Electrical Overview

Year: 2022 Semester: Fall Team: 12 Project: R.A.C.H.E.L.

Creation Date: ­September 12, 2022 Last Modified: October 6, 2022

Author: Micah Morefield Email: mmorefie@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:

1. Electrical Overview

The primary computation component is a 64-bit Linux laptop, which will handle game logic, reading inputs from the secondary computation component and from the depth sensor camera. It will use an object detection algorithm to determine where the ball is and will read data through UART from the microcontroller to find where the ball bounced. This data will be used to update the projector display on the table, which is configured as the laptop’s screen, through VGA.

The secondary computation component is a 32-bit, 64-pin microcontroller [1] (STM32F091RCT6), which will handle processing for user input and sensors. That is, it will read digital data from a 4x4 keypad matrix for user input and analog data from piezoelectric sensors. The microcontroller’s ADC in combination with a protection circuit, filter, and amplifier will interpret the analog data. Using an edge detecting algorithm, it will determine if a ball has bounced on the surface the sensor is mounted to. The relevant data will be shared through UART to the laptop for game processing.

2.0 Electrical Considerations

2.1 Operating Voltage

The main operating voltage of the game will be 3.3V, using our microcontroller to handle a majority of the functionality from our power source. While our laptop is handling most of the processing power, we have opted to isolate the power source for the main system (laptop, projector, and camera) from our designed system (microcontroller, sensors, and peripherals). The keypad matrix is simply a switch array, so no particular voltage is needed since the microcontroller will provide both the input to the array and reading the output into GPIO pins. The piezoelectric sensors can output up to 20 V with significant pressure applied. Of course, this is not something the microcontroller can handle, so we have a diode protection circuit and an operational amplifier that prevent any signal outside the range of 0 – 3.3V from passing into the ADC pins. The safe range for this device is 0 – 3.6V [1].

2.2 Operating Frequency

We have our microcontroller operating with a clock frequency of 48 MHz, the standard speed of the device. We handle various interrupts and need to ensure that we aren’t overworking the controller. The keypad matrix’s “high” lines change at a frequency of 1 kHz. Then, any time one of the output lines goes high, we process which button has been pressed. At too high of a frequency, we may read a column, even though the row hasn’t gone “high” yet, and we’ll read data improperly. In a similar fashion, using a higher frequency may cause the debouncing algorithm to break down – too quickly checking the lines may see “low” for long enough to consider the button up, when it is still bouncing. To solve this, we could 1) increase the size of the history data structure, or 2) reduce the changing frequency. The user will not notice any lag in button press processing at a 1 kHz frequency (4 ms delay at most).

Anytime we receive a UART message, we interrupt current actions to read it. Most processing, though, is done by the ADC, which can operate at a frequency up to 1 MHz. This is overkill for reading the piezoelectric outputs, so we currently sample the ADC at a frequency of 20 kHz. This works well for handling 2 ADC channels, however, since there is only 1 ADC controller, if we were to add channels, we may need to increase the conversion frequency. With these time constraints, there is no concern that our processes cannot finish, especially as we do not expect high frequencies of interrupts from the keypad or from UART.

2.3 Power Supply

The power supplied to the PCB will come from a wall adapter, running at 3.3V and 1A. 3.3V is the necessary voltage for the microcontroller, operational amplifier, and USB-to-serial IC that all live on the board. Our design will consume 194mA of the 1000mA supplied, or 2.11 W of 3.3W supplied, at a maximum. The breakdown of this is depicted in Table 1, below. While a 500mA adapter would be sufficient for our current design, for insurance, and for the possibility that devices change or new ones are added, we will be using a 1A adapter.

|  |  |  |
| --- | --- | --- |
| Device | Maximum Current Draw | Maximum Power Draw |
| Microcontroller (STM32F091RCT6) [1] | 120 mA | 480 mW |
| USB-to-serial IC (FT232RL) [2] | 24 mA | 500 mW |
| Operational Amplifier (LM324N) [3] | 50 mA | 1130 mW |
| Total | 194 mA | 2110 mW |
| Maximum Allowed | 1000 mA | 3300 mW |

Table 1. Power Consumed Per Device

3.0 Interface Considerations

3.1 Serial Interface

A serial interface will be used to send user and sensor data from the microcontroller to the laptop. This will be handled with the UART protocol, running with an 8N1 packet style (1 start-bit, 8 bits of data, no parity bit, 1 stop-bit) at 115200 baud. The packet structure sent from the microcontroller to the laptop will resemble that shown in Table 2 below. As for reverse communication, we currently only plan to implement 1) a request data command, to which the microcontroller will respond with the following packet, and 2) a command to reset the system.

|  |  |  |  |
| --- | --- | --- | --- |
| Control Type (1) | User Data / Errors (4) | Sensor Data (2) | Unused (1) |

Table 2. 8-bit UART Packet Structure

The control type bit tells the computer how to handle the next 4 bits – if 1, we recognize that a key has been pressed, and we should handle it. As there are 16 keys, 4 bits are sufficient to determine which key has been pressed. If the control bit is 0, an error has occurred. The next 4 bits highlight specific information about what has happened. As of the time of this writing, there are no major errors we intend to require reporting for – this is designed with the possibility in mind that the laptop may need to handle problems that occur on the microcontroller. Likely, a binary value from this set of 4 bits will represent a specific type of error, that will be recognized in an enumeration on the laptop code.

The sensor data section is simple – if the first bit is high, we have had a bounce on the left side of the table. If the second bit is high, we have had a bounce on the right. If both bits are low, no bounces have occurred yet. If both bits are high, we may have an example of an error. If this occurs, we either 1) had faulty sensing, or 2) did not request data at a high enough frequency.

The UART protocol will be handled by the microcontroller’s peripheral (embedded C code), and by a USB-to-serial adapter on the laptop (C++ code).

3.2 Various Pre-built Interfaces

Our design also implements communication from the laptop to a camera (through USB) and to a projector (through VGA). Neither of these interfaces have been set up by us, nor count as any engineering effort towards our project, but are a part of it, nonetheless.

4.0 Sources Cited:

[1] STMicroelectronics, “STM32F091RC,” 2017. [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f091rc.html#overview>.

[Accessed 16-Sep-2022].

[2] FTDI, “FT232R USB UART IC Datasheet,” 21-May-2021. [Online]. Available: <https://ftdichip.com/wp-content/uploads/2020/08/DS_FT232R.pdf>. [Accessed 17-Sep-2022].

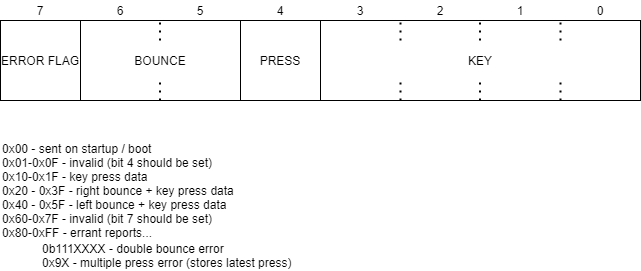
[3] TI, “LM324N Low-Power, Quad-Operational Amplifiers,” Jan-2015. [Online]. Available: <https://www.ti.com/lit/ds/snosc16d/snosc16d.pdf>. [Accessed 17-Sep-2022].

Appendix 1: System Block Diagram

*Graphical user interface

Description automatically generated with low confidence*

Appendix 2: UART Packet Structure

**