 + R2/R1)
*C_Adj is optional, however it will improve ripple rejection.

Fig. 7. Circuit Schematic for TI LM1117 Linear Voltage Regulator[6]

2.2 LED Display Subsystem

The LED Display will act as the main visual interface that displays the current score of the game or record of the user's favorite team. This subsystem will interface with the power supply to receive 5V power, and with the ESP32 to receive digital lighting instructions through its GPIO pins. If a portion of the LED strip were to fail, the rest of the device would function normally, but with some LEDs non-operational.

2.2.1 WS2812B Addressable LEDs

We will be using two rolls of 5 meter, 300 pixel WS2812B LEDs. This gives us 600 LEDs to work with at a resolution of about 1.5 LEDs per inch. If we do an array of 40x15, we have a dimension of ~26 inches by ~9 inches. Each roll will be cut into 7 strips of 40 and one strip of 20, giving us 14 strips of 40 and 2 strips of 20. These will be arranged horizontally to the front of the enclosure, with power connected to each horizontal strip to reduce loss from resistance. The data pins will be serial, giving us an array we can easily map to control the lights as if they are a 40x15 display.

Requirements	Verification
1. Each of the 600 LEDs can operate at 36mA and are visible from 5m away.	1. a. Deliver 36mA to load by changing R_s from the constant current circuit of Figure 6. 5V must be supplied to VDD from a 5V power source. b. Use a meter stick to measure 5m.
2. Each of the 600 LEDs are programmable by the ESP32 microcontroller. and can be set to all color gradients in hex form.	2. a. Write a test script that iterates over 600 LEDs. At each LED, set color to ten different hex color codes (for example: baby blue is #89CFF0). If ten colors are correct, then the given LED is programmable. b. Results will be kept on a grid of 600 rows and 11 columns. The first column is the LED #, and the next 10 columns correspond to the unique color.

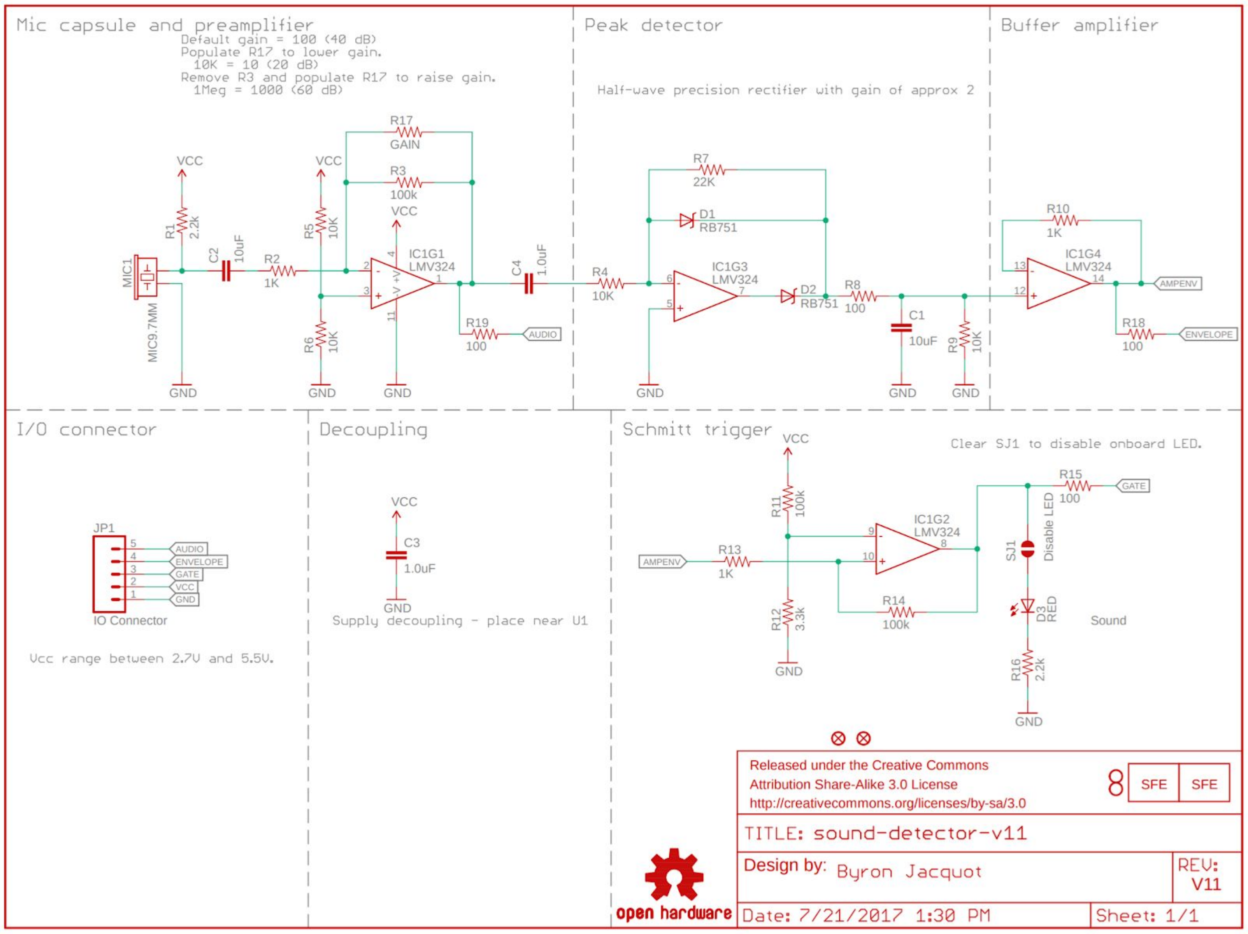
2.3 Audio Reaction Subsystem

The audio reaction subsystem makes the scoreboard an interactive highlight of the room in group settings. It is quite common for sports fans to become vocal during close times or big plays, and the scoreboard will listen to such moments and react accordingly. There may be a "sound level" setting similar to that used at some stadiums, or an animation that occurs after the volume hits a certain threshold. This subsystem interacts with the ESP32 to send the analog microphone volume information via a GPIO pin, and with the power subsystem to receive 5V power. If this subsystem were to fail, the rest of the device would function normally, just without audio reaction in the lights.

2.3.1 Microphone Sensor

We will use a SparkFun Sound Detector or create our own microphone sensor to detect the room's noise level. The sensor takes 5V power so it can be powered by the power supply. It will transmit the sensor data through one of the ESP32's GPIO pins.

Requirements	Verification
1. Must be able to differentiate between normal room talking volumes (~60dB) and yelling volumes (~80-90 dB).	1. a. Test the sensor at a baseline volume level (60dB) and record the sensor output. b. Test the sensor at a yelling volume level (80dB+) and record the sensor output. c. Ensure that there is an observable (+/- 10%) difference in the recorded sensor output.



Mic capsule and preamplifier
Default gain = 100 (40 dB)
Populate R17 to lower gain.
10K = 10 (20 dB)
Remove R3 and populate R17 to raise gain.
1Meg = 1000 (60 dB)
Peak detector
Half-wave precision rectifier with gain of approx 2
Buffer amplifier
I/O connector
IO Connector
Vcc range between 2.7V and 5.5V.
Decoupling
Supply decoupling - place near U1
Schmitt trigger
Clear SJ1 to disable onboard LED.
Disable LED
Sound
Released under the Creative Commons
Attribution Share-Alike 3.0 License
http://creativecommons.org/licenses/by-sa/3.0
TITLE: sound-detector-v11
Design by: Byron Jacquot
REV: V11
Date: 7/21/2017 1:30 PM
Sheet: 1/1
open hardware

Fig. 8. Circuit Schematic for SparkFun Sound Detector

2.4 Wi-Fi and Bluetooth Subsystem

While called the Wi-Fi and Bluetooth subsystem, this subsystem also acts as the driving microcontroller to control the lights, interpret microphone data, and interact with the user through the web application and bluetooth connection. This subsystem interfaces with every other subsystem. It interfaces with the power subsystem to receive 5V power, with the LED display subsystem to send lighting information over the GPIO pins, with the audio reaction subsystem to receive analog volume data over a GPIO pin, and with the user interface subsystem through Wi-Fi 802.11 and Bluetooth Low Energy. This is the most critical subcomponent, and any part of the ESP32 malfunctioning would likely cause the entire system to fail.

2.4.1 ESP32

We are using an ESP32 chip as it has support for both Wi-Fi and Bluetooth built in, and it has a dual core processor to handle both the connectivity and the application code. It is limited to 520 kB of on-chip SRAM, but has support for external Flash and SRAM should our code be larger. Most modules contain an external Flash built in. This interfaces with the power supply for 5V power, the lights to control them via a GPIO pin, and the microphone sensor to read input through a GPIO pin. It also interacts with the user interface subsystem by hosting a local web application through Wi-FI, and retrieving Wi-Fi password information through Bluetooth.

Requirements	Verification
1. Is powered properly by the voltage regulator at 2.2-3.6V.	1. a. Connect the chip to the 3.3V output of the voltage regulator. b. Verify that the chip turns on and can perform a simple script (turning an LED on and off).
2. Has Bluetooth compatibility (Bluetooth Low Energy).	2. a. Create a simple BLE script for testing. b. Connect a phone to the chip using Blynk App. c. Verify the connection works by sending some info across.
3. Has Wi-Fi compatibility (802.11n).	3. a. Create a simple Wi-Fi script for testing. b. Connect to the Wi-Fi and ping a test server. c. Verify the connection was successful.
4. Can properly drive the LEDs through one of the GPIO pins.	4. a. Create a script to drive the 600 LEDs with some colors. b. Run the script and verify that all the LEDs light up properly.

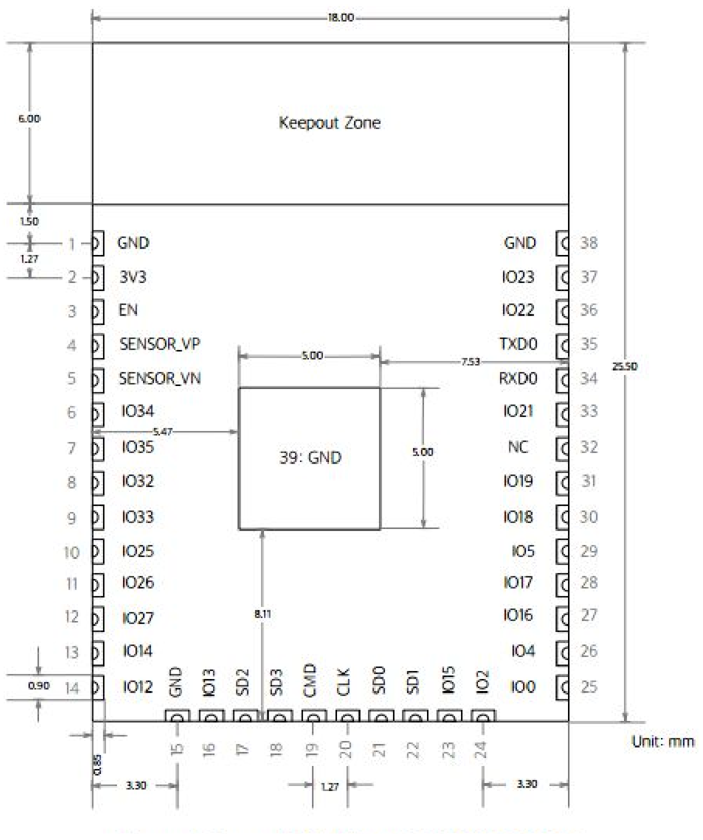


Fig. 9. ESP32 Pin Layout [10]

2.5 User Interface Subsystem

This subsystem is fully in software, but will be the only way the user interacts with the device to change their favorite team or control the visual settings. There will be two user interfaces, the main one is a web application connected to the ESP32 over Wi-Fi to control the settings, and the second one is through a phone app called Blynk connected to the ESP32 over Bluetooth Low Energy to retrieve the Wi-Fi password for the network it will be connected to. If this subsystem fails, the user would no longer be able to connect to new Wi-Fi networks or change their favorite team, but if the password is already entered the rest of the device would continue to function properly.

2.5.1 Web Application

We will create an intuitive web interface for users to communicate with the device. This is where users will tell the device who their favorite teams are, and what visual configuration they want. Some options could include a moving gradient of lights, a scoreboard for when games are in progress, or the current record of the team. This block interfaces with the ESP32 chip's Wi-Fi 802.11 protocol and the user's phone or computer browser.

Requirements	Verification
1. The user must be able to select from all available teams from the following six sports: NFL, NBA, MLB, NHL, College Basketball, and College Football.	1. a. Write software for the web app that properly parses the API response containing the list of teams, storing the teams alphabetically in the form of a list. b. One-by-one, select one team from each sport listed. The scoreboard must display the final score of each team's previous game.
2. The user must be able to toggle between four customization modes: display the current score (if there is a game in progress), most recent score, team's record, or a moving gradient of lights that are the color of the team.	2. a. There will be a dropdown menu that allows the user to select between modes. Select each mode, one-by-one, ensuring that each mode displays accordingly on the LEDs. b. Repeat step (a) for all six sports for quality assurance.
3. The web app must have a latency no greater than 500-750ms. This ensures that a change made on the web app will be displayed immediately on the scoreboard.	3. a. Open the web app running locally on a computer. Open the web inspector by right clicking. b. In the web inspector window and select the 'Timelines' tab if using Safari. Select the 'Network' tab if using Chrome. c. Make a customization change, as explained i n (2). Note the length of the request in the web inspector. d. Repeat step (c) for all four customization modes, for all six sports.

2.5.2 Bluetooth Connection

We will utilize a Bluetooth connection using the ESP32's Bluetooth Low Energy (BLE) to receive the user's Wi-Fi password information. We will utilize Blynk, an iOS app that allows users to create custom UI's and easily connect to Bluetooth enabled microcontrollers suchs as the ESP32. Blynk will allow our group to overcome Android limitations.

Requirements	**Verification**
1. Data must be transmitted at a 1 Mb/s speed.	1. a. Write a test script that interfaces with the ESP32 microcontroller. b. Simulate data being sent and print the data transmission speed on the serial monitor [5].
2. Connection to the 802.11n Wi-Fi network must be successful after first attempt using the Blynk app.	2. a. Simulate taking the scoreboard out of the box as if it is brand new by bringing it to a place where it has never connected to the Wi-Fi network before. b. Enter Wi-Fi network name and password via Blynk. c. Write a test script with a set team (i.e, Illinois Basketball) and get the latest score using the ESPN API. Display it serially or on the LED matrix.

2.6 Board Layout & Schematics

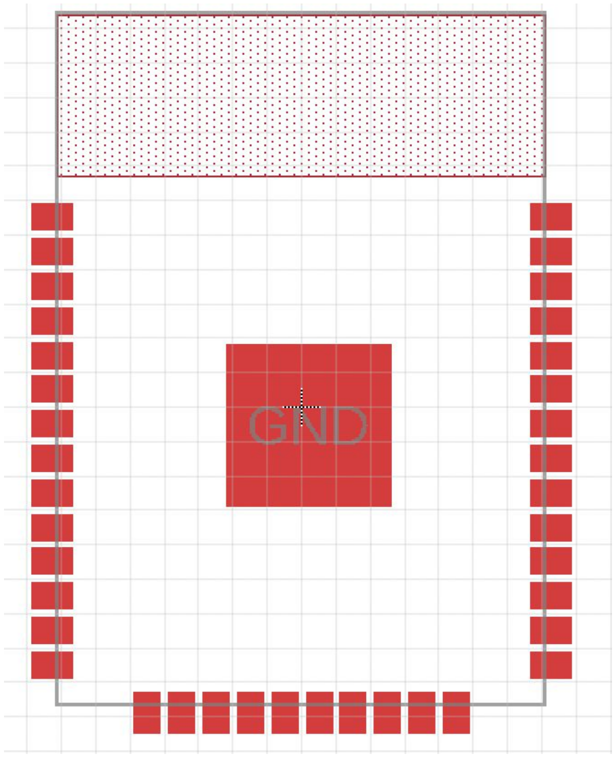


Fig. 11. ESP32 Footprint [12]

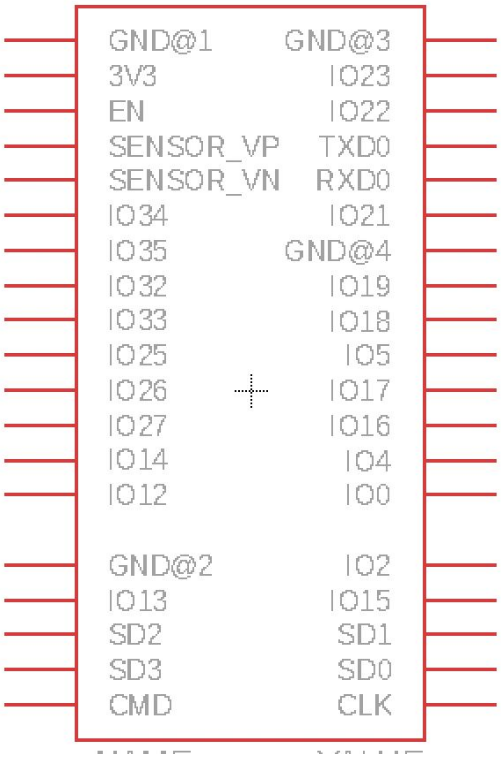


Fig. 12. ESP32 Symbol [12]

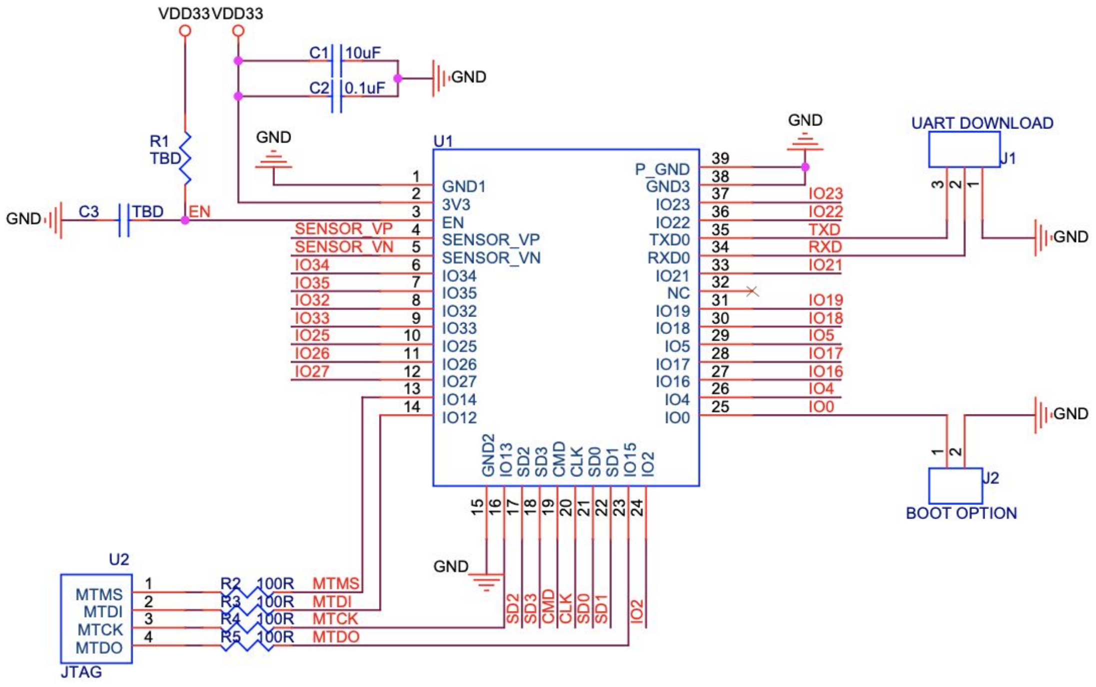


Fig. 13. ESP32 Dev-Kit Schematic [10]

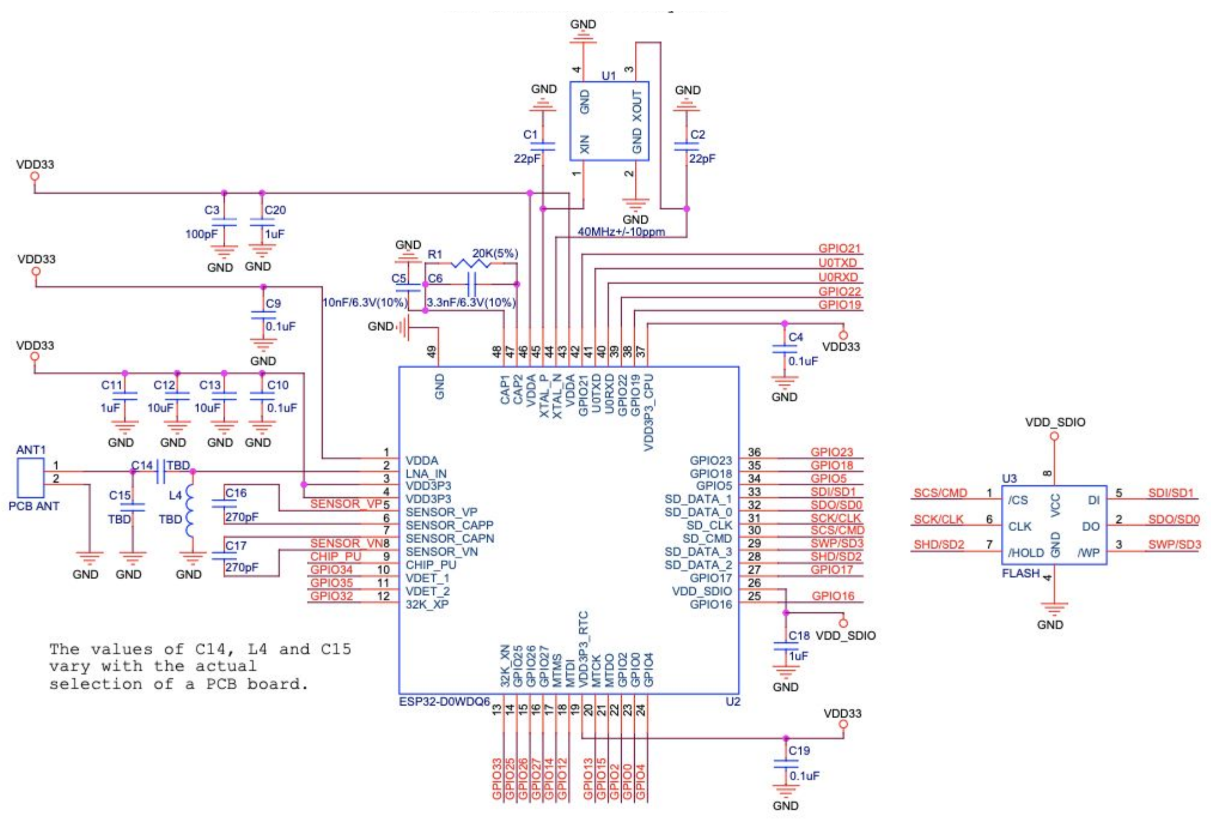

ESP32-D0WDQ6
U1
U2
U3
FLASH
40MHz+/-10ppm
The values of C14, L4 and C15 vary with the actual selection of a PCB board.
PCB ANT
VDD33
VDD_SDIO
GND

Fig. 14. ESP32 Schematic [10]

2.7 Software

The software is critical for data transmission between the scoreboard's subsystems. For the scoreboard to work as intended, the software must be working properly behind the scenes. Its first purpose is to scrape ESPN's APIs and parse the response bodies for the essential data. Once the data is stored, the software will use it to drive the LED system and display correct, up-to-date information. Figure 15 displays the flow of our software system, starting with the user selecting a sport and team and ending with the microcontroller driving the LED system with the correct information.

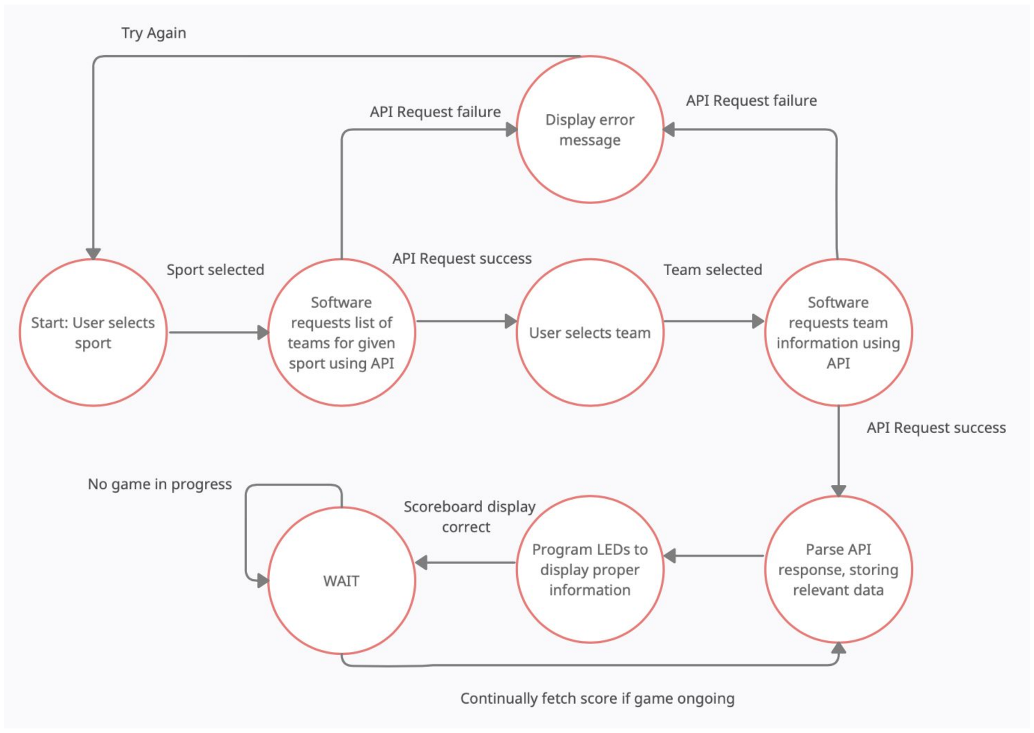
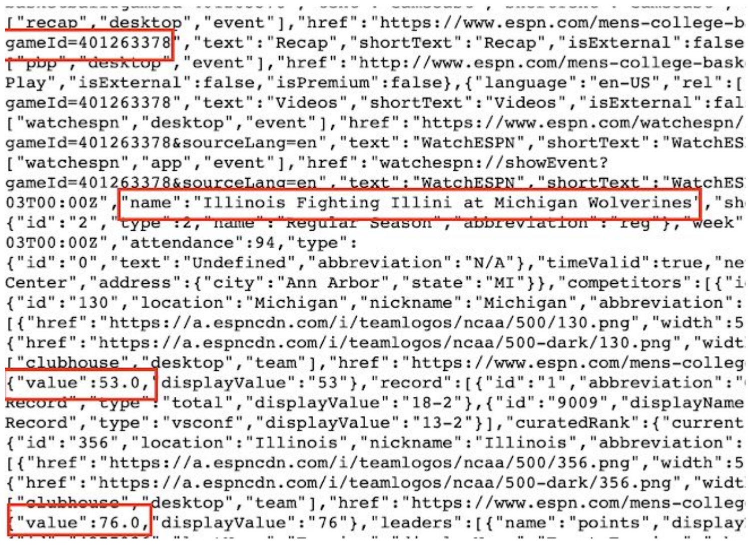

Try Again
API Request failure
Display error message
API Request failure
Sport selected
API Request success
Team selected
Start: User selects sport
Software requests list of teams for given sport using API
User selects team
Software requests team information using API
API Request success
No game in progress
Scoreboard display correct
WAIT
Program LEDs to display proper information
Parse API response, storing relevant data
Continually fetch score if game ongoing

Fig. 15. Software Process Flow

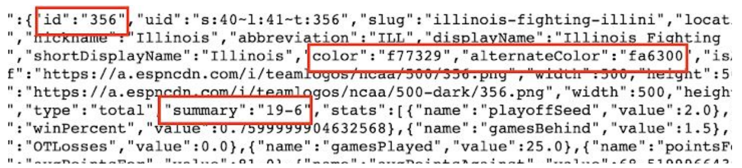
2.7.1 Sports Data Scraper

The API endpoints provide a comprehensive list of teams and their associated team IDs for the six sports listed in Section 2.5.1. The specific ID can be used to reach two other endpoints designated for the given team. The first endpoint is shown in Figure 16, and it has all of the results of the past games. The second endpoint, shown in Figure 17, lists the team's upcoming game, as well as their record and team colors. The first challenge in this section will be parsing the JSON file for the proper information. As stated in Section 2.5.1, there are four different customization modes. Thus, when parsing the API, we will store all information that can possibly be displayed on the scoreboard. The second challenge will be error handling. We cannot control if one of ESPN's endpoints is failing to display information, but we do not expect it to occur on a daily basis. On the off-chance that an error were to occur, our software must be responsible for relaying a simple error message to the user. An example would be "Error retrieving team information! Please try again later.", with a link to return to the homepage.



```
["recap","desktop","event"],"href":"https://www.espn.com/mens-college-b
gameId=401263378","text":"Recap","shortText":"Recap","isExternal":false
["pbp","desktop","event"],"href":"http://www.espn.com/mens-college-bask
Play","isExternal":false,"isPremium":false},{"language":"en-US","rel":[
gameId=401263378","text":"Videos","shortText":"Videos","isExternal":fal
["watchespn","desktop","event"],"href":"https://www.espn.com/watchespn/
gameId=401263378&sourceLang=en","text":"WatchESPN","shortText":"WatchES
["watchespn","app","event"],"href":"watchespn://showEvent?
gameId=401263378&sourceLang=en","text":"WatchESPN","shortText":"WatchES
03T00:00Z","name":"Illinois Fighting Illini at Michigan Wolverines","sh
{"id":"2","type":2,"name":"Regular Season","abbreviation":"reg"},"week"
03T00:00Z","attendance":94,"type":
{"id":"0","text":"Undefined","abbreviation":"N/A"},"timeValid":true,"ne
Center","address":{"city":"Ann Arbor","state":"MI"}},"competitors":[{"i
{"id":"130","location":"Michigan","nickname":"Michigan","abbreviation":
[{"href":"https://a.espncdn.com/i/teamlogos/ncaa/500/130.png","width":5
{"href":"https://a.espncdn.com/i/teamlogos/ncaa/500-dark/130.png","widt
["clubhouse","desktop","team"],"href":"https://www.espn.com/mens-colleg
{"value":53.0,"displayValue":"53"},"record":[{"id":"1","abbreviation":"
Record","type":"total","displayValue":"18-2"},{"id":"9009","displayName
Record","type":"vsconf","displayValue":"13-2"}],"curatedRank":{"current
{"id":"356","location":"Illinois","nickname":"Illinois","abbreviation":
[{"href":"https://a.espncdn.com/i/teamlogos/ncaa/500/356.png","width":5
{"href":"https://a.espncdn.com/i/teamlogos/ncaa/500-dark/356.png","widt
["clubhouse","desktop","team"],"href":"https://www.espn.com/mens-colleg
{"value":76.0,"displayValue":"76"},"leaders":[{"name":"points","display
```

Fig. 16. Game Scores Endpoint for Illini Basketball

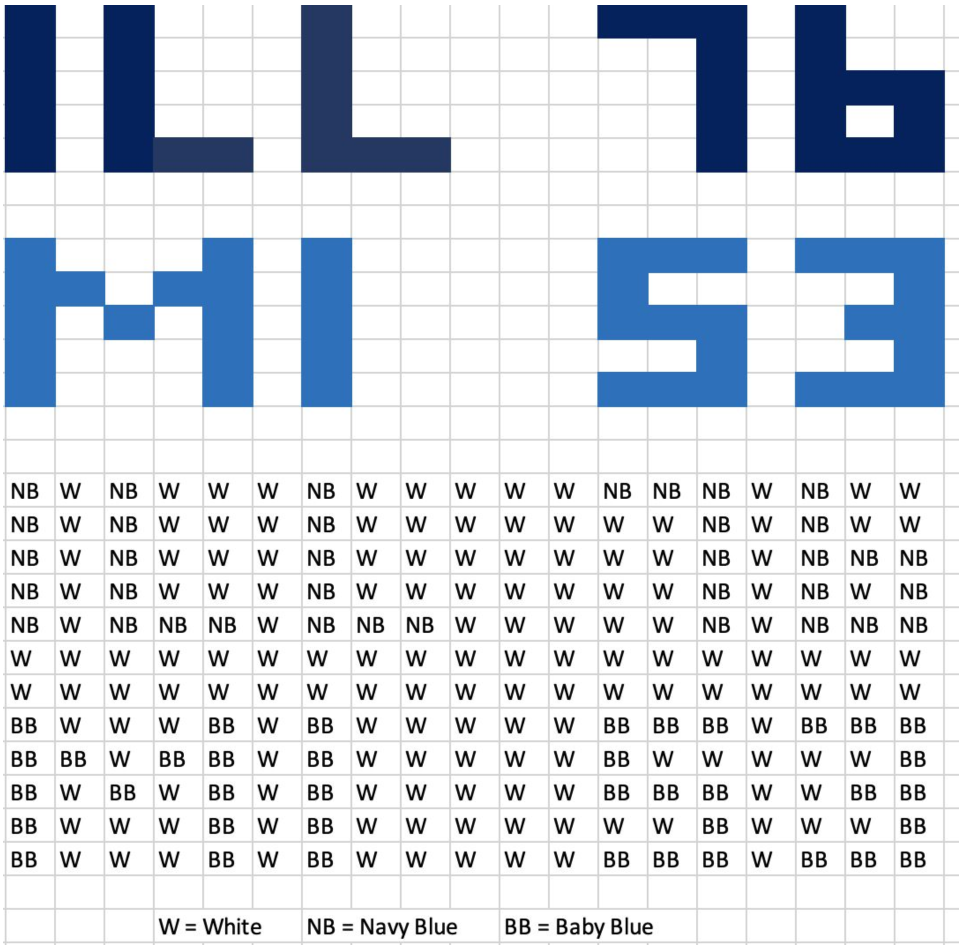


```
":{"id":"356","uid":"s:40~l:41~t:356","slug":"illinois-fighting-illini","locat
","nickname":"Illinois","abbreviation":"ILL","displayName":"Illinois Fighting
","shortDisplayName":"Illinois","color":"f77329","alternateColor":"fa6300","is
f":"https://a.espncdn.com/i/teamlogos/ncaa/500/356.png","width":500,"height":5
":"https://a.espncdn.com/i/teamlogos/ncaa/500-dark/356.png","width":500,"heigh
","type":"total","summary":"19-6","stats":[{"name":"playoffSeed","value":2.0},
":"winPercent","value":0.7599999904632568},{"name":"gamesBehind","value":1.5},
":"OTLosses","value":0.0},{"name":"gamesPlayed","value":25.0},{"name":"pointsF
```

Fig. 17. Team Information Endpoint for Illini Basketball

2.7.2 LED Driver

Once the data is obtained from the API, it must be used to program the LEDs. The LED subsystem will be connected to the scoreboard's microcontroller, ESP32. There will be a piece of software flashed to the ESP32 that has an algorithm to efficiently iterate over the entire LED matrix. Instead of turning an LED on once at a time, the algorithm will store each LED's color in an array of length 600. Once each array element is set, the LEDs will be changed simultaneously to their new values. Figure 18 shows an example display on the scoreboard and corresponding array configuration. The example is not drawn to scale, but the configuration will be the same for a 40x15 LED matrix.



NB	W	NB	W	W	W	NB	W	W	W	W	W	NB	NB	NB	W	NB	W	W
NB	W	NB	W	W	W	NB	W	W	W	W	W	W	W	NB	W	NB	W	W
NB	W	NB	W	W	W	NB	W	W	W	W	W	W	W	NB	W	NB	NB	NB
NB	W	NB	W	W	W	NB	W	W	W	W	W	W	W	NB	W	NB	W	NB
NB	W	NB	NB	NB	W	NB	NB	NB	W	W	W	W	W	NB	W	NB	NB	NB
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
BB	W	W	W	BB	W	BB	W	W	W	W	W	BB	BB	BB	W	BB	BB	BB
BB	BB	W	BB	BB	W	BB	W	W	W	W	W	BB	W	W	W	W	W	BB
BB	W	BB	W	BB	W	BB	W	W	W	W	W	BB	BB	BB	W	W	BB	BB
BB	W	W	W	BB	W	BB	W	W	W	W	W	W	W	BB	W	W	W	BB
BB	W	W	W	BB	W	BB	W	W	W	W	W	BB	BB	BB	W	BB	BB	BB

W = White NB = Navy Blue BB = Baby Blue

Fig. 18. Example LED Display and Array Configuration

2.10 Tolerance Analysis

The most important tolerances we must maintain are the ESP32 chips processing speeds to parse the API data and limiting the latency to drive the 600 LEDs in real time stutter free. The ESP32 utilizes a dual-core Xtensa LX6 processor, with both cores running at 240MHz. The ES2812B LEDs' latency depends on the length of the led array. Each LED requires 3 Bytes of data (1 for each color channel), giving us a total memory requirement of:

$$3 \text{ (Color channels)} * 600 \text{ (LEDs)} = 1800 \text{ Bytes} = 14400 \text{ Bits} \qquad \text{Eq. 1}$$

The ESP32 has 520kB of SRAM, so this is not an issue. Looking at the datasheet, the transmission time takes 1.25µs per bit, and 50µs for the reset, for a total transmission time of:

$$14400 \text{ (bits)} \times 1.25\mu s + 50\ \mu s = 18.05 \text{ ms} = 55.4 \text{ fps} \qquad \text{Eq. 2}$$

Most modern monitors use either 30fps or 60fps with some high end gaming monitors running at 144fps [13]. Our LEDs will not be displaying high speed games, so 55.4fps falls within the acceptable range of real time response rates. If needed, we could use multiple GPIO pins and split the data to shorter segments, but that will not be necessary with this number of LEDs. This would help reduce the length of data propagation needed from the full 600 LEDs to a smaller amount like 100 if we used 6 GPIO pins, reducing the total transmission time.

As part of the ESP32 documentation, one of the processors manages the Wi-Fi and Bluetooth while the other manages the application code. Working backwards from the 18.05ms cycle time, it is easy to calculate the maximum number of operations we can perform per refresh on each of the CPU cores. The ESP32 runs at either 160MHz or 240MHz, so we can analyze both scenarios:

$$240 \text{ MHz} * 1000000 \text{ Hz / MHz} * 18.05 \text{ ms} * 1000 \text{ s/ms} =$$
$$y4.3 \times 10^{12} \text{ Operations per Cycle Time} \qquad \text{Eq. 3}$$
$$160 \text{ MHz} * 1000000 \text{ Hz / MHz} * 18.05 \text{ ms} * 1000 \text{ s/ms} =$$
$$2.88 \times 10^{12} \text{ Operations per Cycle Time} \qquad \text{Eq. 4}$$

This is a very large number of operations, but is failing to consider the Wi-Fi latency we may experience. We can reformat the equation with a variable ℓ to account for latency in ms.

$$2.4 \times 10^{11} (18.05 \text{ ms} - \ell) \text{ Operations @ 240 MHz} \qquad \text{Eq. 5}$$
$$1.6 \times 10^{11} (18.05 \text{ ms} - \ell) \text{ Operations @ 160 MHz} \qquad \text{Eq. 6}$$

As long as the latency is less than 18.05ms, there will be time for operations to occur in real time. Latency can commonly go above that, but that is okay given our requirements. We only need the score and records to update with a latency of up to 30s, so as long as we interrupt the processor to drive the LED animations at a consistent rate of up to 55.4fps, we have 30s of leniency to parse the sports API data and decide if any update to the scoreboard is necessary.

3 Costs and Schedule

3.1 Costs

As shown below, our costs are broken down into two categories, labor and components. The grand total, $30,124.45**.** is composed of the sum of the labor costs and the component costs.

3.1.1 Labor Costs

The labor costs of the project are broken down on a person by person basis and summed up at the end to give us the total cost of labor for the group. We assumed a rate of $40/hour for the project as there is a large software engineering component to our work. We assumed that for the rest of the 10 weeks, we will all be contributing about 10 hours/week to the project. Using these statistics we came up with our total labor costs to be $30,000.

Partner	Hourly Rate	Hours/Week	Weeks	2.5x Multiplier	Total/Person
Michael	40	10	10	2.5	$10,000
Max	40	10	10	2.5	$10,000
Matthew	40	10	10	2.5	$10,000

Total Labor Costs: $30,000

3.1.2 Component Costs

The sheet below contains all of the components we need and the associated costs with them.

Part Name	Description	Manufacturer	Part number	Quantity	Cost per part
AC/DC Power Adapter	5V 15A AC to DC Power Supply Adapter Transformer Converter	ALITOVE	5V-15A-BK	1	$28.99
ESP32 Microcontroller	ESP32S Dev Board with WiFi + Bluetooth Dual Cores Microcontroller	MELIFE	B07Q576VWZ	1	$14.99
SparkFun Sound Detector	Microphone that is going to be used for audio	SparkFun Electronics	SEN-12642	1	$10.49
Individually Addressable LED Strip Light	LED's that are going to be used for LED matrix	ALITOVE	812B300WPBK-FBA	2	$34.99
Breadboard	Breadboard for testing and designing of circuits	ECE shop	N/A	1	Previously acquired
Assorted Resistors	Resistor set for building of circuits	ECE shop	Varies	1	Previously acquired
Assorted Capacitors	Capacitor set for building of circuits	ECE shop	Varies	1	Previously acquired

Total (Pre-tax): $124.45

3.2 Schedule

Week	Michael	Max	Matthew
3/1/2021	Select and order parts	Refine PCB	Acquire methods for obtaining software solutions
3/8/2021	Work with machine shop to finalize UI friendly design	Further Refine PCB with Matthew, place order end of week	Further Refine PCB with Max, place order end of week
3/15/2021	Receive power subsystem and verify requirements associated	Finalize API selections to be used in construction	Begin construction of web application
3/22/2021	Assemble 40x15 LED matrix and verify functionality of each LED	Ensure microphone's connection to system	Continue construction of web application
3/29/2021	Begin logic for addressable LED's	Establish connectivity between ESP32 and Blynk App	Continue construction of web application
4/5/2021	Continue logic for addressable LED's	Establish connectivity between ESP32 and Wi-Fi network	Ensure connection between addressable LED logic and web application
4/12/2021	Verify LED matrix logic	Incorporate audio subsystem into design	Finalize software and test on LED matrix
4/19/2021	Prepare mock demo	Prepare mock demo	Prepare mock demo
4/26/2021	Solder PCB components together after verification	Verify solder connections	Finalize web app and user experience within
5/3/2021	Finalize mechanical components and mounting structure	Clean UI on web-app	Clean UI on web-app
5/10/2021	Prepare Final Report	Prepare Final Report	Prepare Final Report

4 Ethics and Safety

4.1 Safety

The Smart Sports Scoreboard is intended to be used in the homes of sports fans. Indoor use mitigates some risks, but safety issues are still capable of arising due to a number of unforeseen circumstances that could occur within a user's home. This device has been designed with OSHA standards in mind [8]. All energized electrical components will be properly enclosed or isolated such that the user can interact with the device and have no harm to themself. In creating an enclosure, insulators will be used so that electric current can not flow through the device and into the user. Failure to account for this could, although improbably, result in shock, electrocution, or burns. The web interface also allows for the user to operate the device while not having direct contact with the Smart Sports Scoreboard. This component adds another level of safety for the user and their experience.

4.2 IEEE Ethics Accordance

We as a group, in accordance with the IEEE Code of Ethics, understand that it is our responsibility to commit ourselves to the highest ethical and professional standards in creating this device. In particular, our device is responsible for “hold(ing) paramount the safety, health, and welfare of the public” [11]. As a Smart Sports Scoreboard, further versions of the product may include sports gambling features so that users may stay up to date with betting odds on games that they are following. Any sports betting features do not have the ability to make wagers. These features are specifically for entertainment purposes. We as a group only condone sports gambling where it is done legally in states that allow it, and only when it is done responsibly by the individuals making the wagers.
Our device also strives “to treat all persons fairly and with respect” [11]. The purpose of our device is to provide real-time information in a visually appealing way to the user, so that they may spend less time on their smart devices and more time honed in on what is happening around them. We, in no way, intend to alter the information the user receives, and only hope to provide the user with true information. Therefore, we strive to treat all persons, teams, and players being represented on our device fairly and with the utmost respect.

It is our responsibility as a group to hold each other accountable in “striv(ing) to ensure this code is upheld by colleagues” [11]. We, in accordance with the code of ethics, will support our teammates and continually follow up with each other to ensure that we uphold conduct of the highest standard. If a member of the group is to behave unethically, no retaliation will be present against individuals who report a violation. It is our responsibility in creating a device to make sure that positive value is provided to the world, and this is what we plan on doing.

References

[1] Pew Research Center, *Mobile Fact Sheet*, Pew Research Center, June 12, 2019. Accessed on February 15, 2021. [Online]. Available:https://www.pewresearch.org/ internet/fact-sheet/mobile/

[2] Asurion, *Americans Check Their Phones 96 Times a Day*, Asurion, November 21, 2019. Accessed on February 14, 2021. [Online]. Available: https://www.asurion.com/ about/press-releases/americans-check-their-phones-96-times-a-day/

[3] C. Gough, *Sports fans using mobile apps worldwide 2019, by age*, Statista, March 20, 2020. Accessed on February 14, 2021. [Online]. Available: https://www.statista.com/ statistics/1100567/sports-content-mobile-apps/

[4] F. Yanoga, *Does blue light from electronic devices damage your eyes?*, Wexner Medical Center, June 13, 2019. Accessed on March 3, 2021. [Online]. Available: https://wexnermedical.osu.edu/blog/blue-light-and-vision

[5] Random Nerd Tutorials, *ESP32 Bluetooth Low Energy (BLE) on Arduino IDE*, Random Nerd Tutorials, June 4, 2019. Accessed on March 3, 2021. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/

[6] Texas Instruments, *LM1117 800-mA, Low-Dropout Linear Regulator*, LM1117 datasheet, February 2000. [Online].Available: https://www.ti.com/lit/ds/symlink /lm1117.pdf?ts=1614878932564&ref_url=https%253A%252F%252Fwww.ti.com%25 2Fstore%252Fti%252Fen%252Fp%252Fproduct%252F%253Fp%253DLM1117T-3.3%2 52FNOPB%2526keyMatch%253DLM1117-3%2B3%2526tisearch%253DSearch-EN-ev erything

[7] K. Draper, *ESPN Tries to Get With a Mobile, App-Driven World*, The New York Times, April 12, 2018. Accessed on February 15, 2021. [Online]. Available: https://www.nytimes.com/2018/04/12/sports/espn-app.html

[8] OSHA, "CONSTRUCTION SAFETY & HEALTH." [Online]. Accessed: 01-Mar-2021. Available: https://www.osha.gov/sites/default/files/2018-12/ fy07_sh-16586-07_4_electrical_safety_participant_guide.pdf.

[9] IEEE.org, *IEEE Code of Ethics,* IEEE, 2021. Accessed on February 13, 2021. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[10] Espressif Systems, *ESP32 Series*, ESP32-WROOM-32 datasheet, Jan 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[11] H. Regan, *Spoiler Alert: Low Latency Crucial for Sports Apps*, Wowza Media Systems, May 22, 2017. Accessed on March 3, 2021. [Online]. Available: https://www.wowza.com/blog/spoiler-alert-low-latency-crucial-for-sports-apps

[12] L. Podkalicki, *eagle-libraries*, ver 1.7. Accessed on March 4, 2021. [Online]. Available: https://github.com/lpodkalicki/eagle-libraries

[13] B. Stegner, "Do Monitor Refresh Rates Matter? Everything You Need to Know," *MUO*, 12-Feb-2021. [Online]. Available: https://www.makeuseof.com/tag/60hz-vs-144hz/. [Accessed: 05-Mar-2021].