Homework 5: Theory of Operation and Hardware Design Narrative

Team Code Name: \_Augmented Reality Project\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Group No. \_\_5\_\_\_

Team Member Completing This Homework: \_Alec Green\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

E-mail Address of Team Member: \_\_green26\_\_\_\_ @ purdue.edu

NOTE: This is the second in a series of four “design component” homework assignments, each of which is to be completed by one team member. The body of the report should be 3-5 pages, **not** including this cover page, references, attachments or appendices.

Evaluation:

|  |  |  |  |
| --- | --- | --- | --- |
| SEC | DESCRIPTION | MAX | SCORE |
| 1.0 | Introduction | 5 |  |
| 2.0 | Theory of Operation | 20 |  |
| 3.0 | Hardware Design Narrative | 20 |  |
| 4.0 | Summary | 5 |  |
| 5.0 | List of References | 10 |  |
| App A | System Block Diagram | 10 |  |
| App B | Schematic | 30 |  |
|  | TOTAL | 100 |  |

Comments:

*Comments from the grader will be inserted here*

1. Introduction

An augmented reality system with multiple wireless headsets and a central control unit (CCU) is being designed. The headset board comprises a microcontroller and peripherals that serve sensing, wireless communication, and rendering purposes. The CCU, on the other hand, performs processing on a Linux-capable Raspberry Pi, utilizes wireless communication and user interface peripherals. Because the CCU components are primarily a subset of the headset components, the CCU will use the PCB derived from the headset schematic/layout. No exotic circuit techniques have been implemented in the schematic; following the manufacturer’s recommendations with regard to capacitors and resistors is indicated to meet the system’s needs as a whole.

1. Theory of Operation

The augmented reality system comprises multiple battery-powered headsets controlled by a microcontroller and one Raspberry Pi, and a single central control unit controlled by one Raspberry Pi.

Upon insertion of a 3.7V lithium ion battery into the headset, or the contact closure of the battery’s on/off switch, the battery’s voltage is connected to the input of 4 power supplies, the input of the battery “fuel gauge”, and the output of the battery charger. A polarized battery header will be used to prevent the physical insertion of the battery in the wrong polarity direction. The single battery powers all components of the headset except for the battery charger, which is optionally powered by an external 5V source in the form of a USB connection. Three of the four power supplies are low dropout (LDO) regulators which altogether supply 3.3V to the IMU sensors, GPS, Raspberry Pi, microcontroller, and XBee radio. The fourth power supply is integrated in the LCD display package, and accepts the battery’s 3.7V directly. Once the IMU sensors are powered, they are capable of transmitting raw sensor data to the microcontroller for processing via I2C protocol [1]. The battery fuel gauge will similarly communicate the battery’s charge to the microcontroller via I2C [2]. The Raspberry Pi will communicate with the microcontroller via SPI, and both the GPS and XBee modules will both communicate with the microcontroller via the USART protocol [3, 4]. The following table summarizes the communication protocols of microcontroller-connected components:

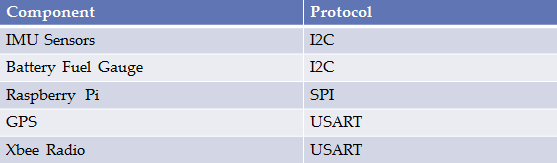


Figure 1.

Digital Communication Protocols Used by Microcontroller-Interfacing Components

The microcontroller will process the IMU sensor and GPS data (e.g. Kalman filtering) and battery fuel gauge measurement via software, and hand off this information in a more palatable form to the Raspberry Pi for rendering on an LCD that communicates exclusively with the Raspberry Pi. The microcontroller will also likely send the processed IMU sensor and GPS data to the CCU via the XBee radio. However, this functionality is software-defined. Throughout operation, the microcontroller will also receive instructions from the CCU via the XBee radio, and interpret the packets based on software loaded via the 6-pin microcontroller debugger, or the USB port. For convenience of mobile code-loading and debugging of the microcontroller, the 6-pin debugger will likely only be used for the required initial code-loading (note the ‘DO NOT STUFF’ directive on the schematic) by physically pressing the debugger connector onto the board for a moment. After the initial code loading, the differential communication lines of the USB port will serve as the debugging path. Note that the USB port serves two uses: regular debugging of the microcontroller, and charging of the headset battery.

The central control unit PCB will be powered by a wall supply. The wall supply directly powers the Raspberry Pi, which in turn powers all other devices on the CCU (XBee radio, user interface LCD/pushbuttons). The schematic and PCB layout are designed in such a way that the central control unit can utilize the headset PCB, even though the CCU uses significantly less components.

Indicator LEDs and ‘spare signal’ vias will be important during the debugging process. Currently, only a battery and battery charger LED are included in the schematic. However, once the PCB has been routed with the current components (which have higher priority and less mobility in terms of where they can be placed), more LEDs and signal pad-outs will be added in accessible/visible locations.

Although most components on both the headset and CCU boards could operate from voltage within the range 2.2-3.6V, the GPS module [4] had an operating voltage range of 2.8-3.6. Given that 3.3V is roughly in the center of these ranges, and that 3.3V is a common supply voltage for small components which could be used if we needed to swap parts later, all schematic components on the headset PCB (with exception of the battery charger and LCD power supply) are powered by 3.3V to avoid the need for level translation. Apart from voltage supply considerations, the operating speed and mode of most devices are configured in software. The microcontroller, for example, has a software-configurable operating speed. The 8MHz crystal that supports the microcontroller was chosen among two manufacturer-recommended oscillator circuit configurations in lieu of a lower-frequency option [5]. The higher-frequency base clock speed was chosen in order to allow for a faster maximum clock speed (in this case, 168MHz), which will increase the time in which sensor data can be processed and handed off to the Raspberry Pi for rendering. The additional power that accompanies a faster clock speed are negligible in this case as the power consumption of the microcontroller as a whole (~30mA) will be dwarfed by the power consumption of the Raspberry Pi and headset LCD alone (together, >1A).

1. Hardware Design Narrative

The microcontroller used on the headset (Appendix B, Figure 4), comprises three primary ports: A, B, and C. Because the oscillator will physically sit near the pad out of port C, and the oscillator creates relatively high noise, it would be desirable to locate all the connections to the microcontroller at ports A and B. However, in order to reduce the distance and complexity of the PCB routing, it eventually became necessary to move some of the component connections to port C. Given that the signals migrated to port C are all digital, the migration is more permissible than if the signals were analog. The following table summarizes which devices are connected to which ports:

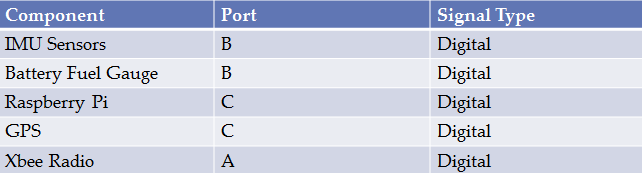


Figure 2. Microcontroller Port Assignments

These ports will be used for digital communication protocols as indicated in Figure 1. Per the manufacturer’s recommendations, a single 1uF bypass capacitor was placed on the microcontroller’s VBatt pin, 2x 2.2 uF capacitors are placed on the microcontroller’s internal regulator, and 10x .1uF bypass capacitors are applied to the microcontroller’s power pins. Because the manufacturer’s recommendations on schematic and layout configuration of the microcontroller will be followed, it is anticipated that a simple ground pour will appropriately stabilize the power/ground rails, and that no special components or routing considerations will need to be dealt with to avoid, for example, ground loops.

All other components were similarly configured in the schematic and layout according to manufacturer’s recommendations. For example, both the battery charger and fuel gauge (Appendix B, Figure 7) use resistor and decoupling capacitor configurations as specified in their respective datasheets for supporting a 3.7V lithium ion rechargeable battery. The bulk 22uF electrolytic capacitors (Appendix B, Figure 8) were suggested by the power supply data sheet as well.

On the CCU board, a single wall supply will indirectly power the entire board through the Raspberry Pi. The headset board, on the other hand, uses four power supplies. One power supply is already integrated into the LCD, and therefore must remain. However, components were split among the other three power supplies in consideration of noise and current consumption. The STEVAL board that contains the analog IMU sensors (Appendix B, Figure 5) receives a dedicated power supply in order to decouple the IMU sensing from noise caused by radio transmission and graphics rendering in the GPS/Xbee radios and Raspberry Pi respectively. The microcontroller, GPS, Xbee, and Raspberry Pi are split up among two 500mA power supplies as indicated in Figure 3 based on their respective current consumption; the Raspberry Pi can consume up to 400mA during peak processing, which is more current than the other three devices combined during peak current draw.

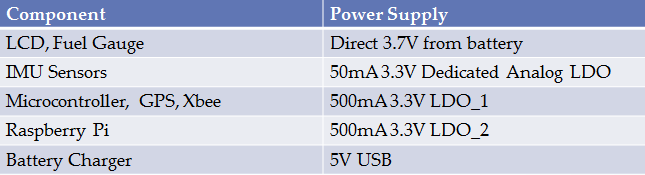


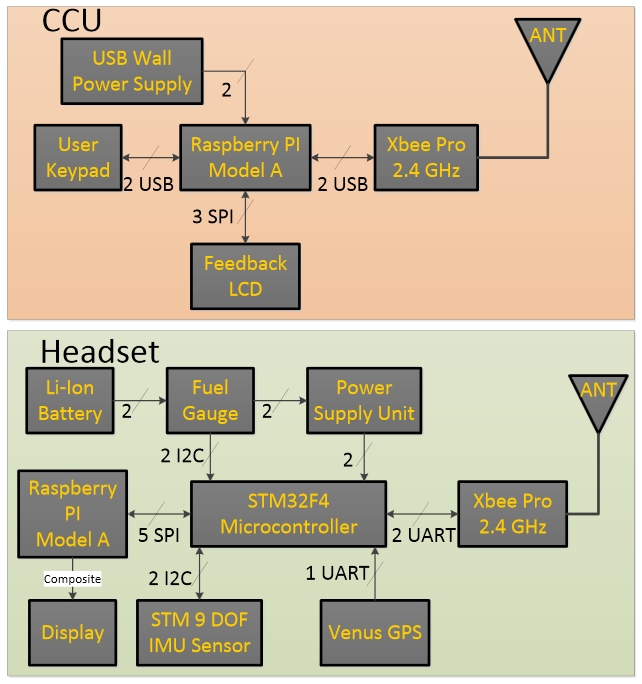
Figure 3. Headset Power Supply Distribution

1. Summary

The necessary circuitry and support components have been organized in the schematic screenshots of Appendix B to satisfy the block diagram of Appendix A. Logical subsections of the PCB include: microcontroller, IMU sensors, GPS/Xbee radios, power supplies, battery charging/monitoring, and interfacing pads (e.g. Raspberry Pi/LCD headers, debugger port). Manufacturer recommendations were exclusively followed, and common sense was applied where manufacturer recommendations were not available: deciding to power the board with 3.3V power supplies, moving digital communication traces to the microcontroller’s port C despite potential noise emission from the proximal oscillator (Figure 2), and dividing up power supplies to accommodate both low noise and high current consumption requirements (Figure 3).5.0 List of References

1. SGS-Thomson Microelectronics, “L3GD20 and LSM303DLHC 9-axis module for a standard DIL socket”, STEVAL-MK108V2 datasheet, Nov. 2011 <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00041389.pdf>
2. Maxim, “Compact, Low-Cost 1S/2S Fuel Gauges with Low-Battery Alert”, MAX17043 datasheet, Sep. 2009 [Revised Aug. 2012] <http://datasheets.maximintegrated.com/en/ds/MAX17043-MAX17044.pdf>
3. SkyTraq Technology, “Venus638FLPx GPS Receiver”, Venus638FLPx datasheet, Feb. 2010 [Revised Jan. 2011] <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/Venus/638/doc/Venus638FLPx_DS_v07.pdf>
4. Digi International, “XBee-Pro 900HP,” Xbee Pro 900HP datasheet, Aug. 2013 <http://www.digi.com/pdf/ds_xbeepro900hp.pdf>
5. SGS-Thomson Microelectronics, “ARM Cortex-M4 32b MCU+FPU”, STM32F405RG datasheet, Sep. 2011 [Revised Jun. 2013] <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf>

Appendix A: System Block Diagram

**

Port B

Port A

Port C

Port B

Port C

**Figure 4. Block Diagram**

Appendix B: Schematic

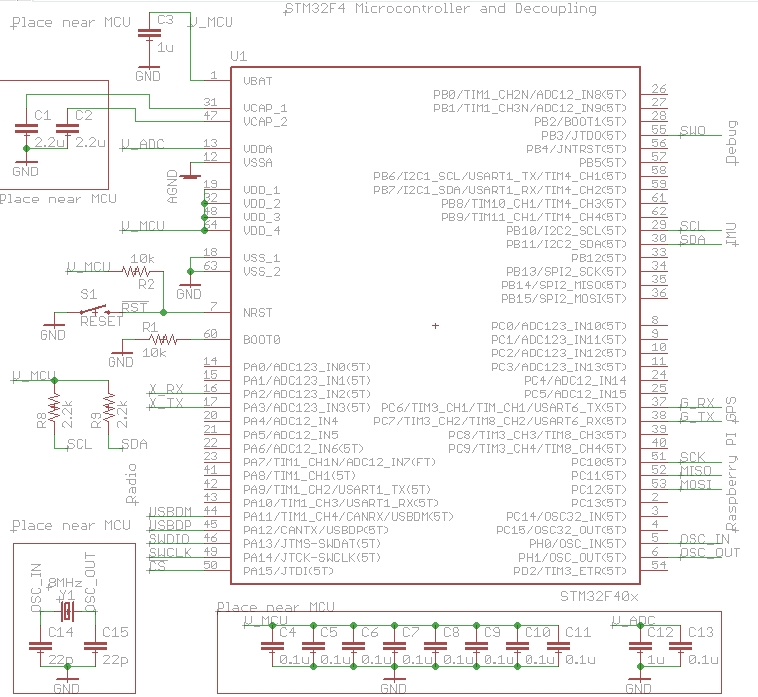


Figure 5. Microcontroller (STM32 F4)

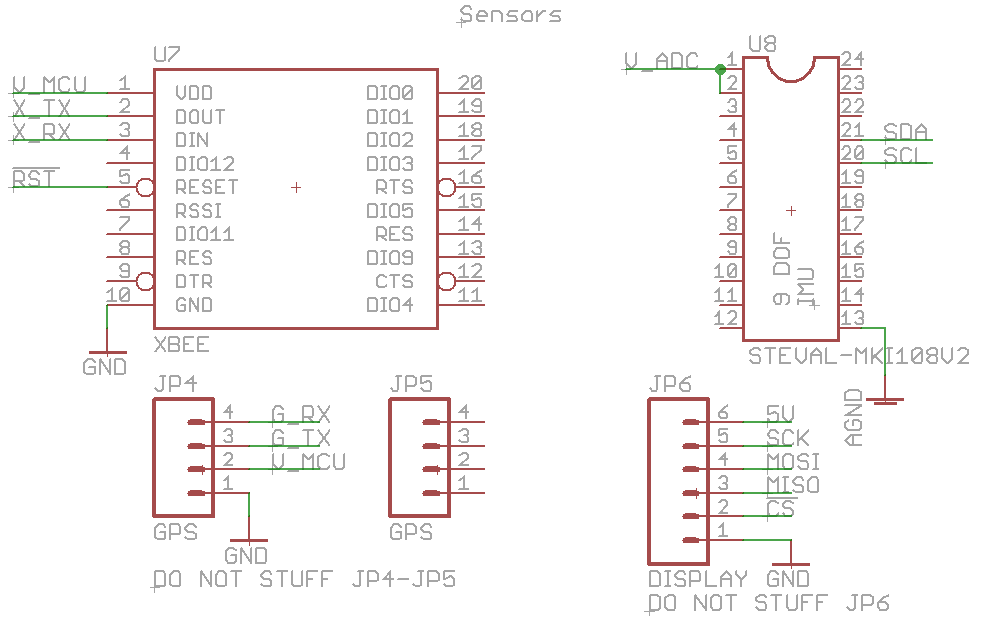


Figure 6. IMU, XBee, GPS headers, Display headers

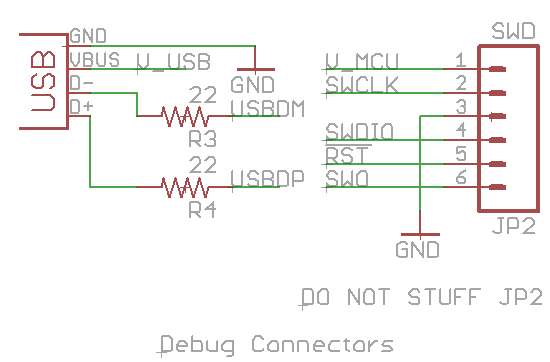


Figure 7. Headers for USB and Microcontroller Debugger

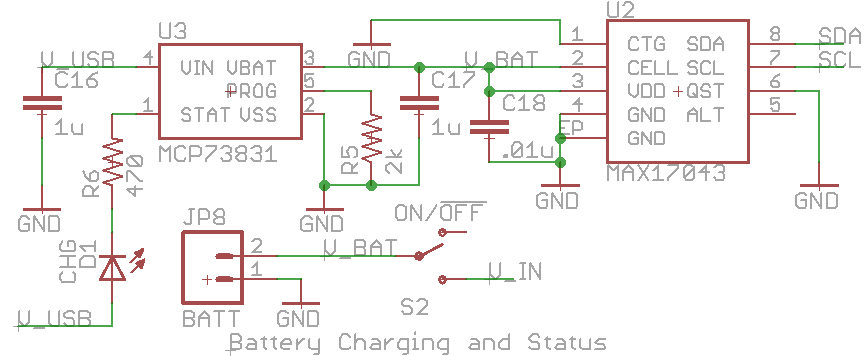


Figure 8. Li-Ion Battery Charger + Battery “Fuel Gauge”

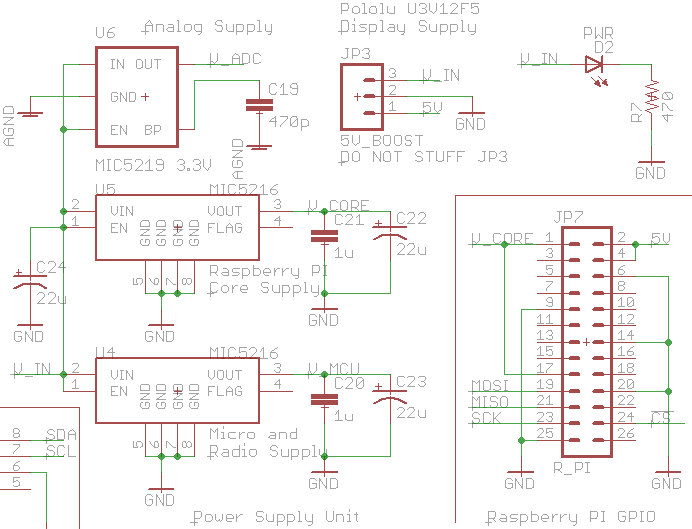


Figure 9. Power Supplies + Raspberry Pi Header