MTD BUS TRACKING DISPLAY

Design Document

By

Amber Wilt, Daniel Vlassov, Ziad Dohaim

Design Document for ECE 445, Senior Design, Fall 2025

6 October 2025

Project No. 4

Professor: Arne Fliflet

TA: Wesley Pang

Contents

1 Introduction

1.1 Problem

Champaign has an extensive and complex bus system that is administered by the Champaign-Urbana Mass Transit District (MTD), which serves both the UIUC campus and the surrounding city. With so many buses, routes, schedules, and transfer points scattered throughout the city/campus, it can be extremely challenging for students to easily plan and navigate their daily commutes. This challenge can become significantly difficult for students who have to travel large distances between classes, dormitories, or off-campus housing in a limited timeframe between classes.

Although the MTD does provide real-time tracking information, this data is typically only displayed on electronic screens located at major bus stops. These screens show the estimated arrival times for upcoming buses, but are only accessible once a student has already reached that stop. In many cases, a student may have to travel 5-10 minutes from instructional buildings to reach a specific stop, meaning they may only discover they've missed the bus by the time they've reached the stop.

While some apps exist to assist in planning, students often run into issues that limit their reliability and effectiveness. These may include connectivity problems, device power, and hard-to-follow user interfaces that can make these apps more frustrating than helpful, sometimes causing students to miss their desired bus. For example, when opening an app to check arrival times, the display may only show the nearest stop without clearly indicating which direction the bus is heading. In other cases, the lack of route comparison tools forces students to sort through options manually.

1.2 Solution

To address the limitations of the current bus-tracking methods, we propose the design of a large 3D-printed display model that shows the real-time locations of all buses in the surrounding campus area. The model would feature a physical map of Champaign-Urbana and the campus area, with addressable RGB LEDs to represent different bus lines. Each bus would be color-coded, and its position would be updated approximately every 30 seconds via the publicly available MTD API. To further improve usability, the system would periodically light up the entire path of each bus, making it easier for students to identify which route they need to take. Additional customization features, such as adjustable themes, brightness levels, and controllable route display (user chooses to light up the entire route), would allow the display to be used effectively in a variety of lighting environments.

This 3D-modeled implementation offers several advantages over simply showing bus information on a digital screen. Traditional screens often present data in lists or small maps that can feel abstract and overwhelming, requiring users to interpret bus numbers, directions, and estimated arrival times within a short period. In contrast, the 3D-printed display provides an intuitive and easy-to-follow visualization of bus movement, making it immediately clear where a bus is relative to the student's building or destination. The 3D model also makes it more engaging and accessible, particularly in busy, communal spaces where many students may take a glance without needing to navigate through an app. By combining real-time data with a physical, visual representation, this approach reduces confusion, reduces the time needed for students to plan trips, and diminishes the risk of students missing important events in their tight schedules.

1.3 Visual Aid

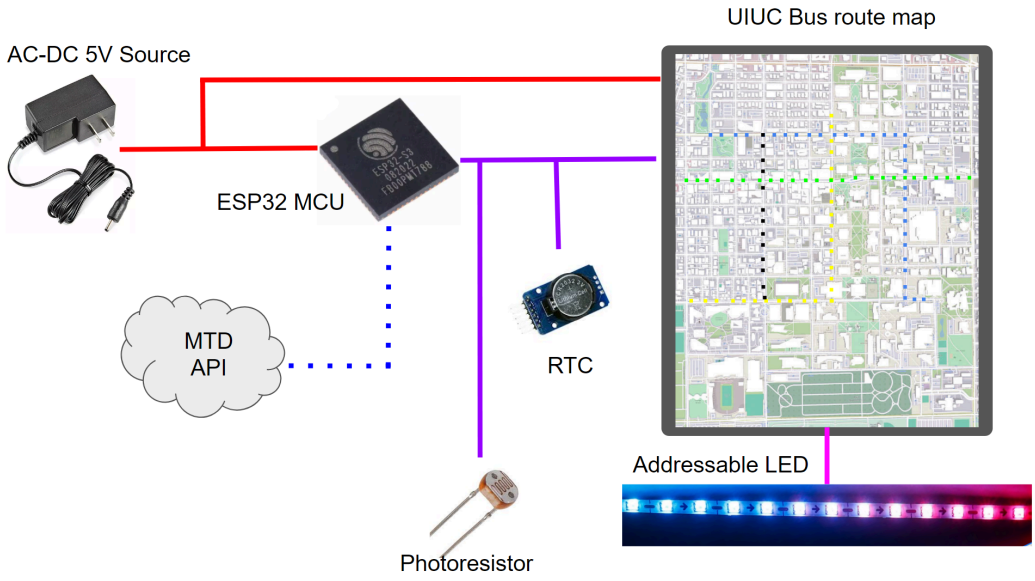

Figure 1 - Visual Aid of Design

1.4 High-Level Requirements

Three high-level requirements that can demonstrate the success of our project in solving the described problem include:

1. Real-Time Accuracy - the system must be able to display the current bus location with a positional accuracy within 1 minute compared to the actual location
2. Visibility and Accessibility - the display will need to be clearly visible and understandable within a distance of 15 feet
3. System Reliability - the system must be able to function continuously for 7 days without error or timing issues, with no human intervention.

2 Design

2.1 Physical Design

The scope of our project will include the area of campus contained between University Avenue (north limit), Lincoln Avenue (east limit), Gregory Street (south limit), and South Third Street (west limit). This area is shown in Figure 2 and will include notable landmarks pertinent to the purpose of our design, including the Electrical and Computer Engineering Building, Ikenberry Commons (IKE), Main Library, Lincoln Avenue Residence (LAR), and many more residential/educational buildings.

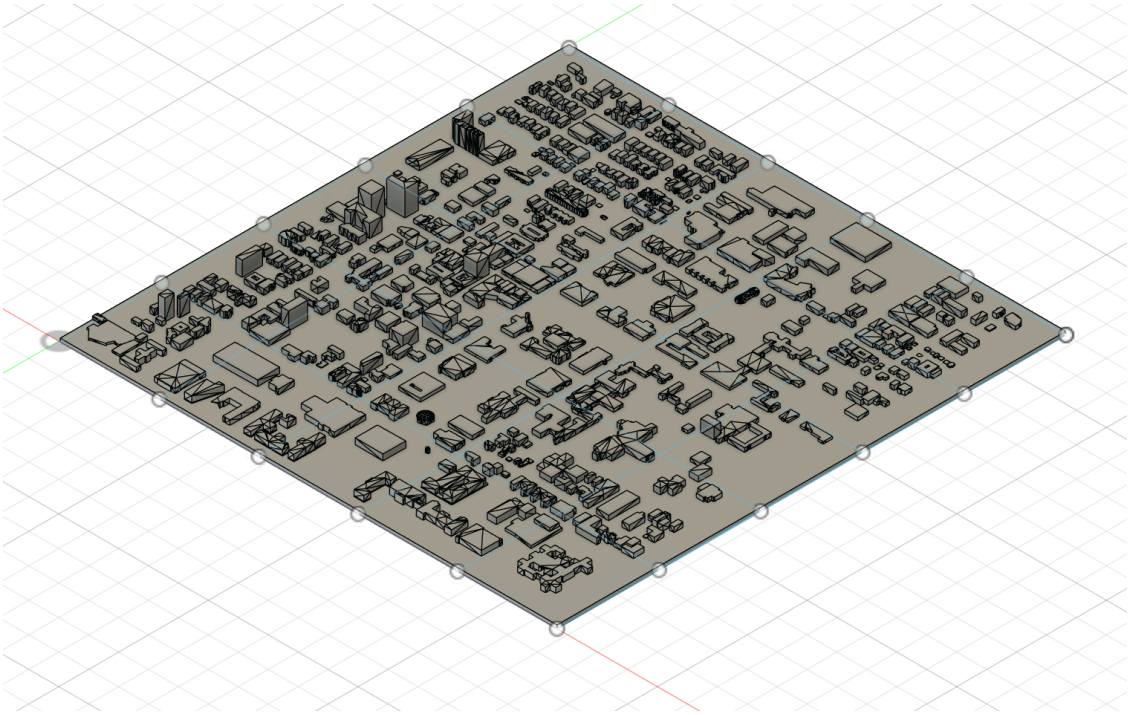

Figure 2- Scope of the 3D Model of Champaign-Urbana used in the Project

We will create the model using a previously available 3D model from RenderHub. This model will then be modified and parsed into 180mm x 166mm divisions and 3D printed. There will be 25 divisions of the map, with a demonstration of the northmost section (Beckman quad) shown in Figure 3, which will be connected via a backing wooden board to create the full model, which will be 850mm x 820mm in size.

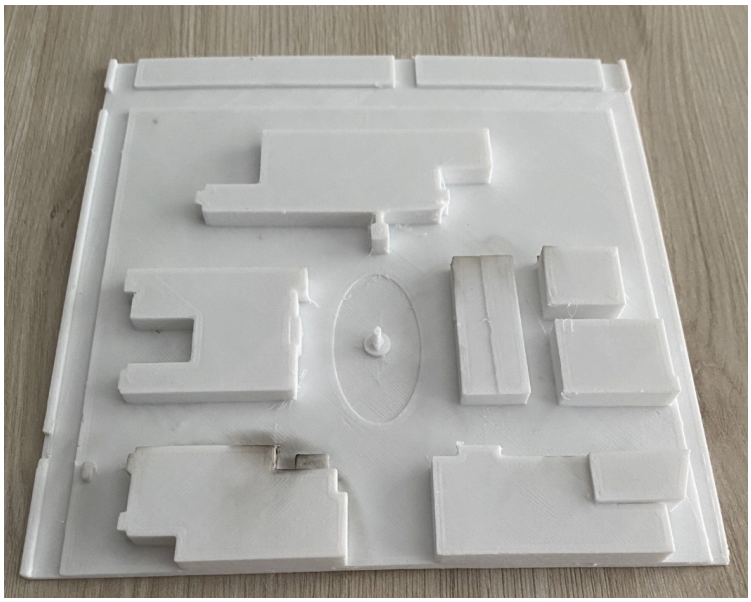

Figure 3 - 3D Printed Model of Beckman Quad

Once the model has been printed and assembled, approximately 6-7 meters of LED strips will be laid within the roads within the 3D model, which were previously designed to be 7mm wide to accommodate two strips of LEDs per road to allow for bidirectional representation of bus movement (ex., show a bus going north while another goes south on same road). The final, completed map will be displayed on an easel-like structure to allow the user to easily view and interact with the display.

2.2 Block Diagram

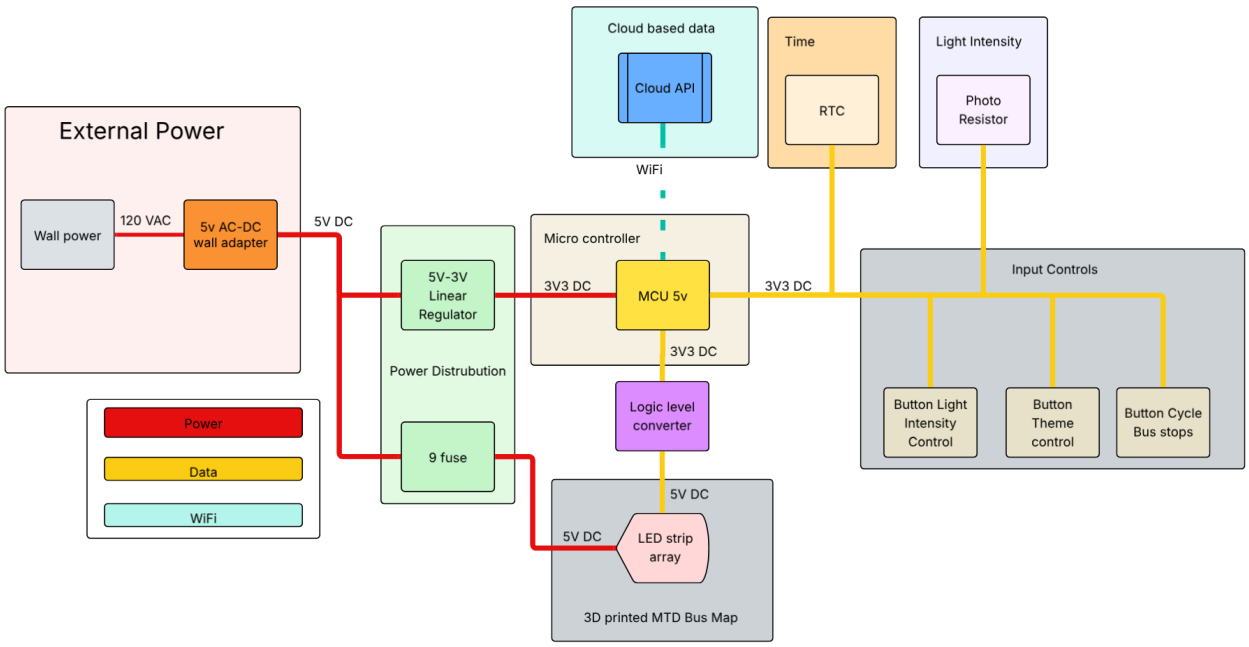

Figure 4 - Detailed Block Diagram of Design

2.3 Subsystem Overviews and Requirements

2.3.1 External Power

The External Power Subsystem is responsible for providing power to both the microcontroller and the LED strip array via the Power Distribution Subsystem. This external power will come from a 120V AC wall plug and will be stepped down by a barrel jack wall adapter (rated for 5V and 10A) to a 5V DC signal. Step-downs and protection circuitry will be implemented by the Power Distribution Subsystem to make the external power usable by the project.

Table 1: External Power Subsystem - Requirements & Verification

Requirements	Verification
• The external power supply must output a voltage between 4.75V and 5.25V DC under a load of up to 2A.	• **Equipment**: Digital Multimeter (DMM), programmable DC load. • **Test**: Connect the power supply to the DMM and set the programmable load to draw 2A. Measure the voltage at the output terminals. • **Demonstration**: Table with current vs. measured voltage values.
• The power supply must be capable of supplying at least 2.5A of current continuously without thermal shutdown or voltage drop below 4.75V.	• **Equipment**: DC electronic load, DMM, IR thermometer, or thermal camera. • **Test**: Connect a 2.5A load and operate for 10 minutes. Monitor output voltage and temperature. • **Demonstration**: Voltage and temperature log vs. time graph.
• The power distribution module must attenuate high-frequency noise such that the output ripple is less than 50mV peak-to-peak.	• **Equipment**: Oscilloscope with 10x probe, capacitor probe tip. • **Test**: Measure voltage ripple at the output of the power distribution module under normal load. • **Demonstration**: Oscilloscope screenshot showing ripple waveform.
• The system must contain fuses or equivalent protection such that a short in either the microcontroller or LED circuit will not damage the other.	• **Equipment**: Replaceable fuses, jumper wires to simulate a short DMM. • **Test**: Intentionally short the microcontroller power line and verify the LED side continues to function (and vice versa). Fuses should trip only in the affected path. • **Demonstration**: Table showing system behavior before/after fault injection.

2.3.2 Power Distribution

The Power Distribution Subsystem is responsible for transferring power from the External Power Subsystem and making it usable for the ESP32-s3 and the LED strip. This system uses a 5V to 3V3 DC-DC Linear regulator chip to step down the external power and supply a steady 3V3 power to the microcontroller. The rest of the Power Distribution goes into resettable fuses that will trip if the input current goes above 10A, which powers the LED strip that is housed in the 3D printed map. The fuses are implemented to provide redundant protection for the LED Subsystem to limit the risk of blowing or damaging any LEDs. For more details about the design of the power distribution, refer to section 2.4.1.

Table 2: Power Distribution - Requirements & Verification

Requirements	Verification
• The DC-DC linear regulator must step down 5V to 3.3V with a maximum output ripple of 100mV peak-to-peak and a load current capacity of at least 1.5A for the ESP32-s3.	• **Equipment**: Digital Oscilloscope, Electronic Load, DMM. • **Test**: Connect the output of the DC-DC regulator to the oscilloscope to measure ripple. Simulate load by drawing up to 1.5A from the output using an electronic load. Measure voltage stability and ripple. • **Demonstration**: Oscilloscope trace showing ripple and voltage level under load.
• The DC-DC linear regulator should maintain a 3.3V output ± 5% (i.e., between 3.135V and 3.465V) under full load conditions.	• **Equipment**: DMM, Electronic Load. • **Test**: Apply a full load (1.5A) to the output of the regulator and measure the voltage with the DMM. • **Demonstration**: Table or graph showing the measured voltage at different loads (0A, 0.5A, 1.5A).
• The system must provide steady power to the LED strip such that the voltage does not drop below 4.75V even under full load.	• **Equipment**: DMM, Electronic Load, Power Supply. • **Test**: Apply a full load to the LED strip (equivalent to 9A) and measure the voltage at the power input terminals. • **Demonstration**: Graph showing voltage drop across the system under different load conditions (0A to 9A).
• The total power distribution system	• **Equipment**: Power Meter, DMM.

(from 5V to 3.3V) should have a total efficiency of at least 85% at full load.	• **Test**: Measure the input power to the power distribution system (at 5V) and the output power (at 3.3V). Calculate efficiency by dividing the output power by the input power and comparing it with the required efficiency. • **Demonstration**: Efficiency calculation report with input and output power values at different load levels.

2.3.3 Cloud API

The Cloud API Subsystem describes the API provided by the MTD Service that we will be utilizing to acquire the necessary data for our project solution. We will request different data for each bus every 30 seconds to obtain the longitude and latitude of each bus. All of the requested data will be sent to the Microcontroller Subsystem, where it will be parsed and processed to be mapped to the LED Map Subsystem. For more information about the process of connecting to the API program, refer to section 2.5.

Table 3: Cloud API - Requirements & Verification

Requirements	Verification
• The programmed system must fetch bus data every 60 seconds (±5 seconds tolerance).	• **Equipment:** ESP32-S3, stopwatch or timestamp logger, Serial Monitor. • **Test:** Record timestamps of 5 consecutive API fetches from the Serial output. Calculate time intervals between fetches. • **Demonstration:** Table of timestamps showing Δt between 55-65 seconds for each interval.
• The program must correctly extract vehicle_id, route_id, latitude, and longitude from the JSON response.	• **Equipment**: Serial Monitor, raw API response from browser/Postman. • **Test**: Compare Serial output values for 3 vehicles against raw JSON data from the same timestamp. • **Demonstration**: Table comparing extracted values vs. actual JSON values (must match 100%).
• The program must be able to differentiate between and track all	• **Equipment:** Serial Monitor. • **Test:** Trigger API fetch when multiple routes

	routes within the scope of the project.	are active. Count total vehicles in response vs. tracked buses in the Serial output. • **Demonstration:** Serial log showing "Total vehicles: N" and "Found X target buses" where $X \leq N$..
•	GPS coordinates must map to LED positions with an accuracy of ±2 LEDs within bounds	• **Equipment:** Calculator, Serial Monitor. • **Test:** Record bus latitude from Serial. Manually calculate expected LED position: LED = ((lat - 40.09) / 0.04) × 1120 (number of LEDs). Compared to the reported position. • **Demonstration:** Table with 5 test cases showing lat, calculated LED, reported LED, and error (must be $\leq$2).
•	The system must detect a WiFi disconnection and retry the connection automatically without crashing.	• **Equipment:** WiFi router/hotspot, Serial Monitor. • **Test:** Disable WiFi while the system is running. Monitor Serial for "WiFi not connected!" Re-enable WiFi and verify reconnection within 20 attempts (10s). • **Demonstration:** Serial log showing disconnection detection, retry attempts, and successful reconnection.
•	The system must track up to 2 buses simultaneously without position conflicts.	• **Equipment:** Serial Monitor, Display • **Test:** Trigger 5 manual API fetches via Button 1 while 2 bus routes are active. Record bus IDs and positions on the monitor and confirm each bus maintains a unique position/ID • **Demonstration:** Table of 5 bus IDs and positions obtained from API pull

2.3.4 Microcontroller

The microcontroller we will be using is the ESP32-S3-WROOM-1. This chip has many capabilities, including an onboard RTC, BLE, and WiFi modules. The microcontroller will be the brain of our project, having an FSM that will constantly probe the MTD Cloud API Subsystem for bus data, then will parse, process, and analyze the data to determine information such as bus locations. This information will then be mapped to the LED map, and an LED control signal (GPIO output of the microcontroller) will be sent to the LED Map Subsystem, where bus locations and ID will be shown on the display. The Microcontroller

Subsystem will also take in 4 inputs from the Input Subsystem that will be used to control bus ID colors, LED intensity, bus lines, and themes that will be communicated to the LED Map Subsystem via the Microcontroller Subsystem.

Table 4: Microcontroller - Requirements & Verification

Requirements	Verification
• The ESP32-S3 program must successfully connect to a WiFi network within 5 seconds after boot, with a connection success rate of 95% or higher over 100 trials.	• **Equipment**: Serial monitor/logging, router, test script. • **Test**: Reset the ESP32 and log connection time over 100 trials. Count the number of successful connections and the average connection time. • **Demonstration**: Table and graph showing success rate and timing distribution.
• The ESP32 program must control the LED map to reflect bus location data with a latency of less than 500ms from API response to LED update.	• **Equipment**: Serial logging, stopwatch/video. • **Test**: Trigger known API response and measure time to the corresponding LED output change. • **Demonstration**: Timestamped log showing API response and LED update event.

2.3.5 Inputs

The Input Subsystem will control the 4 previously discussed inputs that will be communicated to the LED Map via the Microcontroller Subsystem. These inputs consist of three buttons for user input and a photo resistor for environmental input. The user will be able to change the color theme of the LED strip corresponding to the time of year (holidays), change the LED lights' base intensity, and cycle through the different bus routes. The tunable photoresistor circuit will then affect the base intensity of the LED by measuring the ambient light and dim or brighten the circuit as a whole based on the light levels around it, dimming in the dark and getting brighter during the day.

Table 5: Inputs- Requirements & Verification

Requirements	Verification
• The photoresistor circuit must detect ambient light changes and adjust LED brightness in 4 discrete levels within a range of 0–100% PWM duty cycle.	• **Equipment**: Variable light source, DMM, oscilloscope (for PWM measurement). • **Test**: Expose the photoresistor to varying light levels. Observe analog reading and verify corresponding PWM output to LED control pins. • **Demonstration**: Table of light level to PWM level mappings, graph showing correlation.
• Each button must register a press within 50ms, including software debounce, and trigger the associated function (theme, brightness, or route cycling).	• **Equipment**: Oscilloscope or logic analyzer, GPIO debug output, test firmware. • **Procedure**: Press each button and measure the time between physical press (voltage change on GPIO) and software acknowledgment (GPIO toggle or serial print). Repeat 10 times per button. • **Demonstration**: Table showing latency per button press, averaged and max.
• Button presses must not trigger multiple unintended events (i.e., no false multiple triggers) with debounce time <50ms.	• **Equipment**: Oscilloscope, test firmware with counter per button press. • **Procedure**: Hold each button for 1s. Log number of detected press events. Should only count one per press. • **Demonstration**: Table showing number of registered presses per test trial.
• The route button must allow the user to cycle through the bus routes, and this must update the LED map accordingly.	• **Equipment**: Serial monitor, test bus data, LED map. • **Procedure**: Press the route button and verify that the route selection variable changes and the corresponding LEDs update correctly. • **Demonstration**: Table showing route selected vs. visible LED map pattern.

2.3.6 LED Map

The LED map will be made up of a few meters of LED strip going both ways, indicating two lanes of traffic. There will be 160 LEDs per meter of strip, but the overall streets will be smaller in length, making sure to save as much power and cost of material as possible by only wiring the roads that have bus routes. The LED map will sit in lots on a custom 3D printed Map of the Champaign campus. 25 tiles of 850x830mm in area. The LED Map will be powered by the External Power Subsystem, which includes a 5V supply and some fuse protection via the Power Distribution Subsystem. The LED Map will also receive a 5V logic control signal from the Microcontroller Subsystem that will indicate which LEDs within the map should be lit up and at what color or intensity.

Table 6: LED Map- Requirements & Verification

Requirements	Verification
• The LED map should be capable of being controlled to represent real-time bus location, with the bus position updated every 10 seconds or less (as per the FSM).	• **Equipment**: Serial monitor, LED test mode, bus route simulator. • **Test**: Simulate bus route data and verify that the LEDs update in real time based on bus location every 10 seconds. Ensure the LED representation matches the bus position on the map. • **Demonstration**: Screenshot/log of bus location vs. LED behavior, showing bus position update.
• The brightness adjustment from the photoresistor must update LED intensity with a latency of less than 1 second after an ambient light change.	• **Equipment**: Light source, oscilloscope, and stopwatch. • **Test**: Change the light level suddenly and measure the time until the PWM signal or LED brightness changes. • **Demonstration**: Table showing latency per light level change.

2.4 Hardware Design

2.4.1 Operating Voltage & Regulation

For our project to operate, we will need to provide regulated voltage to the microcontroller (ESP32-s3-WROOM-1). According to the ESP32-s3-WROOM-1 datasheet seen in Figure 5**,** the microcontroller will need to operate on a voltage within the range of 3-3.6V.

Symbol	Parameter	Min	Typ	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	—	—	A

Figure 5: ESP32 Operating Conditions [6]

To accomplish this, we will use a 120V AC wall supply connected to a DC power jack with a voltage/current rating of 36V and 10A. The power jack will output 5V at a max of 10 amps as the power supply to the entire circuit. Then, the 5V supply will be stepped down using an AMS1117-3.3 low-dropout voltage regulator with a 1A current output. According to the AMS1117-3.3 datasheet shown in Figure 6, with an input voltage of approximately 4.8V, the voltage regulator will output 3.3V with a V_{ripple} of approximately ±0.099V.

AMS1117-3.3	V_{IN} = 4.8V	3.251 **3.201**	3.300 **3.300**	3.349 **3.399**	V V

Figure 6: AMS1117-3.3 Operating Conditions [5]

The AMS1117-3.3 will provide a power input within the given operating conditions of the microcontroller, with a sufficiently small voltage ripple, thereby protecting the ESP32.

After powering the microcontroller, we will need to provide both power and a logic signal to control the LED strips. The LED strips require a 5V logic signal as well as 5V power. Each meter of the strips (160 LEDs) requires roughly 5-6W. So, with a generous estimation of a maximum current of 10 Amps, we can provide enough power to supply 10 meters of LED strip at maximum power, which should never happen. The power will come directly from the previously discussed barrel jack connected to the 120V AC wall supply. From this supply, we will have a fuse that goes into the linear regulator for the ESP32, then through

another fuse, which leads to the LED strip power. The fuses are to protect the rest of the circuit (ESP32) if the LED strip shorts, and to the power supply if the ESP32 also shorts. To control the LEDs, we need a 5V digital signal. The ESP32 cannot handle 5V signals, so we will need to step up the 3V3 logic output from the microcontroller to 5V to be usable by the LED strip. We will use a 3V3 logic level converter, BSS138.

2.5 Software Design

Below are two flow diagrams of our program's functionality. Figure 7 describes the connection and polling of the MTD API, while Figure 8 shows the main loop for input and output control for our 3 buttons and ambient light detection. Table 7 describes the API functions that will be implemented within our program to access the relevant data.

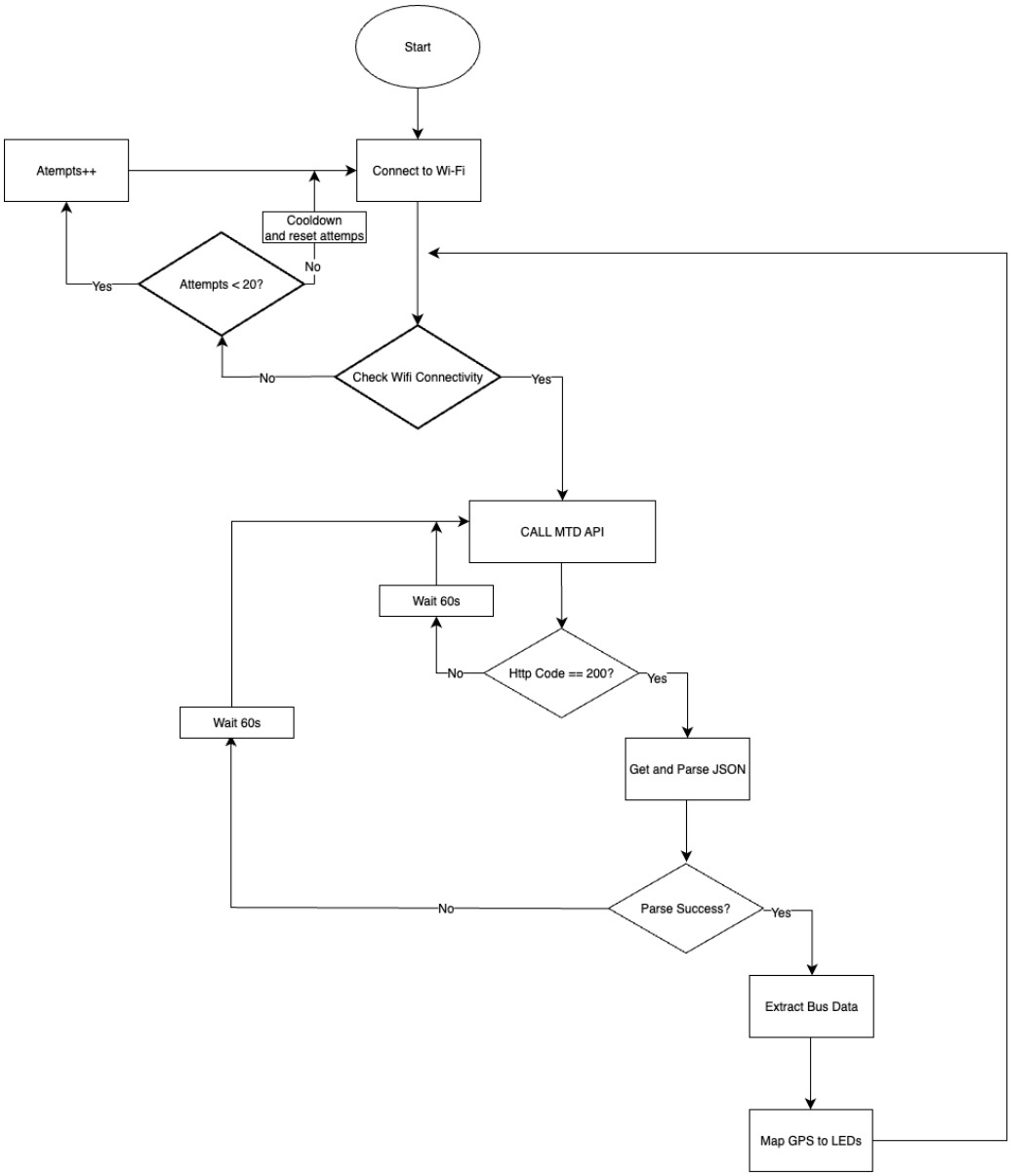

Figure 7: Flow Chart of Program connecting to Wifi, and Extracting the MTD API

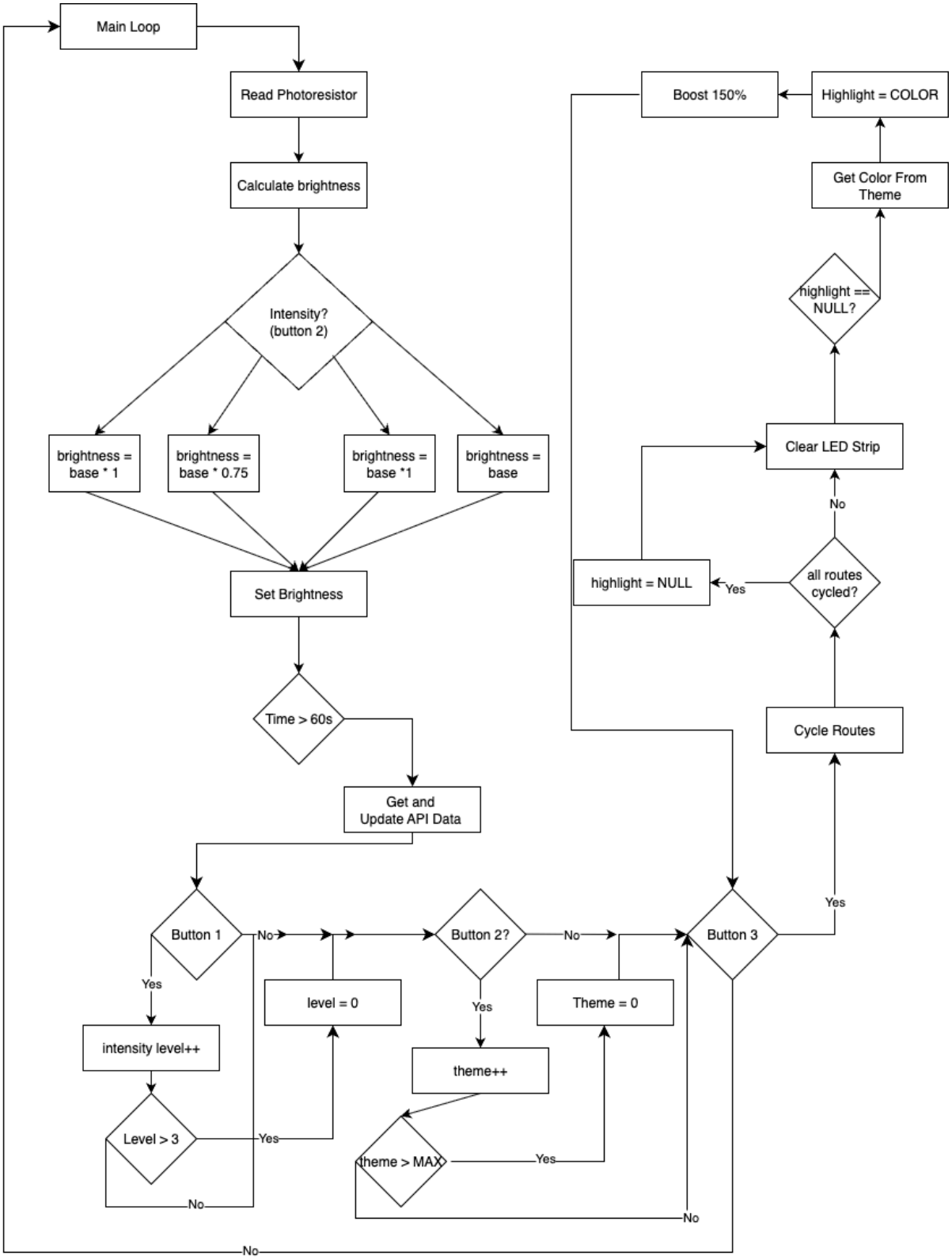

Figure 8: Flow Chart for Button Inputs

Table 7: API Documentation Table

API Endpoint	Method	Description	Response	Usage in Project
GetVehicles	GET	Returns the real-time GPS location of all active buses	JSON with vehicle_id, route_id, lat/lon, timestamps	**Main data source** - Fetched every 60s to update bus positions on the LED strip
GetRoutes	GET	Returns a list of all bus routes with colors and names	JSON with route_id, route_color, route_name	Used to get official route colors for LED themes
GetShape	GET	Returns GPS path points that define a route's shape	JSON with lat/lon points along the route	Could be used for more accurate LED position mapping

Parameters:

- key: API authentication key (required for all endpoints)
- shape_id: Route shape identifier (required for GetShape only)

2.6 Tolerance Analysis

2.6.1 Power Distribution and Logic-Level Conversion

One of the most critical functions for the project's performance will be the interface between the 3.3V supply to the microcontroller and the 5V logic signal to the LED strips. As previously discussed in section 2.4.1, the 3.3V power supply to the microcontroller will come from a 5V to 3.3V converter (5V from barrel jack), then the control signal (microcontroller output) will be stepped up to 5V to support the LEDs. The key performance requirement of this circuitry is to maintain the 3.3V node voltage within the safe operating input range of the microcontroller and ensure proper communication with the LED system, even under worst-case variations in component values, supply tolerances,

and MOSFET parameters. Exceeding these limits could damage the microcontroller or result in losing control of the LED loads.

The critical components within these two blocks that will affect the tolerance analysis will be the linear regulator (AMS1117-3.3) that facilitates the 5V to 3.3V power supply step down, and the N-Channel MOSFET (BSS138) used to implement the 3.3V to 5V logic level converter.

As discussed in section 2.4.1 and Figure 6, the AMS1117-3.3 will operate with an output voltage of 3.3 ± 0.99V. So, the minimum microcontroller power voltage, V_{dd}, would be 3.201V, while the maximum will be 3.399V when taking tolerance analysis into account. Since the voltage being supplied to the microcontroller will be in the range of [3.201, 3.399], the power supply will be within the operating conditions of the ESP32 shown in Figure 5, which require the voltage to be between 3 and 3.6V. This means that the microcontroller will still operate within the tolerance limits of the voltage step-down supply. When this voltage is used to supply power to the microcontroller, the high-level and low-level output voltage of the ESP32 then becomes contingent on this value of $V_{dd.}$

According to the ESP32 datasheet shown in Figure 5, the high-level output voltage (V_{OH}) will range from 0.8 x V_{dd} to V_{dd}, and the maximum low-level output voltage (V_{OL}) will range from 0V to 0.1 x V_{DD}.

V_{OH} [2]	High-level output voltage	0.8 × VDD [1]	–	–	V
V_{OL} [2]	Low-level output voltage	–	–	0.1 × VDD [1]	V

Figure 9: DC Characteristics of the ESP32 [6]

Given this, we know that the LED control signal ($V_{control}$) of the ESP32 (coming from the GPIO pins) will be in the range of $V_{control,low} \in [0, 0.3399]$ and $V_{control,high} \in [2.56, 3.399]$. This LED control signal is then provided to the logic-level converter shown in Figure 10.

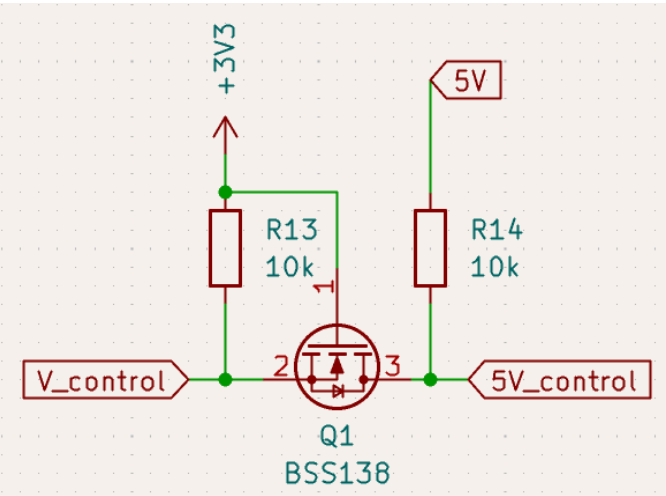

Figure 10: 3.3V to 5V Logic-Level Converter

In this configuration, when $V_{control}$ is high, the value of V_{GS} = 3.3V - $V_{control,\ high}$ has an expected value of 0V, turning the MOSFET (BSS138) off and pulling the LED control signal to 5V. When $V_{control}$ is low, the value of V_{GS} = 3.3V - $V_{control,\ low}$ has an expected value of 3.3V. Since the BSS138 MOSFET has a gate threshold voltage ($V_{GS(th)}$) equal to 1.3V (according to the datasheet seen in Figure 11), we expect that $V_{GS} > V_{TH}$; therefore, the MOSFET will turn on and the LED control signal will be pulled to 0V.

$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$, I_D = 1 mA	0.8	1.3	1.5	V

Figure 11: BSS138 MOSFET Gate Threshold Voltage [7]

However, to ensure the expected operation, we need to take into consideration the previously discussed tolerances of both the 3.3V supply and the ESP32 output signal $V_{control}$. In all, we must ensure that even with variations in the 3.3V regulatory supply and with variations in the microcontroller output, the value of V_{GS} will stay above the maximum possible threshold V_{TH} when the MOSFET is supposed to be on and below the minimum possible V_{TH} when the MOSFET is supposed to be off (values of V_{TH} found from the BSS138 datasheet shown in Figure 11).

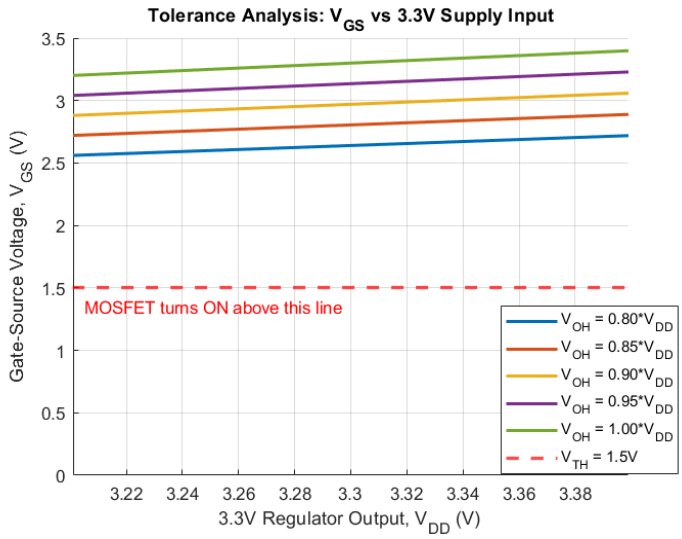

Figure 12: $V_{GS,\,high}$ Under Variations from 3.3V Supply (V_{DD}) and Variations in Microcontroller Output

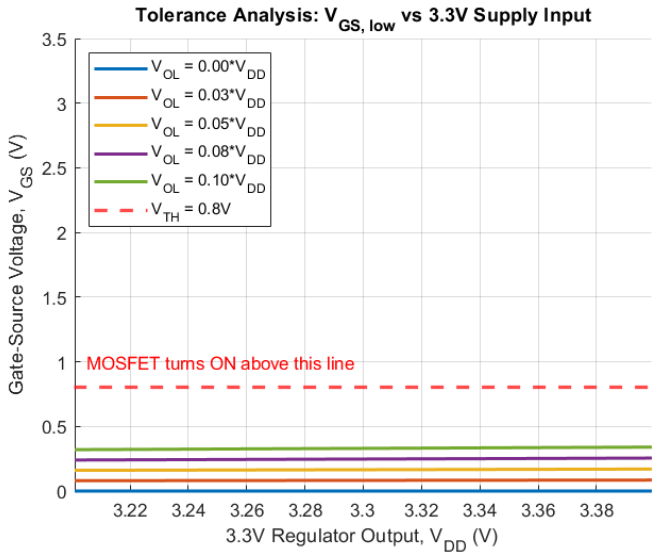

Figure 13: $V_{GS,\,low}$ Under Variations from 3.3V Supply (V_{DD}) and Variations in Microcontroller Output

In Figures 12 and 13, we can see that the values of V_{GS} stay within the desired ranges of operation throughout any of the tolerated values provided from the datasheets of the linear regulator (AMS1117-3.3), the microcontroller (ESP32), and the MOSFET (BSS138). This demonstrates that our circuitry will have proper control of the LED system when working within the operating ranges of the components we've chosen to implement.

3 Cost & Schedule

3.1 Bill of Materials

LABOR					
Team Member	$/hr	Hours Worked/ week	$/week	Weeks Worked	Total Cost
Daniel Vlassov	$42	6	$252	14	$3528
Amber Wilt	$42	6	$252	14	$3528
Ziad AlDohaim	$42	6	$252	14	$3528
Machine Shop	$50	2	$100	1	$100
Total Labor Costs:					**$10,684**
PARTS					
Description	Manufacturer	Quantity	Link	Unit Cost	Total Cost
22uF Capacitor	Samsung Electro-Mechanics	4	CL21A226MAYNNNE	$0.08	$0.16
0.1 uF Capacitor	YAGEO	4	CC0805KRX7R9BB104	$0.08	$0.64
1uF Capacitor	Samsung Electro-Mechanics	2	CL21B105KBFNNNE	$0.08	$0.16
10uF Capacitor	KYOCERA AVX	2	TPCR106K016R1800	$2.93	$5.86
22uF Polarized Capacitor	KYOCERA AVX	2	TCJB226M025R0150E	$1.01	$2.02
Barrel Jack	Same Sky	2	PJ-080BH	$2.15	$5.30
USB Connecter	GCT	2	USB4216-03-A	$0.72	$1.44
Resettable Fuse	Bel Fuse Inc.	4	0ZCF0500FF2A	$1.60	$6.4
Potentiometer	Vishay Sfernice	2	T73YP103KT20	$1.62	$3.24

3 Pin Connector	TE Connectivity AMP Connectors	2	282837-3	$1.16	$2.32
100nF Capacitor	Yageo	2	CC0805KRX7R9BB104	$0.08	$0.16
BSS138	Good-Ark Semi	4	BSS138	$0.1	$0.2
10k Resistor	Susumu	20	RR1220P-103-D	$0.31	$0.62
100 ohm Resistor	Susumu	2	RG2012P-101-B-T5	$0.1	$0.2
5.1k Resistor	Yageo	4	RC0805JR-075K1L	$0.1	$0.4
1k Resistor	Susumu	4	RR1220P-102-D	$0.1	$0.4
Photoresistor	Advance Photonix	2	PDV-P8103	$0.7	$1.4
Push Button	TE Connectivity ALCOSWITCH Switches	10	1825910-6	$0.118	$1.18
AMS1117-3.3	EVVO	2	AMS1117-3.3	$0.27	$0.54
LM393N/NOPB	Texas Instruments	2	LM393N/NOPB	$0.82	$1.64
ESP32-s3-WROOM-1	Espressif Systems	2	ESP32-S3-WROOM-1-N8	$5.49	$10.98
LED Strips	Xnbada	1	WS2812B	$42.99	$42.99
PLA	Elegoo	1	PLA	$28.98	$29.98
3D Model	ClickShop 3D	1	3D Model	$35	$35
Power Supply	Alitove	1	Power Supply	$22	$22
Display Stand	Amazon Basics	1	Easel	$12.59	$12.59
Total Parts Costs					**$231.73**

3.2 Schedule

Week	Task	Person
Sep 29 - Oct 5	Order Parts for Breadboard Demo	Everyone
	Start PSB Design	Amber & Daniel
	Get API calls and test	Ziad
	Start CAD Modeling	Daniel
Oct 06 - Oct 12	Finish PCB Design	Amber & Daniel
	Finish Breadboard Demo Set Up Components on Breadboard	Everyone
	Code and flash ESP32	Ziad
	Work on CAD Modeling	Daniel
Oct 13 - Oct 19	Finish ECE445 Design Document	Everyone
	Revision for Machine Shop (For Stand)	Everyone
Oct 20 - Oct 26	Finish Breadboard Demo 2	Everyone
	Solder and Test PCB (when it arrives)	Amber & Daniel
Oct 27- Nov 2	Modify PCB for 3rd round	Amber & Daniel
	Finish printing 3D Model	Daniel
	Get Machine Shop Stand	Ziad
Nov 3 - Nov 9	PCB Round 3	Amber & Daniel
	Finalize Code	Ziad
	Assemble Display & LEDs	Everyone
Nov 10 - Nov 23	Mock Demos	Everyone
Dec 1 - Dec 11	Final Demos	Everyone

4 Ethics & Safety Considerations

4.1 Safety Concerns

Electrical Safety

The LED matrix and ESP32 microcontroller system require a power supply capable of handling significant current. Risks include overheating, short circuits, or electric shock. Compliance with Underwriters Laboratories and IEC electrical safety standards is required. Proper fuses, insulated wiring, and thermal management must be integrated properly.

Fire Hazard

High current draw in LED matrices can generate heat. Overloaded circuits could present a fire hazard. Following NFPA (National Fire Protection Association) guidelines on electronic equipment and ensuring circuit breakers and buck converters are properly rated will mitigate this.

4.2 Ethical Concerns

Accuracy and Reliability of Information

Displaying real-time bus data should be accurate. If the system displays incorrect bus positions or delays, students may miss classes. According to the IEEE Code of Ethics [1], engineers must hold paramount the safety and welfare of the public and avoid misleading claims. To ensure accuracy, we must test data accuracy against the official MTD API and include error handling for delayed and/or incomplete data, and display so accordingly.

Fairness and Accessibility

The ACM Code of Ethics [2] emphasizes avoiding harm and ensuring equal access to technology. A display located only in certain buildings risks creating an accessibility divide. To address this, the design should support replication in multiple locations and adhere to ADA-compliant visibility and brightness standards, ensuring that visually impaired users are not excluded. In addition, brightness and light patterns must be managed to reduce the risk of triggering photosensitive epilepsy.

Data Privacy and Security

We will be using public API data; however, there are some constraints on its usage and how often the API should be called. Additionally, intentional misuse could occur if the system were modified to track or log users' travel habits, which could aid in stalking people. The ACM Code highlights respecting privacy and ensuring systems are not used for unjust

surveillance. To avoid misuse, we will never collect or store user-specific data and will abide by the rules set by the MTD Bus API.

4.3 Regulatory and Standards Compliance

To comprehensively address the ethical and safety concerns described above, the project must adhere to the relevant safety and regulatory standards. These include:

- IEEE and ACM Codes of Ethics - Require prioritization of public safety, accuracy, and avoidance of harm. [1]
- FCC Part 15 Compliance - Since the ESP32 uses Wi-Fi, it must not interfere with other wireless devices.
- UL and IEC Standards - Apply to electronic component safety and power supply certification.
- ADA Standards - Ensure the display is accessible to individuals with disabilities (contrast, brightness, and placement).
- Campus Facilities and Safety Codes - Enforced through UIUC's DRS and building management.

4.4 Mitigation of Ethical and Safety Risks

To address the ethical and safety risks described above and ensure the project meets the regulatory standards, we will implement a few design decisions to mitigate these concerns. We will implement redundant checks for API accuracy and display error messages on the user interface if the data has been deemed unreliable. This will allow us to ensure the reliability of the information being provided to the user. We will use brightness control and energy-efficient components to reduce the environmental impact and power consumption of our devices. We will follow UL/IEC standards [3] for wiring, insulation, and overcurrent protection to ensure the safety of the display. We will prevent data misuse by never storing or transmitting personal location information and limiting the data we access from the API.

5 References

[1] IEEE Policy 7.8. *IEEE Code of Ethics*. IEEE. https://www.ieee.org/about/corporate/governance/p7-8.

[2] ACM. *ACM Code of Ethics and Professional Conduct*. Association for Computing Machinery, 2018. https://www.acm.org/binaries/content/assets/about/acm-code-of-ethics-and-professional-conduct.pdf.

[3] Simcona. “Understanding UL Wire Standards: Key Specs & Compliance Explained.” *Simcona Blog*, 2 May 2025. https://simcona.com/blog/ul-wire-specifications.

[4] Clark Testing. “IEC 61000-4-11: Electromagnetic Compatibility (EMC) – Part 4-11: Testing and Measurement Techniques – Voltage Dips, Short Interruptions and Voltage Variations Immunity Tests.” *Clark Testing*, 2025 https://clarktesting.com/testing-standard/iec-61000-4-11/?msclkid=7e3ac1736d2c191acf85a9b1491526f4.

[5] Advanced Monolithic Systems, "AMS1117-3.3: 1A Low Dropout Linear Regulator," datasheet, Advanced Monolithic Systems, [Online]. Available: http://www.advanced-monolithic.com/pdf/ds1117.pdf

[6] Espressif Systems, "ESP32-S3-WROOM-1 & ESP32-S3-WROOM-1U Datasheet," v1.6, Espressif Systems, Oct. 29, 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[7] Fairchild Semiconductor Corporation, "BSS138: N-Channel Logic Level Enhancement Mode Field Effect Transistor," Rev. C(W), Oct. 2005. [Online]. Available: https://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf

[8] Worldsemi, "WS2812B: Intelligent Control LED Integrated Light Source," Rev. 1.0, Dec. 2012. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf