Electrical Overview

Year: 2022 Semester: 2 Team: 2 Project: VRms

Creation Date: February 2, 2022 Last Modified: February 2, 2022

Author: Emma Clary Email: clarye@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:

1.0 Electrical Overview

**Component List:**

32-bit microcontroller, Buzzer, SD card, raspberry-pi, two synchronizable cameras, 8 servos (180 degrees), 8-bus servo driver, and a VR headset and controllers.

**General Flow of Data:**

Our VR controllers will be generating servo angle data points in a unity simulation, and that data will be sent to a server that is connected to the raspberry-pi. The raspberry-pi will send that data to the microcontroller through a wired connection, where the data will be put on a buffer, parsed, and sent to the servo data bus. In the other direction, data will be read off of the servo driver and will be sent back through the raspberry-pi to the server and to the unity simulation.

**Raspberry Pi and 32-Bit Microcontroller:**

The Raspberry-pi will be in charge of receiving the angular data and sending it to the microcontroller and vice versa, but it will also be in charge of reading camera data and sending it to the server to be relayed to the VR simulation. The microcontroller’s purpose is to receive the data, parse angular data into linear servo positions (500-2500), and then send that data to the servo drivers. The buzzer will be triggered by the microcontroller whenever the micro receives a message from the Pi indicating the connection to the server has been lost.

2.0 Electrical Considerations

**Operating Voltage:**

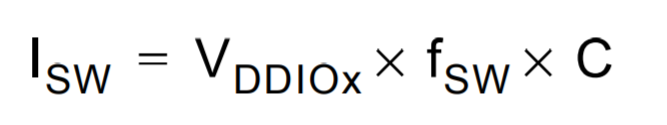
Our microcontroller allows a maximum of 3.6V to be supplied. The raspberry Pi has a maximum input voltage of 5V. The servo driver module that will control all of the servos has an input voltage maximum at 5V. The buzzer has a maximum input voltage of 5V. Since the

**Operating Frequency:**

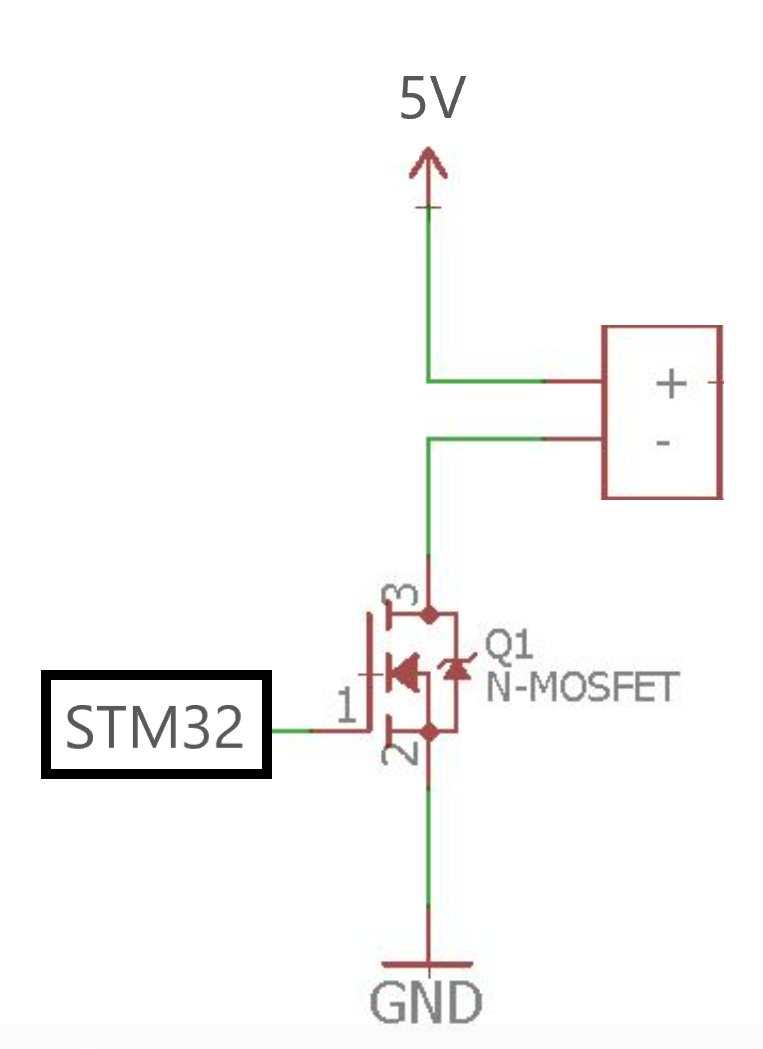
We need the translation from input from the VR controllers to output on the robot arm to be seemingly immediate to the user. We set this threshold to be at maximum a 0.5 second response time. The microcontroller being utilized has an operating frequency of 48 MHz. We will operate at this maximum frequency to get the best possible data transfer rate. The baud rate will be 9600 bps for the microcontroller/servo driver and 115200 bps for the Raspberry-Pi/microcontroller. The baud rate for the servo driver is set by the manufacturer, but we chose a higher baud rate for the Raspberry-Pi so we don’t run into any performance issues. When operating at these frequencies and data rates, we will definitely hit the 0.5s tolerance that we set in a previous document. Therefore the only cause for concern is the transmission rate of the unity to server to Raspberry-pi connection. We found this data rate by timestamping the data in unity, and we found it was received well under the 0.5 second threshold.

**Power Limits:**

In this project, we will be using solely wall power. Using an AC to DC converter we will input 9V to our system and step it down to the voltages required for each component. The microcontroller has a maximum power dissipation of 444mW. The UART3 peripheral is listed as consuming 16.2 micro amps, the TIM1 peripheral consumes 2.6 micro amps, and assuming a couple other timers and peripherals are used, the total on chip peripheral current consumption will still be <1mA. Therefore the only peripheral drawing serious current will be the I/O dynamic current consumption from the buzzer and the servo driver when they are connected. The formula for the dynamic current consumption is



As long as the two peripherals we have do not exceed the 444mW power rating, we are fine. Since the output voltage is ~3.5V and the max power rating is 444mW, the max current we can pull from the peripherals at once is ~126 mA. Since the buzzer runs 100mA of current, we will be utilizing a mosfet to reduce the load on the micro.



The servo driver will also be powered with the wall outlet directly, which means none of our peripherals will be drawing heavy current from the micro itself.

**Tolerances:**

Microcontroller: Vdd = 2.0V to 3.6V w/ power dissipation limit of 444mW.

Raspberry Pi: 5V +/- 5% (min 4.75V, max 5.25V)

Buzzer: Vdd = 3V to 5V w/ rated current of 100mA

Servo Driver: Vdd = 5V to 8.4V with a 3.3v output

3.0 Interface Considerations

The only wired communication protocol we are using in our project is UART. In our project, we will be using UART to communicate between the Raspberry-Pi/microcontroller and microcontroller/servo driver. The baud rate will be 9600 bps for the microcontroller/servo and will be 115200 bps for the Raspberry-Pi/microcontroller.

4.0 Sources Cited:

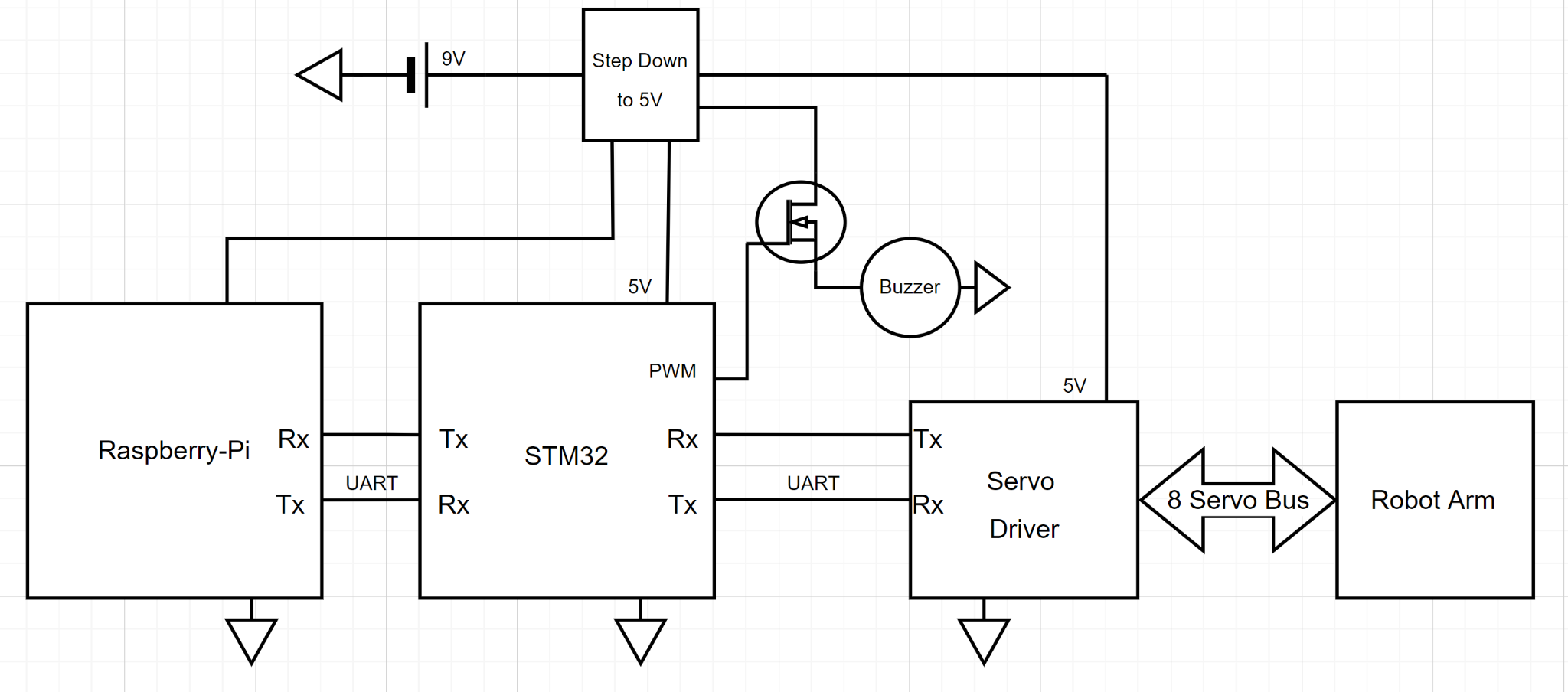
[1] “Most common baud rates table,” *Most common baud rates table | Lulu's blog*. [Online]. Available: https://lucidar.me/en/serialib/most-used-baud-rates-table/. [Accessed: 04-Feb-2022].

[2] STM32F091xB STM32F091xC Datasheet

[3] ““RPI voltage status test,” *English*. [Online]. Available: https://www.eginnovations.com/documentation/Raspberry/RPi-Voltage-Status-Test.htm#:~:text=The%20voltage%20requirement%20for%20all,USB%20devices%20connected%20to%20it. [Accessed: 04-Feb-2022].

[4] Androidfanboy, “Driving piezo buzzer with n-MOSFET?,” *Arduino Forum*, 30-Sep-2017. [Online]. Available: https://forum.arduino.cc/t/driving-piezo-buzzer-with-n-mosfet/483406. [Accessed: 04-Feb-2022].

Appendix 1: System Block Diagram

**