UNIVERSITY OF
BIRMINGHAM

MSc Small Embedded Systems (2020)

The Assignment Report

Lyu Xinghan 2006850

January 20, 2020

Contents

Chapter 1
Introduction

The objective of the assignment is to design a game controller for Augmented Reality using various sensors, communication protocols and system architecture. The report details the design and how to select the various components in the concept. It includes product descriptions, how someone will use the product, and how the product compares to competing products. The report also introduces the specific hardware design of the concept, such as selecting the appropriate sensor, processor, communication protocol, and using a flowchart and circuit schematic to explain the design in detail. Besides, the report introduces specific software design, such as using some key pseudo-codes to explain how the sensor's data is passed to the processor and how to implement the function of the game controller. Finally, it indicates how power requirements vary with the operation, and the choice of battery and non-technical requirements are evaluated.

Chapter 2
Product Description

This report designed a gravity-sensing game controller that can control the movement of a car in a computer racing game. The player can control the car to move forward and backwards by tilting the game controller, and at the same time, the player can intuitively observe the tilt angle of the game controller through the LCD on it.

2.1 Purpose of the product and the likely users

The purpose of the product is to give players a new gaming experience by designing a gravity-sensing game controller, a physical handle without arrow keys, for an Augmented Reality racing game on the computer. Players can operate the game controller, such as tilting forward to make the car move forward and tilting to the left to make the car turn left. This is far more fun than pressing the up, down, left and right keys on the screen of phones or pads. Therefore, it provides super fun of virtual driving experience during the game.

Players who are potential uses are those who're not satisfied with the simple keyboard and mouse for game operations. Today's computer game market is booming and stable, driving the consumption of peripheral game products, such as PC game controllers. Players are not satisfied with the simple keyboard and mouse for game operations. The gamepad attracts consumers with its comprehensive and professional features. The gamepad technology is mature, the types and functions are becoming more and more abundant, and people with different hobbies can find suitable options. Moreover, as the age of game players continues to expand, the number continues to increase, and the base of potential gamepad purchasers continues to increase.

2.2 How someone will use the product

The game controller requires an external power supply. Players should load the battery before using it. The controller provides a Wi-Fi module to connect with the computer. After the connection is successful, users can control the car to start the game.

2.3 Compare the product with competitor product

We compare this product with a Nintendo Switch, also a game controller. Nintendo games have core players with high loyalty to a series of games operating by the controller, which has brought many sales to its products, and its company has launched a series of supporting games for the switch controller. These games are expensive to develop, as a result, game controllers are expensive and out of the spending power of most users. Our game controllers are cheap and easy to play and are suitable for beginners.

Chapter 3
Hardware Design

3.1 Selecting the key sensor, module hardware and communication protocols

Selecting the right sensor and hardware module is the focus of hardware design. First, choosing the right sensor is a significant part of the hardware design. The accuracy of the sensor directly affects the user's gaming experience. For example, if an appropriate and high-precision gravity acceleration angle sensor is used, the user gets good operation feedback during the operation of the game controller. Second, adding an LCD screen to the game controller that can display the status of the game character, the car in motion. It can intuitively and allow the player to operate the controller better and obtain an excellent gaming experience. Third, using a Wi-Fi module can wirelessly connect the game controller with the computer, which significantly improves the operating space for gamers.

3.1.1 ADXL345

A digital accelerometer should be appropriately selected to detect the change of the inclination angle of the game controller. Then, users control the direction and speed of the car by tilting the game controller three-dimensionally. There are two conventional 3-Axis acceleration sensors as follows:
1) The MPU6050 (Full link of the datasheet in Appendix [1])
2) The 3-Axis Digital Accelerometer ADXL345 (Full link of the datasheet in Appendix [2])

First of all, The MPU-60X0 is the world’s first integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP). Though it can be very constructive and accurate for applications of game controllers, it is very tough and complex to use the onboard DMP, which means programming it is an advanced topic. Besides, it’s expensive, £1.00 for each.

While compared with MPU6050, the ADXL345, a 3-axis accelerometer, is suitable for tilt angle measurement and can perform static gravity acceleration detection. At the same time, it is also suitable for tracking the state of movement and measuring the instantaneous acceleration caused by movement. Its high resolution (4mg / LSB) enables it to sense tilt angles that vary less than 1 °. Besides, it is accessible via SPI or I2C digital interface and extremely simple to use with the Arduino extension boards. Therefore, using a simple ADC and taking readings from the accelerometer can be very useful and handy. And what’s more, it’s cheaper than MPU6050, £0.99 each. Thus, we choose ADXL345 as the 3-axis accelerometer sensor in the game controller.

3.1.2 LCD1602

Add an LCD1602 screen on the game controller to display the status of the car, and visually display the tilt angle, so that players can better operate the controller. The LCD1602 capacity is 16 × 2 characters, which is enough to display the three-axis acceleration. The working voltage of the chip would be 3V or 5V, the Arduino development board supports this optimal working voltage. It's sold on eBay with the price of £1.91 for one.

3.1.3 Wi-Fi module

We can use the ESP8266 Wi-Fi module to enable the game controller to communicate with the computer. [4]

3.1.4 Communications protocols

ADXL345 supports two communication modes, SPI and I2C. We choose the latter one - I2C. And LCD is supported I2C communication mode as well. If the CS pin is connected high, I2C mode is enabled. If the CS pin is not connected, there is no default mode, so it should always be connected high or driven by an external controller. When LCD1602 is connected to the same I2C bus, the nominal operating voltage level of those other devices cannot exceed VDD I/O by more than 0.3 V. External pull-up resistors are necessary for proper I2C operation. Used in this connection diagram are two 4.7 kΩ resistors. I2C addresses several drawbacks in the other communication protocols, giving it an advantage over the others in some applications. These include:

- The ability to connect multiple masters to multiple slaves
- Synchronicity (just like SPI), which means higher speed communication
- Simplicity: implementation only requires two wires and some resistors

3.2 Selecting of the main processor and motherboard

After selecting the appropriate sensors and modules, determine the appropriate processor and development board to carry out the overall circuit design of the game controller. The central processor at the core of the game controller is the ESP8266, contained the Wi-Fi module that enables the communication between game controller and PC, implemented on an Arduino based expansion board known as the Arduino Feather HUZZAH (AFH). AFH's 3.3V voltage output can give LCD1602 the best working voltage. AFH supports SPI and I2C communication methods and can realise data transmission with ADXL345 and LCD1602. Besides, it's easy to utilise this extensional board for rapid prototyping. And it's available on eBay for less than £20 each.

3.3 Power

We can make the AFH easy to power both when connected to a computer as well as via battery. There are two ways to power a Feather. One way is to connect with a MicroUSB cable; another way is to connect 4.2/3.7V Lithium Polymer (Lipo/Lipoly) or Lithium-Ion (LiIon) battery to the JST jack. This will let the Feather run on a rechargeable battery. When the USB power is powered, it will automatically switch over to USB for power, as well as start charging the battery (if attached) at 100mA. [4] (Full link for the datasheet of ESP8266 in Appendix [4]).

In this game controller design, neither of the above is appropriate. Because charging with MicroUSB cable does not meet the original intention of the wireless game controller and the lifetime significantly reduced if connected with 3.7V lithium battery. Considering that LCD1602 consumes abundant energy and the game controller should have long battery life. Therefore, two 9V batteries in parallel are used to power the game controller. Then, the power capacity is 1000mAh. 9V batteries are available on eBay with a price of £2.99 for 2.

3.4 Design Architecture

3.4.1 High-level design and architecture

Before we simulate the schematic of the game controller, a high-level view of the game controller and the components within its outer casing is shown in figure 3.1. The central blue box in the centre of the diagram with the white text is merely representing the main processor ESP8266 and board AFH. The 3-axis digital accelerometer ADXL345 and display module hardware LCD1602 are detailed within the light blue box and purple box. These two connect and interface directly to the AFH via communication type I2C. The Wi-Fi module is detailed within the yellow box inside the blue box because the AFH ESP8266 contains the Wi-Fi module, which is available to communicate with PC. The primary power source for the game controller is two 9V batteries in parallel to double the power capacity for the design to give it longer battery life.

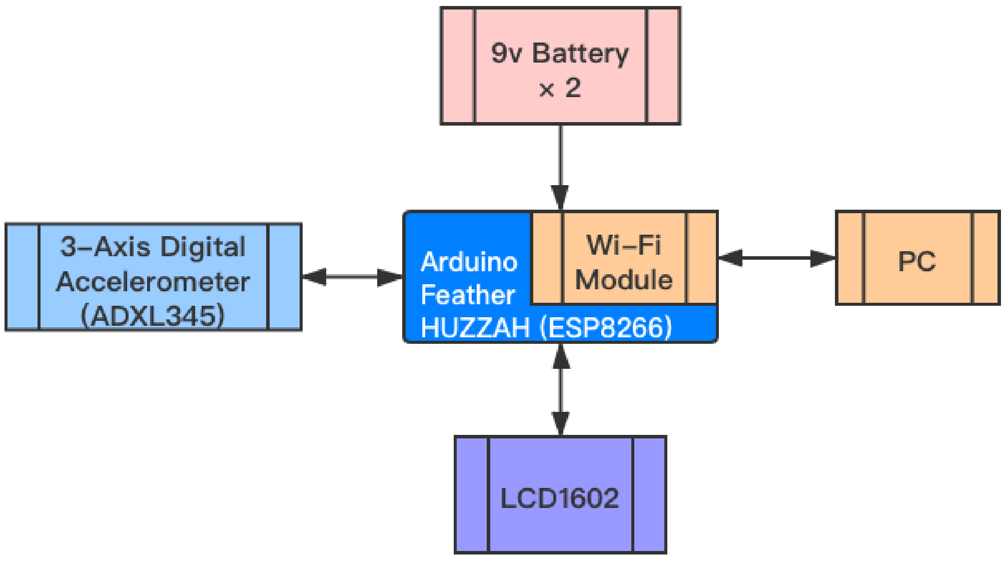
9v Battery × 2
3–Axis Digital Accelerometer (ADXL345)
Arduino Feather HUZZAH (ESP8266)
Wi–Fi Module
PC
LCD1602

Figure 3.1 The high-level view of the game controller

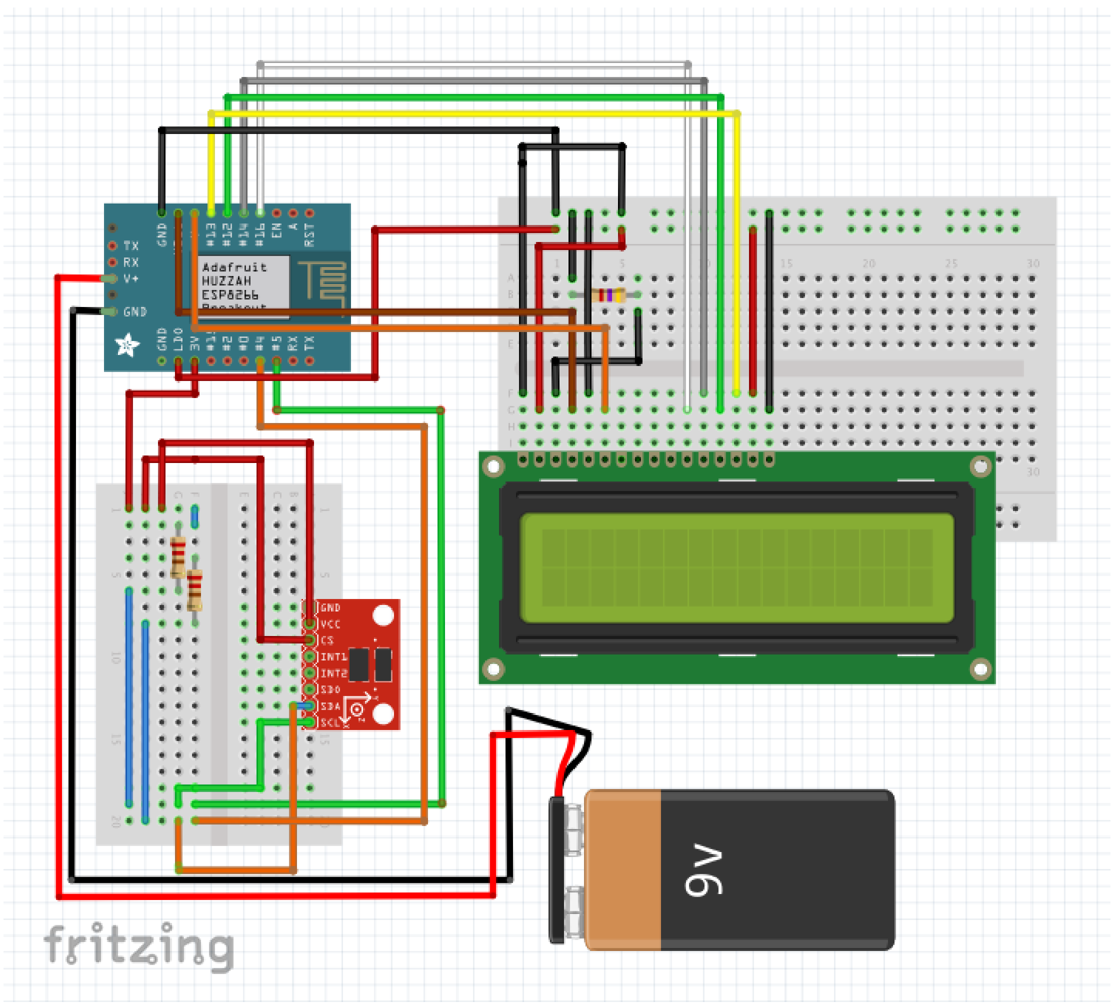

Figure 3.2 Figure to show the full electronic schematic and all of its elements via a breadboard based drawing

3.4.2 Constrains

1) Size: 14cm × 10cm × 4cm (length × width × height)
2) Weight: less than 200g
3) Power and battery life: 9V battery × 2, 500mAh × 2 = 1000mAh
4) Operating Envirnment: 10 ℃ - 25 ℃, 65 ± 20% dry environment
5) Safety-Critical Functions: may be at risk of short circuit
6) Costs: £26.99 in total (included 9V battery × 2)

Chapter 4
Software Design

4.1 Memory needed for the software

Estimate the memory space required to implement the software. The ADXL345's integrated memory management system uses a 32-level first-in-first-out (FIFO) buffer that can be used to store data, thereby minimizing the load on the host processor and reducing overall system power consumption. Besides, the on-board memory in the AFH set provides sufficient functionality for the design. Therefore, the total needed ROM is less than 4k, even if the full program is at max around 2k lines (based on the assumption that a line of code requires 2 bytes of ROM).

4.2 Dataflow in the design

A simple diagram illustrating how the sensor data is passed to the processor and displaying the dataflow for the game controller is shown in figure 4.1. This diagram shows how the essential software works at a high level for each of the main components within the main design. Such as the 3-axis accelerometer sensor ADXL345 which sends the value of 3-axis accelerator and tilted angle to the processor for it to be analysed, as well as the processor passes the value to display on LCD1602 and packs the data then transmits it to PC through Wi-Fi module. Once the AFH and PC receive the data of accelerator and tilted angle, the calculations and checks are then completed by the code to determine what the correct course of motion is for the car in a racing game.

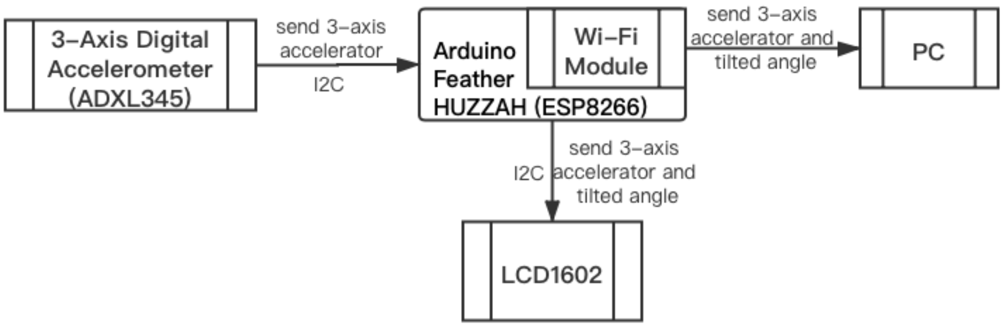
3–Axis Digital Accelerometer (ADXL345)
send 3–axis accelerator
I2C
Arduino Feather HUZZAH (ESP8266)
Wi–Fi Module
send 3–axis accelerator and tilted angle
PC
I2C
send 3–axis accelerator and tilted angle
LCD1602

Figure 4.1 Dataflow for the software

4.3 Examples of code

The software design steps are introduced below, showing the specific code and threads for implementing the function of the game controller.

(1) Initialize ADXL345: Initialize I2C; Set low-level interrupt output, 10-bit resolution and 16g range; If the initialization is successful, it returns 1, and if it fails, it returns 0.

(2) Write to ADXL345 register: Send write instruction; Send register address; Send Value; Generate a stop condition.

(3) Read from ADXL345 register: Send read instruction; Send register address; Read a bit and return the value.

(4) Read the value of the x, y and z axis 10 times and get an average result.

(5) Automatic calibration:

(6) Read 3-axis coordinate value 10 times then get an average result of each axix coordinate.

(7) Calcute and return the tilted angel.

(8) Processor passs the value to LCD1602, then it displays the value.

(9) Configuate Wi-Fi: Send *AT + CWMODE = 1* to configure the module to sta mode (parameters 1, 2 and 3 correspond to the modes sta, AP and sta / AP, respectively); Send *AT + CWLAP* instruction to scan the current nearby Wi-Fi, the module will return the list of available Aps; Use *AT + CWJAP = "WiFi name", "WiFi password"* to connect to the designated router. The returned *"WIFI CONNECTED"* indicates that the connection was successful, and *"WIFI GOT IP"* indicates that the module has been assigned an IP; Finally, use *AT + CWQAP* to disconnect the currently connected WiFi.

A part of code shows that how ADXL345 realises the function.

```
#include <Wire.h>
#define Register
int ADXAddress = 0xA7 >> 1;    // the default 7-bit slave address
int reading = 0;
int val=0;
int X0,X1,X_out;
int Y0,Y1,Y_out;
int Z1,Z0,Z_out;
double Xg,Yg,Zg;

void setup()
{
        Wire.begin();
        Serial.begin(9600);
        delay(100);                    // enable to measute g data
        Wire.beginTransmission(ADXAddress);
        Wire.write(Register_2D);
        Wire.write(8);                 //measuring enable
        Wire.endTransmission();        // stop transmitting
}

void loop()
{
  //--------------X as an eample, same with Y and Z
        Wire.beginTransmission(ADXAddress); // transmit to device
        Wire.write(Register_X0);
        Wire.write(Register_X1);
        Wire.endTransmission();
        Wire.requestFrom(ADXAddress,2);
        if(Wire.available()<=2)
        {
                X0 = Wire.read();
                 X1 = Wire.read();
                X1=X1<<8;
                X_out=X0+X1;
		}
		Serial.print();
		delay(200);
	}
```
Full example of code in Appendix [5].

Chapter 5
Evaluation

5.1 Power requirements

Considering power consumption is necessary on this game controller, an external battery (two 9V batteries in parallel, 1000mAh) powered the device. The energy consumption of the sensor, LCD and AFH is as follows.

First, in low power consumption mode, users can perform power management based on ADXL345 motion sensing, while only consuming deficient power consumption as low as 23 μA in measurement mode and 0.1 μA in standby mode at VS = 2.5 V. Second, the working current of LCD1602 is 2mA. When the backlight is not set to work, it consumes a small amount of power. Third, ESP8266EX is designed with advanced power management technologies. The power consumption is 0.5uA if it's shut down. The average current of ATH's Wi-Fi module is about 60mA when Wi-Fi TX and RX packet is active.

Now state the battery life would be:

(1) When the whole system is in active mode, it means the game controller is fully functional. After the calculation, the battery life would be 16.13 hours.

(2) When the whole system is in sleep mode, the battery life would be more than 1000 hours.

When the whole system is in active mode, the product performance is quite good. Players can control the game controller in real-time with low response time and can precisely control the motion of the car in a racing game on PC. As the battery power drains, the power supply is not enough for the Wi-Fi module to work functionally, and the delay between the game controller and the computer increases until it completely loses response.

5.2 Non-technical requirements

Some main non-technical requirements can be evaluated and discussed, such as the speed of the response between PC and game controller as the power drains. When the USB cable is powered, AFH will automatically switch over to USB for power, as well as start charging the battery (if attached) at 100mA. In this situation, users can connect the game controller with USB cable in the case that external batteries are running out of energy.

Chapter 6
Appendix

[1] (Full link for the datasheet of ADXL345) - https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf

[2] (Full link for the datasheet of MPU0605) - http://www.4tronix.co.uk/arduino/specs/mpu6050.pdf

[3] (Full link for the datasheet of LCD1602) - http://www.eigenstudium.de/fileadmin/user_upload/Download/Datenblaetter/LCD-1602A.pdf

[4] (Full link for the datasheet of ESP8266) - https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[5] (Full example of code) –

```
#include <Wire.h>
#include <LiquidCrystal.h>

#define Register_ID 0
#define Register_2D 0x2D
#define Register_X0 0x32
#define Register_X1 0x33
#define Register_Y0 0x34
#define Register_Y1 0x35
#define Register_Z0 0x36
#define Register_Z1 0x37

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
int ADXAddress = 0xA7>>1;
int reading = 0;
int val = 0;
int X0,X1,X_out;
int Y0,Y1,Y_out;
int Z1,Z0,Z_out;
double Xg,Yg,Zg;

void setup()
{
  lcd.begin(16, 2);
  delay(100);
  Wire.begin();
  delay(100);
  Wire.beginTransmission(ADXAddress);
  Wire.send(Register_2D);
  Wire.send(8);
  Wire.endTransmission();
}

void loop()
```

```
{
 Wire.beginTransmission(ADXAddress);
 Wire.send(Register_X0);
 Wire.send(Register_X1);
 Wire.endTransmission();
 Wire.requestFrom(ADXAddress,2);
 if(Wire.available()<=2);
 {
    X0 = Wire.receive();
    X1 = Wire.receive();
    X1 = X1<<8;
    X_out = X0+X1;
 }

 Wire.beginTransmission(ADXAddress);
 Wire.send(Register_Y0);
 Wire.send(Register_Y1);
 Wire.endTransmission();
 Wire.requestFrom(ADXAddress,2);
 if(Wire.available()<=2);
 {
    Y0 = Wire.receive();
    Y1 = Wire.receive();
    Y1 = Y1<<8;
    Y_out = Y0+Y1;
 }

 Wire.beginTransmission(ADXAddress);
 Wire.send(Register_Z0);
 Wire.send(Register_Z1);
 Wire.endTransmission();
 Wire.requestFrom(ADXAddress,2);
 if(Wire.available()<=2);
 {
    Z0 = Wire.receive();
    Z1 = Wire.receive();
    Z1 = Z1<<8;
    Z_out = Z0+Z1;
 }

 Xg = X_out/256.00;
 Yg = Y_out/256.00;
 Zg = Z_out/256.00;
 lcd.clear();
 lcd.print("X=");
 lcd.print(Xg);
 lcd.setCursor(8, 0);
 lcd.print("Y=");
 lcd.print(Yg);
 lcd.setCursor(0, 1);
 lcd.print("Z=");
 lcd.print(Zg);
 delay(300);
}
```