Software Formalization

**Year:** October 12, 2017 **Semester:** Fall **Team:** 3 **Project:** Virtual Sport

Creation Date: ­10/06/2017 Last Modified: December 7, 2017

Author: Yutao Hu Email: hu298@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** | 5 | x2 | 10 |  |
| **Description of Components** | 4 | X3 | 12 | See the comments in section 2.1.2.2 and 2.2.1. |
| **Testing Plan** | 5 | x3 | 15 |  |
| **Software Component Diagram** | 5 | x4 | 20 |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** | 4 | x2 | 8 | See the tracked changes. |
| **Formatting and Citations** | 4 | x1 | 4 | See the tracked changes. |
| **Figures and Graphs** | 5 | x2 | 10 |  |
| **Technical Writing Style** | 4 | x3 | 12 | See the tracked changes. |
| **Total Score** | 91 | | |  |

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

General Comments:

*Well written in general with details, but could benefit from more clarifications. The formatting changes should be viewed in “Simple Markup” mode.*

1. Utilization of Third Party Software

The project utilizes Unity3D[1], STM32 HAL drivers[2], STM CubeMX[3], Google VR SDK[4], and Bluetooth LE Unity plugin[5].

The detail of the software is in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | License | Description | Use |
| Unity3D | Unity Personal  (Free to use and sell) | This is a 3d game engine which allows for modeling objects in a virtual environment. | The Virtual Sport project utilizes this software to design the virtual reality scenario. |
| STM32 HAL drivers | Berkeley Software Distribution  (Free to use and sell) | The HAL driver layer provides a generic multi-instance set of APIs to interact with the upper layer. | The Virtual Sport project uses the HAL driver to control the TIM, PWM, I2C and UART modules. |
| STM CubeMX | Personal  (Free to use and sell) | The code generator for STM microprocessor initialization. | The Virtual Sport project utilizes this software to generate initializations, which include clock configurations, GPIO settings, and timer parameters. |
| Google VR SDK | Open Source  (Free to use and sell) | The Google VR SDK for Unity provides additional features for Cardboard. | The Virtual Sport project uses this SDK to enable the Virtual Reality capabilities on an iOS device. |
| Bluetooth LE Unity plugin | Purchased license | This is a Unity plugin that provides the access to the CoreBluetooth API in iOS and the GATT Bluetooth API in Android. | The Virtual Sport project uses this API to exchange information between the VR app and the microcontroller. |

1. Description of Software Components

The software for Visual Sport consists of two major components: microcontroller software and a mobile app. The microcontroller will transmit and receive data with its peripherals and communicate with the phone to finish a series of tasks. The mobile app would work with a VR headset to simulate a virtual world.

2.1 Microcontroller Software

The software in the microcontroller is divided into several modules, specifically the TIME PWM module, I2C module, and UART module.

2.1.1 TIM4 PWM

The TIM4 module will be used to generate PWM output and to drive the servo that provides haptic feedback. In the servo update, the micro will accept the data that it receives from Bluetooth. After receiving the data, the micro calls functions according to the command to update the PWM duty cycle and to have the servo slide the contact handle up and down.

Void slide\_up(int length) will receive an integer that indicates the length of movement, and this value would be converted to a rational value for the motor. This function will not return anything because this is a command function.

Void slide\_down(int length) will do the similar job as slide\_up(), but it will move in the opposite direction.

2.1.2 I2C

2.1.2.1 Accelerometer

The ADXL345 accelerometer is the selected candidate in this project. According to its datasheet [6], the module is well suited for tilt-sensing applications.

The microcontroller uses the HAL library to communicate with the chip over an I2C interface. It enables the accelerometer’s power mode by setting the corresponding control register during the initialization stage. After the device is powered on, the microcontroller pings the accelerometer at a constant rate and stores the information in a buffer.

HAL\_I2C\_Master\_Receive() will be called to receive the data from the accelerometer. This is a function from HAL library.

2.1.2.2 Haptic Driver/ Vibration Motor Update Logic

The microcontroller also uses I2C to communicate with the haptic driver, a device which receives instructions from an I2C interface on one side and uses PWM to drive the vibration motor on the other side. Although the micro has a PWM module, we decided to isolate the motor control for safety and clarity. A vibration motor control chip will be soldered to the PCB, and the microcontroller will use I2C to send commands to control the motor. The microcontroller decodes the command sent from the VR app and calls functions to generate the corresponding vibration waveform.

HAL\_I2C\_Master\_Transmit() will be used to send the command data. The motor control chip will decode this data and send different PWM signals to give different vibrations according to the command.

2.1.3 GPIO

Apart from rotating the digital sword, the movement handler also uses the pushbutton input to manipulate the player settings. The reading from the digital IO pin would be sent together with the accelerometer data as a packet using the Bluetooth\_Send\_packet() function.

2.1.4 UART

Bluetooth LE

The microprocessor uses the Bluetooth LE peripheral to communicate with outside world.

HAL\_UART\_Receive() is used to get the data from VR app to the microprocessor. As shown in Figure 1, this function would read the command(CM) from the VR app.

Bluetooth\_Send\_packet() is used to send the data packet from the microprocessor to VR app.

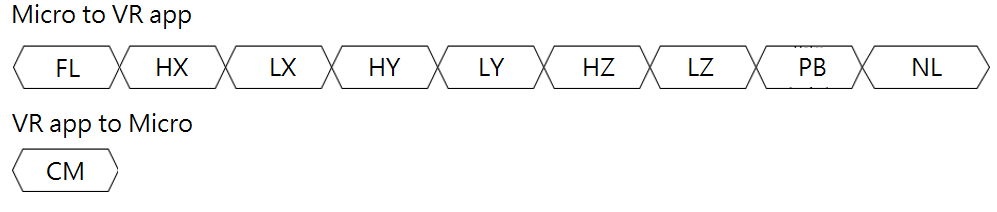


Figure 1 Wireless Transmission Data Structure [8]

2.2 Phone App

The phone app is composed of a virtual display and a backend algorithm for opponents.

2.2.1 Virtual Display

Sword

The sword is rotated based on the accelerometer data. 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. [8]

calculate\_angle() will be called to calculate the tilting angle according to the raw data of linear acceleration in the module’s reference frame and the gravitational field vector. This algorithm is learned from the BMA220 accelerometer data sheet (Figure 2) [7]. With a 0.5 % non-linearity, the device can determine the hand tilting angles at an accuracy that meets the design specification.

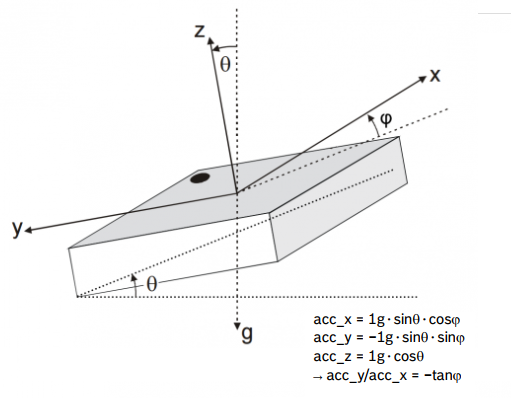


Figure 2 Accelerometer Data to Tilting Angle Conversion Algorithm [7]

Hitting target

The hitting target will move when it collides with the sword. The movement of objects should obey the laws of physics, such as gravity, elastic collision, inelastic collision, explosion and so on.

2.2.2 Backend algorithm for Opponent

One of the training scenes is fighting with another man in the virtual world. The AI Non-Player Character (NPC) would have to move randomly, and the player’s task is to find the correct timing to wave the sword to hit the NPC.

1. Testing Plan
   1. UART and Bluetooth

This test is to ensure the UART sends Bluetooth a data packet and reads command from it successfully. To test this, we will complete the following steps.

1. Confirm that the Unity app has received the packet from microcontroller by printing the packet.
2. Confirm that the Unity app has sent the command data by tracking the command using the Keil debugger.
3. Confirm that the microcontroller can receive commands from the VR app by toggling an LED, vibrating the motor, and/ or rotating the servo.

These will be the tests to ensure both UART and Bluetooth working correctly.

* 1. I2C and Accelerometer

This test is conducted to check if the accelerometer accurately senses the acceleration value and sends the value back to the microprocessor. For this test, Keil Microvision IDE will be used to display the values of data.

1. Connect the microprocessor with a computer that is running Keil Microvision IDE and the accelerometer.
2. Read the data from accelerometer through I2C and display information on the debugger watch window.
3. Tilt the accelerometer to the orientation demonstrated in the figure below and verify if the three axis output matches with what is in the datasheet.

These will be the tests to ensure both I2C and accelerometer working correctly.

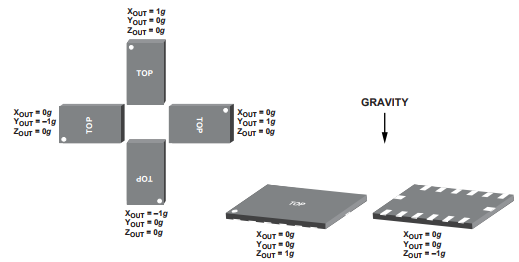


Figure 3 Chip’s Orientation

* 1. PWM and Servo

This test is conducted to check if the servo can make the haptic slider to move up and down. We also need to test if the torque is enough for our design.

1. Print the 3D modelled gear, and install the servo to the gear and slider model.
2. Send sliding signals from microprocessor PWM module in quick succession to confirm it is moving agilely.
3. Use hand to push some force to the slider model and make sure the servo gives enough force and run correctly.

These will be the tests to ensure both the servo and 3D slider model working correctly.

* 1. Unity Scene

This test is conducted to check if the virtual environment works correctly. To achieve this goal, we will simulate the environment in Unity. The Unity keyboard’s up, down, left and right keys will be used to simulate the sword movement.

1. Use the keyboard to move the sword in up, down, left and right four directions as shown in Figure 4.
2. Hit a stack of boxes in the scene to make the top boxes fall from the bottom one. Make sure the collision works correctly.
3. Fight a still NPC in the scene with a sword held in hand. When the number of contacts with NPC’s sword is exactly 3 times, NPC would drop his sword.
4. Turn on the microcontroller, rotate the accelerometer to the orientation described in the figure below, and verify if the sword is tilting towards the desired orientation. Because the user’s vision is completely immersed in the virtual environment, none of the physical components will be seen by the user. As long as the tilting display matches the feeling user perceives, the success criteria is satisfied.

These will be the tests to ensure our virtual environment working correctly.

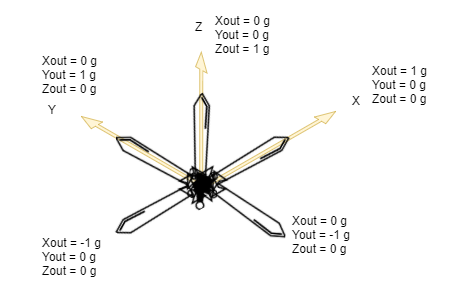


Figure 4 The Sword Orientation

1. Sources Cited:

[1] U. Technologies, “Welcome to the Unity Scripting Reference!,” Unity - Scripting API: [Online]. Available: <https://docs.unity3d.com/ScriptReference/>. [Accessed: 30-Sep-2017].

[2] “UM1725 User Manual .” [Online] Available: <http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf>. [Accessed: 30-Sep-2017]

[3] “STM32Cube initialization code generator” [Online] Available: <http://www.st.com/en/development-tools/stm32cubemx.html>. [Accessed: 30-Sep-2017]

[4] “Google VR SDK for Unity” [Online] Available: <https://developers.google.com/vr/unity/>. [Accessed: 30-Sep-2017]

[5] “Bluetooth LE for iOS, tvOS and Android” [Online] Available: <https://www.assetstore.unity3d.com/en/#!/content/26661>. [Accessed: 30-Sep-2017]

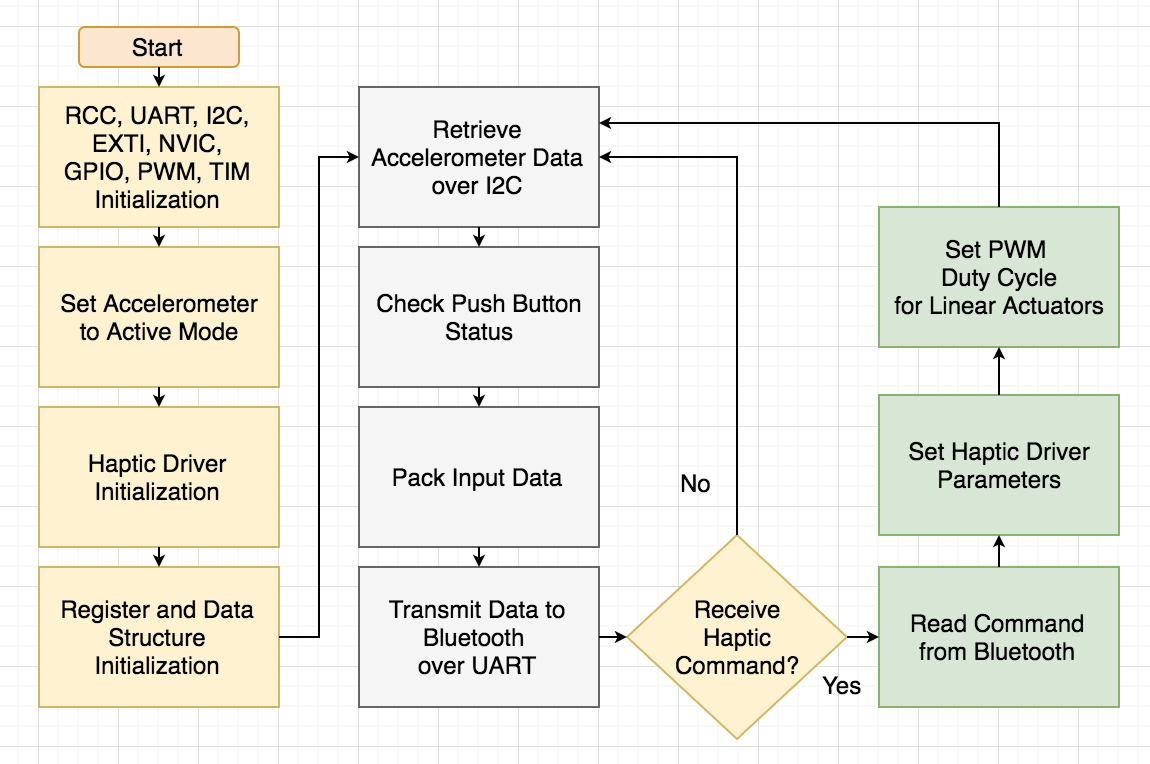
[6] Analog Devices, Inc. *Digital Accelerometer ADXL345*. [Online]. Available: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf> [Accessed: 30-Sep-2017]

[7] Bosch Sensortec. *BMA 220 Digital triaxial accelerometer sensor*. [Online] Available: <http://image.dfrobot.com/image/data/SEN0168/BMA220%20datasheet.pdf> [Accessed: 30-Sep-2017]

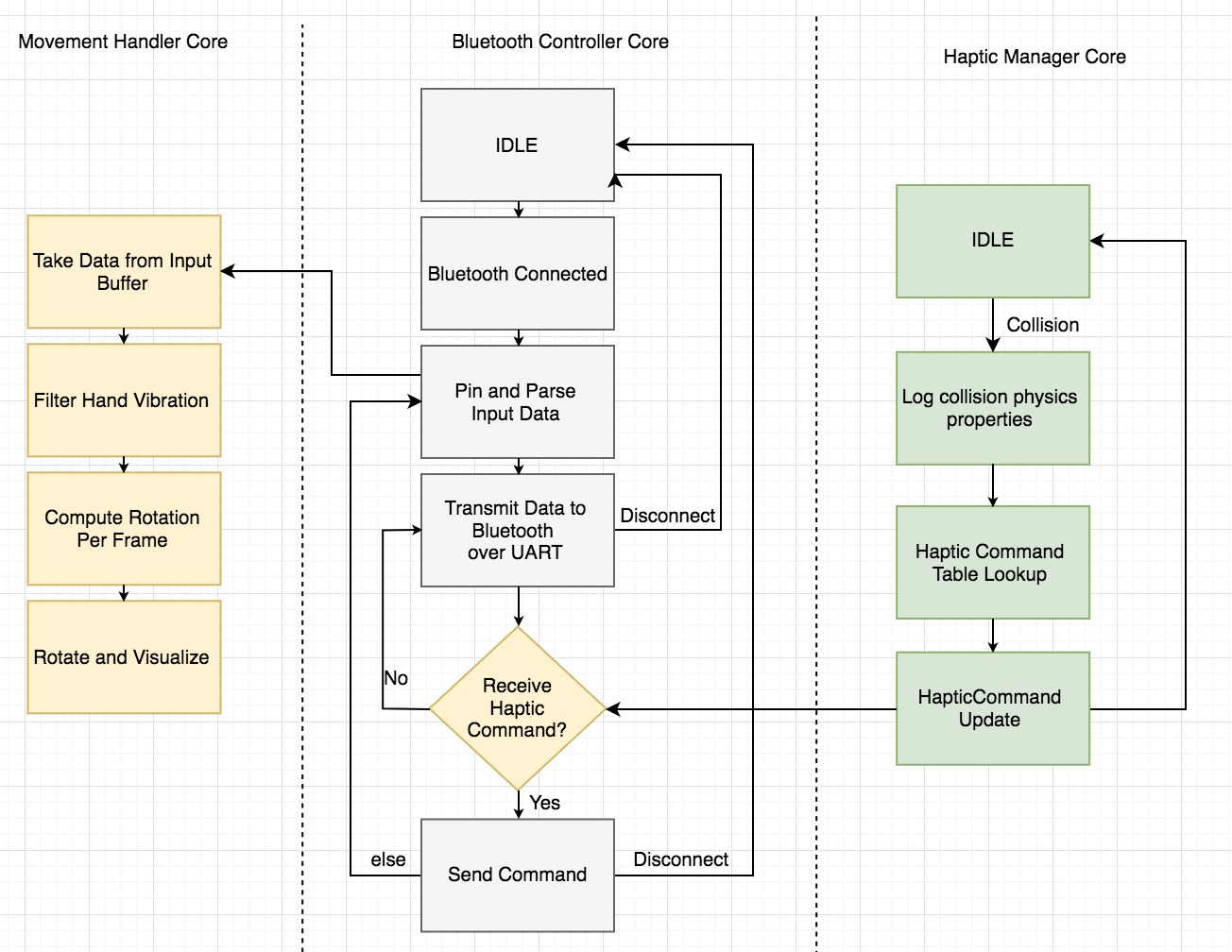
[8] Chia-Hua Peng. *Software Overview*. [Online] Available: <https://engineering.purdue.edu/477grp3/Files/documents.html>. [Accessed: 30-Sep-2017]

Appendix 1: Software Component Diagram

Microcontroller Program Flow Diagram

**

Phone APP Program Flow Diagram



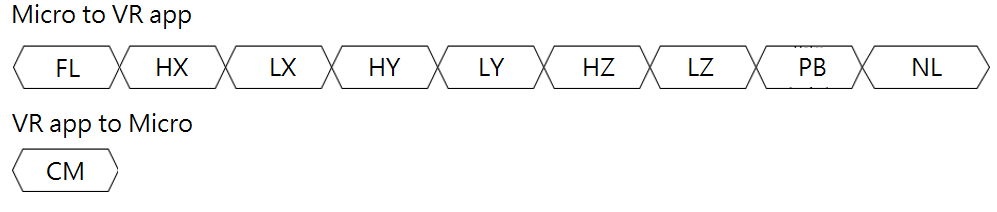


Figure 5 Wireless Transmission Data Structure [8]

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 |