Electrical Overview

Year: 2021 Semester: Fall Team: 8 Project: Sink or be Sunk

Creation Date: Sept. 13, 2021 Last Modified: Sept. 13, 2021

Author: Mitchell Arndt Email: [arndt20@purdue.edu](mailto:arndt20@purdue.edu)

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Electrical Overview** |  | x3 |  |  |
| **Electrical Considerations** |  | x3 |  |  |
| **Interface Considerations** |  | x3 |  |  |
| **System Block Diagram** |  | x3 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

*Relevant overall comments about the paper will be included here*

1.0 Electrical Overview

The core component of our project is a 32-bit ESP 32 WiFi and Bluetooth microcontroller. The microcontroller will need both connectivity features. Bluetooth is required for provisioning the WiFi subsystem. The user will connect to the device via a smartphone to provide the WiFi SSID and password for the microcontroller to connect. After this, Bluetooth is no longer used. Instead, the WiFi system takes over to connect to a WebSocket server via the internet. This WebSocket acts as a bidirectional data link where game actions are relayed between two opposing player’s boards. Examples of the data being sent to and from the microcontroller via the WebSocket include the following: an update event when the user positions their ships, an event to one player when the other player attacks their ships, and an event when a player has sunk a ship.

Another critical electrical component is the analog multiplexer. The game board contains 64 ship positions. Each one needs to be periodically monitored for an analog voltage value. It is virtually impossible to find a microcontroller with this number of analog input pins. The multiplexer allows for expanding the IO capabilities of the ESP32 to account for this considerable number of pins. A cascading group of multiplexers will be positioned in a way to bring the 64 inputs down to one ESP32 pin. Eight 8 to 1 multiplexers will be used per row of the game. This brings the input space to 8. One additional multiplexer will be used to scan through the columns. It is important to note that using the multiplexer reduces the analog input pins but it also requires the addition of digital select lines from the microcontroller (in this case 3 per wave, 6 total).

Like the reasoning for the analog multiplexer, this project also requires shift registers to drive the 128 RGB LEDs (128\*3=384) outputs. These shift registers can be daisy-chained together so that the ESP32 requires only 1 data output pin. The driver chip has the additional advantage of dimming each LED with 8 bits of resolution. This leads to 256 brightness levels for each color. This dimming will help to cut down the energy cost of the many LEDs. To save physical space on the PCB, the LEDs chosen for the project include this shift register IC in one surface mount package.

The most computationally intensive task of the microcontroller after the WiFi connectivity is the scanning of user inputs. This includes the analog inputs of the ship positioning as well as the scanning of the user attack button matrix. The ship positioning requires passing through multiplexers, which in turn means that select logic needs to be implemented. This will loop continually as the microcontroller checks each pin and updates the LEDs when the user changes the ship's positions. The button matrix follows the same principle but on a smaller scale. The rows will be continually scanned, and an update event is triggered when a user presses a button.

2.0 Electrical Considerations

The ESP32 microcontroller has a clock speed of 240MHz [2]. This will be handled by the internal clock of the ESP32 RF package being used for the project. There are two ADC peripherals on the ESP32, however, ADC 2 is used by the WiFi system, and any explicit software calls to the ADC, while a WiFi communication is occurring, will be blocking. As this is undesirable for a system that requires frequent analog scanning, ADC 1 will be the exclusive module used for ship positioning detection.

The ADC 1 module will be scanned at 128 Hz based on a hardware timer interrupt. This frequency is desirable for two reasons. The first is that this will allow for detection at each of the 64 ship pins with an equivalent 2Hz frequency. This will give the illusion of a virtually instantaneous response to positioning the ships. The second reason for this frequency is that it will provide minimal loading on the system. The smaller the frequency of hardware interrupts, the better performance the system will have with other subsystems.

The other subsystem that relies on scanning is the user push-button matrix. This 4 x 4 matrix will have each of the rows scanned at 4Hz each. This will require that the hardware timer be configured for an update event at 16Hz. The reasoning for this update frequency is almost identical to the ADC sampling rate. Again, this is based on the minimal system impact while maintaining a seemingly instantaneous response. The reason that each pin gets a slightly higher frequency of 4Hz as opposed to the 2Hz in the ADC is due to the pushbuttons having quicker action of punching in a key as compared to the plugging in motion that is used for putting the ships into the board.

Another goal for this project is to keep it as portable as possible. Because of this, it is paramount that the power of each component is analyzed. As seen from the table below, the main power consumption comes from the ESP Microcontroller. One important note for this table is that it is tallying the absolute max values listed on each device’s datasheet. Because of this, there are two totals listed. The first describes the absolute max power draw expected for the system. The second provides a more typical operating condition. This is done by taking a Root Sum Square (RSS) analysis of the data. This is a statistical method for acquiring an “average” usage of the system. It is important to note that this RSS value is also a conservative estimate and actual power usage may dip even lower than this. The reason for this is that this analysis assumes that every LED is turned on. This will seldom be the case in a normal game.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **description** | **voltage (V)** | **current (mA)** | **quantity** | **power (watt)** |
| ESP32 | 3.3 | 750 | 1 | 2.475 |
| LED RGB | 3.3 | 5 | 128 | 2.112 |
| Rumble Motor | 3 | 165 | 1 | 0.495 |
| LCD Screen | 5 | 250 | 1 | 1.250 |
| Speaker | - | - | 1 | 1.000 |
| Resistive Ships | 3.3 | 0.00033 | 10 | 0.000 |
|  |  |  | **Total** | **7.332** |
|  |  |  | **Total RSS** | **3.660** |
|  |  |  |  |  |
| Battery (Min) | 3.7 | 2600 | 1 | 9.62 |
|  |  |  |  |  |
|  |  | hrs-> | Play Time (Min) | 1.31 |
|  |  |  | Play Time (Typ) | 2.63 |

*Power Budget Analysis*

Following the Totals in the chart above, there is also included an analysis of the estimated playtime of the device. Using our selected battery with a 2600mAh capacity, the system can reasonably operate in a ~1-2.5hr range before requiring a recharge.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| description | voltage (min) | voltage (max) | current mA (min) | current mA (max) |
| ESP32 | -0.3 | 3.6 | 500 | 750 |
| LED RGB | 2 | 5.3 | 20 | 60 |
| Rumble Motor | 2.4 | 3.6 | 110 | 165 |
| LCD Screen | -0.3 | 7 | 120 | 250 |
| Speaker | - | - | 1W | 1.5W |
| Resistive Ships | - | - | 0W | 1/4W |

*Absolute Min/Max Analysis*

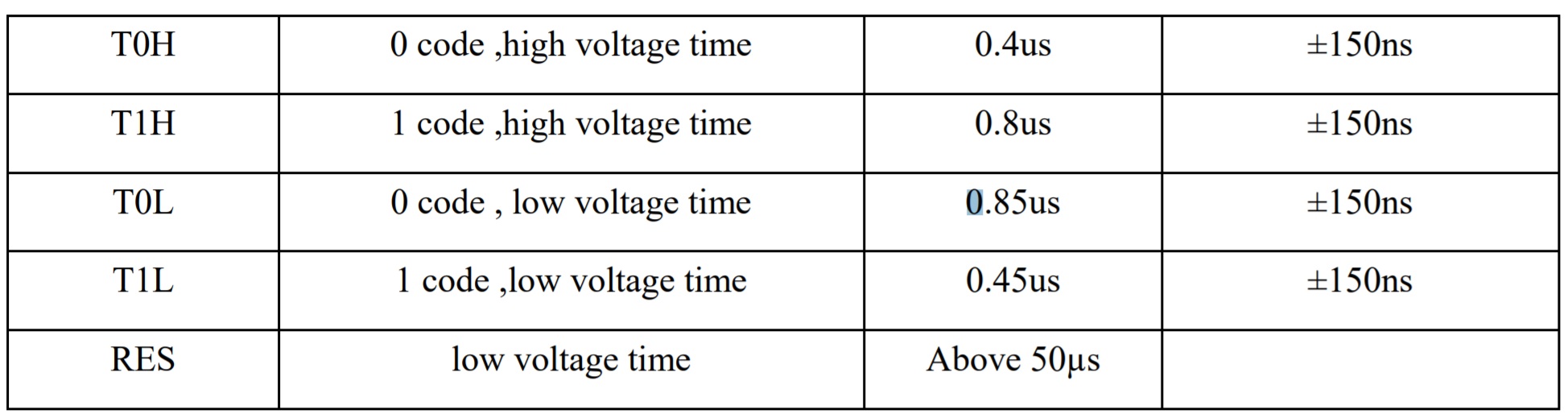
The above table delineates the absolute minimum and maximum ratings for the key subcomponents of the project. This is important to consider while in the design stage to protect components from burning out and being destroyed due to too much power dissipated in the unit. Without careful planning, there is a likelihood that some of the LEDs would burn out or even the microcontroller could be destroyed and ruin the whole project.

One final consideration of the design is the loading effects of analog circuit components. This is important for this project especially due to the matrix of analog multiplexing taking place. The analog signal must pass through two multiplexer devices before it reaches the microcontroller input pins. This means that voltage will be lost along the way to these devices. To limit the effects of this loading, the team specifically chose low impedance muxes. The resistance of these devices is O(100ohms). This means that each ship position peg will effectually have a 100ohm resistor in series between the microcontroller and the voltage divider created from the ships. When using ship resistors on O(100k), this resistance will have very little effect on the scanned voltage at the ADC input.

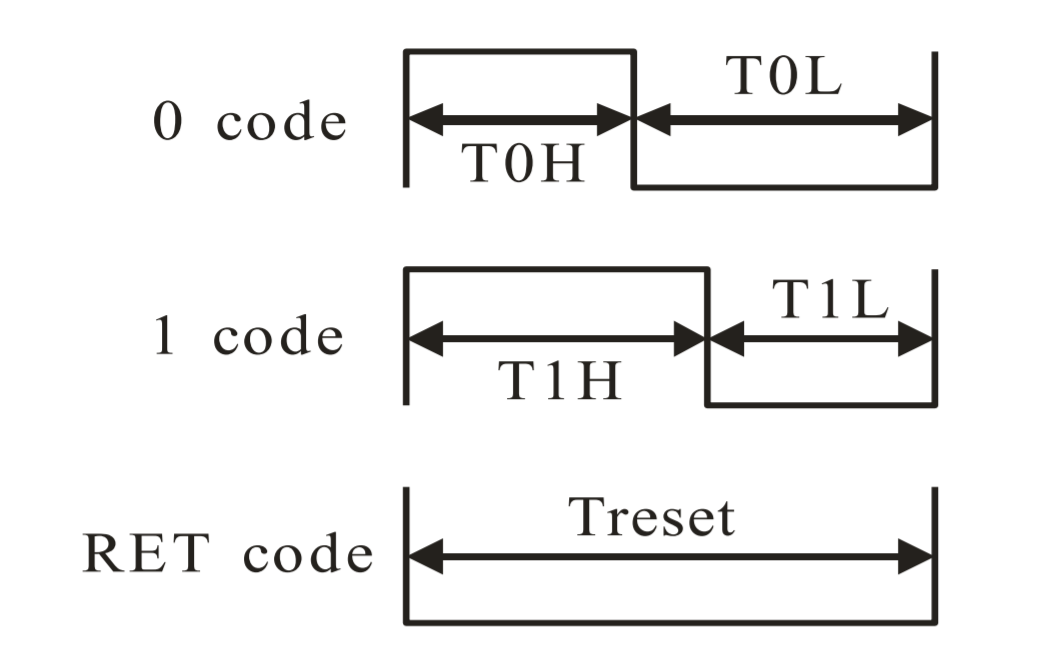
3.0 Interface Considerations

This project uses three main protocols for interfacing with the various subsystems. The first is SPI. SPI will be used to send data to the LCD [6]. Internally the LCD controller IC is essentially shifting in the data from the SPI protocol to be used to light different pixels on the display. The ESP32 microcontroller has an APB clock of 80MHz which allows for a max data rate of the same frequency for the SPI subsystem. This clock can also be divided with a Prescaler to achieve a slower data transfer frequency.

Another communication protocol will be used for the LED driver IC. These devices use a single NZR communication stream requiring only one data wire. The datasheet for the LED provides specific timings for this communication so that the device can decode a 1 or 0 bit. The timings and decoding chart can be seen below. Each device contains 24 bits of registered data corresponding to the red, green, and blue color levels. This means that each color has 8 bits of brightness or 256 levels. The microcontroller sends only sends these 24 bits of data serially to the first LED and the data is passed through the daisy chain to the next LEDs down the line.



LED NZR Timings [1]



LED NZR Decode Diagram [1]

The third interface for data transfer is ADC sampling based on a timer interrupt. The hardware timer has a max ISR at 80MHz, however, due to the sample and hold time for the ADC, it has a max sampling frequency of 6000Hz. The ADC sampling allows for continuous scanning of the ship positioning analog inputs so that the system can detect in real-time when a user places or removes any given ship.

4.0 Sources Cited:

[1] Burgess, P. (n.d.). *Adafruit Neopixel überguide*. Adafruit Learning System. Retrieved September 16, 2021, from <https://learn.adafruit.com/adafruit-neopixel-uberguide/powering-neopixels>.

[2] Industries, A. (n.d.). *ESP32 WIFI-BT-BLE MCU module / ESP-WROOM-32*. adafruit industries blog RSS. Retrieved September 15, 2021, from <https://www.adafruit.com/product/3320>.

[3] *Lion-1865-26*. LION-1865-26 Dantona Industries | Battery Products | DigiKey Marketplace. (n.d.). Retrieved September 15, 2021, from <https://www.digikey.com/en/products/detail/dantona-industries/LION-1865-26/13692661>.

[4] *MCP608-I/p*. DigiKey. (n.d.). Retrieved September 15, 2021, from <https://www.digikey.com/en/products/detail/microchip-technology/MCP608-I-P/319470>.

[5] *SP-4005Y Speaker*. DigiKey. (n.d.). Retrieved September 15, 2021, from <https://www.digikey.com/en/products/detail/soberton-inc/SP-4005Y/9924431>.

[6] *TC1602A-01T*. Adafruit. (n.d.). Retrieved September 14, 2021, from <https://cdn-shop.adafruit.com/datasheets/TC1602A-01T.pdf>.

[7] *Z4TH5B1709181L*. DigiKey. (n.d.). Retrieved September 15, 2021, from <https://www.digikey.com/en/products/detail/jinlong-machinery-electronics-inc/Z4TH5B1709181L/12323581>.

Appendix 1: System Block Diagram

