Software Overview

Year: 2017 Semester: Fall Team: 3 Project: Virtual Sports

Creation Date: September 1, 2017 Last Modified: September 8, 2017

Author: Chia-Hua Peng Email: peng81@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Software Overview** | 5 | x2 | 10 |  |
| **Description of Algorithms** | 5 | x2 | 10 |  |
| **Description of Data Structures** | 5 | x2 | 10 |  |
| **Program Flowcharts** | 5 | x3 | 15 |  |
| **State Machine Diagrams** | 5 | x3 | 15 |  |
|  | | | | |
| **Spelling and Grammar** | 5 | x2 | 10 |  |
| **Formatting and Citations** | 5 | x1 | 5 |  |
| **Figures and Graphs** | 5 | x2 | 10 |  |
| **Technical Writing Style** | 5 | x3 | 15 |  |
| **Total Score** | 100 | | |  |

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

General Comments:

*Very well written in general with comprehensive clarifications.*

1.0 Software Overview

The Virtual Sport project utilizes a variety of peripherals to visualize haptic data and to provide touch feedback. It requires two software components – a VR application running on a mobile device and a program operating on a microcontroller. The two components will be connected over Bluetooth. The software overview is demonstrated in Figure 1.

The first component, the VR app, is what will immerse the user in 360-degree visuals. The app collects hand orientation data as well as pushbutton inputs from the microcontroller and visualizes the digital sword movement. It also tracks the physics and collision information and tells the microcontroller to provide haptic feedback when the user swings the digital sword. This software component is divided into three blocks – a Bluetooth controller, a movement handler, and a haptic manager. The Bluetooth controller is responsible for communicating the app with the microcontroller. It reads and parses the data at a constant rate and stores the information in a buffer. The movement handler then takes the data from the buffer and uses a data visualization algorithm to display the digital sword movement. Finally, the haptic manager logs the physics and collision information, determines the type of haptic feedback the hardware needs to provide, and requests the Bluetooth controller to send a haptic feedback instruction back to the microcontroller.

The second component is the program running on the microcontroller. It is the control center that governs the hardware peripherals. The microcontroller retrieves the accelerometer values as well as the pushbutton status and packs the data in a buffer. Then, it transmits the information to the VR app over a Bluetooth module. The microcontroller also receives haptic instructions from the VR app, uses a table lookup algorithm to decode the commands, and updates the haptic driver and PWM parameters accordingly. The motor devices will receive pulses from the haptic driver and make proper touch feedback.

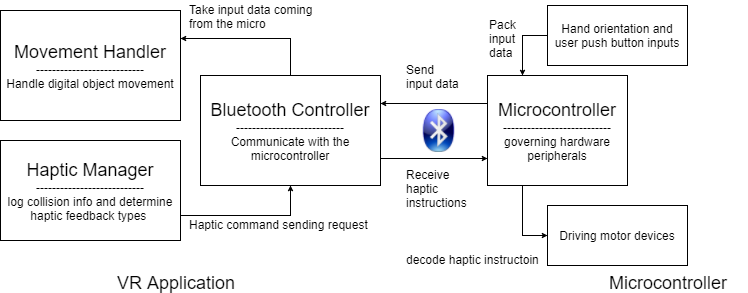


Figure 1 Software Overview

2.0 Description of Algorithms

The VR application utilizes a variety of algorithms for visualizing the data. It will be designed in Unity3D and coded in C#. Again, this software component consists of three blocks – the Bluetooth controller, the movement handler, and the haptic manager. A low-level block diagram for this software component is demonstrated in Appendix 1-1.

The Bluetooth controller can be described as a state machine. After the app is launched, it will scan the Bluetooth module using the device name and service UUID and attempt to make a connection if the device is found. If the device is not found or if the connection is weak, the controller will stay in the device scanning mode. The state machine diagrams are demonstrated in Appendix 2-1. After the device is connected, it will ping the input data feeding from the microcontroller and store the values in a buffer. The Bluetooth controller will also send haptic instructions to the microcontroller in an event driven fashion. This software block will be designed using the Bluetooth LE Unity plugin created by the Shatalmic Company. According to its datasheets [1], [2], the plugin provides basic access to the Core Bluetooth API for iOS and the GATT Bluetooth API for Android. The app will be able to be deployed on both iOS and Android devices, but this project will focus on the iOS version only. The plugin also has the capability of logging the Bluetooth RSSI (Received Signal Strength Indication) value. This value will be used to determine the state transition between the device scanning mode and active mode.

The movement handler is going to read and visualize the data stored in the Bluetooth controller buffer. It will perform a mapping algorithm and convert the accelerometer data to the Euler angle format adopted by Unity. The user’s vision is completely immersed in the virtual environment. He or she cannot see the actual device movement. The design requirement is satisfied as long as the tilting orientation of the digital sword matches the perceived touch feelings. The ADXL345 accelerometer is one of the selected candidate that will be used in this project. According to its datasheet [3], the module is well suited for tilt-sensing applications. With a 0.5 % non-linearity, the device should be capable of determining the hand tilting angles at an accuracy that meets the design specification. An accelerometer measures the difference between the linear acceleration in the module’s reference frame and the gravitational field vector. A sample output of the module is described in Figure 2. The software will convert the raw data to tilt angles using the formula described in Figure 3. This algorithm is adopted from the BMA220 accelerometer data sheet [4]. The data will also go through a high pass filter to prevent the tiny hand shaking vibration from affecting the visualization. If the rotating angle does not reach the threshold, the rotation will be ignored. If the threshold is reached, the movement handler will compute the angles the digital sword needs to rotate and visualize the movement in the virtual environment. Apart from rotating the digital sword, the movement handler also uses pushbutton input to manipulate player settings.

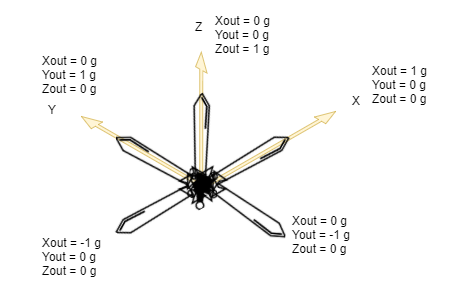


Figure 2 Sample Accelerometer Output at Different Tilting Angles

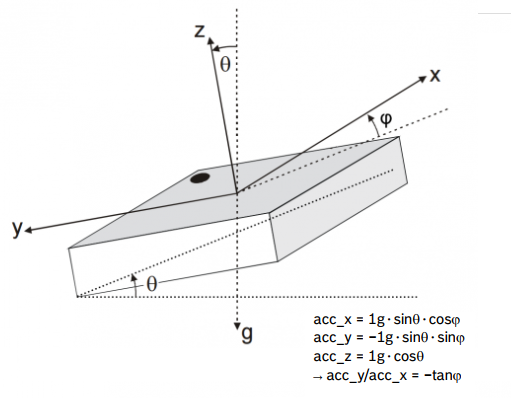


Figure 3 Accelerometer Data to Tilting Angle Conversion Algorithm [4]

The haptic manager block will log the collision information from the Unity physics engine and use conditional statements to determine the types of haptic feedback the hardware needs to provide. It will handle the vibration intensity as well as the vibration duration according to the impulses detected by the physics engine. The haptic manager also determines the moving directions of the sliding contact handle based on the orientations of the digital sword. It will request the Bluetooth controller block to send corresponding haptic instructions to the microcontroller. A sample of haptic commands is described in the data structures section of the report.

The algorithm for the microcontroller operation is relatively simpler. The majority of what it will be doing is collecting sensor data and governing the haptic feedback mechanism. A block diagram for this software component is described in Appendix 1-2. Protocols such as I2C and UART are utilized to communicate the microcontroller with accelerometer, haptic driver, and Bluetooth. The microcontroller retrieves the accelerometer data through an I2C interface, packs the values with the pushbutton status, and transmits the information to the VR app over Bluetooth. The microcontroller also receives haptic commands coming from the VR app and tells the haptic driver to make a response. The feedback mechanism can be described as a state machine in Appendix 2-2. A haptic response is triggered when the state machine receives a change signal which indicates that the VR app has sent a new instruction. The output logic block will decode the haptic command and export the haptic driver parameters as well as PWM duty cycles. This will cause the mechanical component to vibrate at different intensity and have the contact handle slide up and down. The DRV2605 haptic driver is the selected module that will drive the vibration motors. It is connected with the microcontroller through an I2C protocol. According to its datasheet [5], the module can provide a various waveform sequences. The amplitude as well as the duration can be easily adjusted by changing register values and these values are the haptic driver parameters in the state machine output logic.

3.0 Description of Data Structures

The Virtual Sport project will use the UART protocol with a Bluetooth module to communicate the microcontroller with the VR application. The data transmitting over the Bluetooth will have the format demonstrated in Figure 4. The meaning of each packet field is described in Table 1.

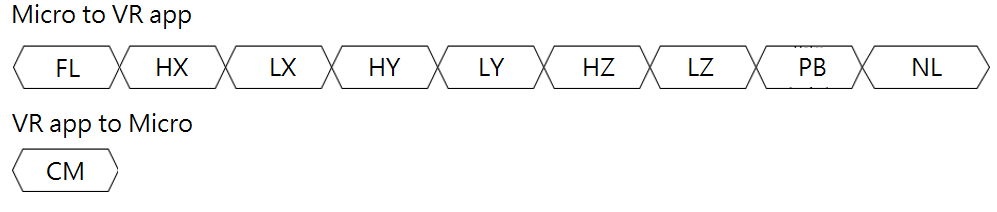


Figure 4 Wireless Transmission Data Structure

Table 1 Bluetooth Transmission Packet Data Structure Reference

|  |  |
| --- | --- |
| **Field** | **Meaning** |
| FL | The type of packet being sent. This byte will be 0x01 for this type of format. Additional format may be added for future development. |
| HX | The higher byte of X-axis value retrieved from the accelerometer |
| LX | The lower byte of X-axis value retrieved from the accelerometer |
| HY | The higher byte of Y-axis value retrieved from the accelerometer |
| LY | The lower byte of Y-axis value retrieved from the accelerometer |
| HZ | The higher byte of Z-axis value retrieved from the accelerometer |
| LZ | The lower byte of Z-axis value retrieved from the accelerometer |
| PB | 0x01 indicates that pushbutton is pressed; 0x00 otherwise |
| NL | New line character (End of transmit packet byte) |
| CM | Haptic command  Higher 4 Bits Lower 4 Bits  ‘0000’ – No Vibration ‘0000’ – No torque feedback  ‘0001’ – Vibration with low intensity ‘0001’ – Clockwise torque cue  ‘0010’ – Vibration with medium intensity ‘0010’ – Counter-clockwise torque cue  ‘0011’ – Vibration with high intensity ‘0011’ – Vibration Torque Cue |

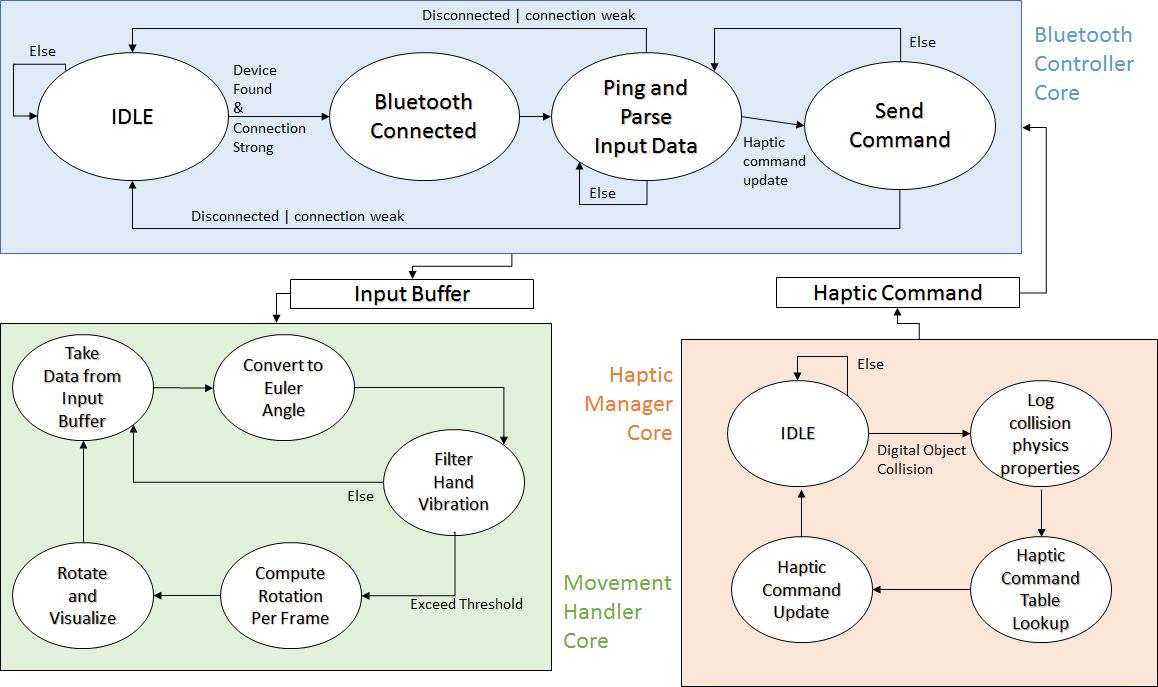
The Bluetooth controller in the VR app will read and parse the “Micro to VR app” packet. The first byte indicates the packet format and contains the value 0x01. The VR app will send the haptic command byte to the microcontroller, and the meaning of each command is described in table 1. Additional packet format and haptic command will be added during the design prototype stage.

Apart from the data transmission packet, the microcontroller and the VR app also use various variables to store information such as the digital sword rotating vector, the Bluetooth controller status, and the I2C device address.

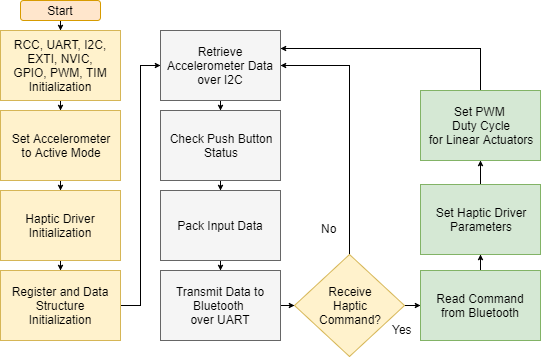
4.0 Sources Cited:

1. Shatalmic, LLC. *Unity Bluetooth LE Plugin for iOS*. [Proprietary].
2. Shatalmic, LLC. *Unity Bluetooth LE Plugin for Android*. [Proprietary].
3. Analog Devices, Inc. *Digital Accelerometer ADXL345*. [Online]. Available: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf>
4. Bosch Sensortec. *BMA 220 Digital triaxial accelerometer sensor*. [Online] Available: <http://image.dfrobot.com/image/data/SEN0168/BMA220%20datasheet.pdf>
5. Texas Instrument. *DRV2605 Setup Guide*. [Online]. Available: <http://www.ti.com/lit/an/sloa189/sloa189.pdf>

Appendix 1: Program Flowcharts

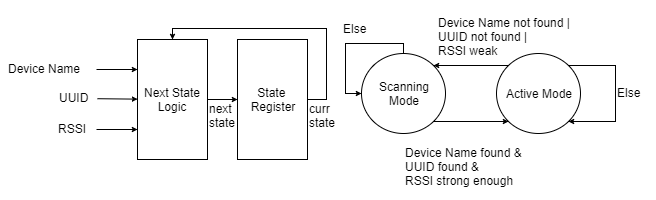


**Appendix 1-1 VR Application Flow Diagram**

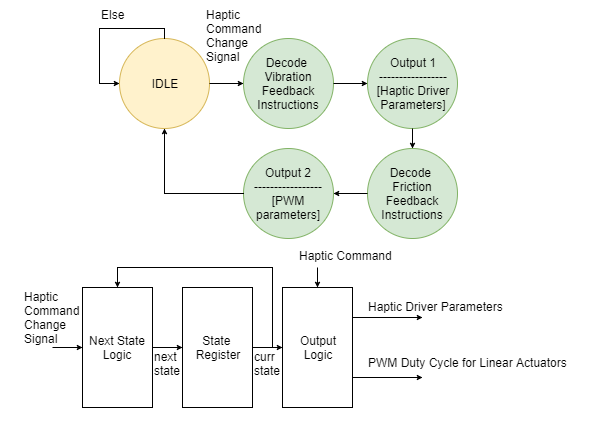


**Appendix 1-2 Microcontroller Software Flow Diagram**

Appendix 2: State Machine Diagrams



**Appendix 2-1 Bluetooth Controller Mode State Machine**



**Appendix 2-2 Haptic Response State Machine Diagrams**