VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Phuoc Trinh

CONTROLLER DEVICE FOR EXTENDED REALITY

School of Technology
2021

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author: Trinh Huu Phuoc
Title: Controller Device for Extended Reality
Year: 2021
Language: English
Pages: 38
Name of supervisor: Chao Gao

Extended reality is expected to be widely adopted in the near future, including mixed reality, virtual reality, and augmented reality. New technologies regarding both hardware and software are being developed to meet the expectation, with the objective is to create a truly wearable device that can be comfortably used daily, even in outdoor environment. The development comes with many challenges, one of which is an input solution for seamless interactions.

In this project, with the help of Bao Vu regarding the hardware implementation, the idea of a thumb mounted controller is transformed into a working prototype. The prototype comprises an Adafruit Feather Sense development board and Ohmite FSR05 force sensitive resistors, the software is developed using Arduino for processing input from users and Unity for showcasing the usability of the controller in an augmented reality environment. The prototype controller communicates with an augmented reality device using Bluetooth Low Energy.

Keywords: Augmented reality, Bluetooth Low Energy, Extended Reality, Unity, Virtual Reality

CONTENTS

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
ADV	Advertisement
AR	Augmented Reality
ATT	Attribute
BLE	Bluetooth Low Energy
Bluetooth SIG	Bluetooth Special Interest Group
CCCD	Client Characteristic Configuration Descriptor
DPS	Degree per Second
FSR	Force Sensitive Resistor
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General-purpose Input/Output
HID	Human Interface Device
IMU	Inertial Measurement Unit
ISM	Industrial, Scientific, and Medical Band
MEMS	Micro-electromechanical Systems
MIC	Message Integrity Check
MR	Mixed Reality
PDU	Protocol Data Unit
RX	Receiver
TX	Transmitter
UUID	Universally Unique Identifier
VR	Virtual Reality
XR	Extended Reality

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

Extended Reality (XR) including Virtual reality (VR) and augmented reality (AR) have been in research and development since the late 1970s, but for the consumer market, the technology has only been available for about 10 years. Even so, there are still limitations that prevent us from having a truly wearable AR/VR device for everyday use, one of which is an appropriate input solution. Two main approaches for inputting in AR/VR are physical controllers and hand tracking with the physical controller being the more popular approach. A typical physical controller is usually equipped with a set of buttons and a joystick which can be used for manipulating AR/VR elements; however, it is usually bulky and is likely to cause arm and hand fatigue for long time usage. Hand tracking, on the other hand, requires no external hardware making it a more natural experience but it can only be operated within the tracking range of a tracking device such as a camera or short-range radar, and the accuracy may be a problem in low-light environments. Figure 1 shows the representations of the physical controllers and hand tracking are used for manipulating a menu in a VR environment.

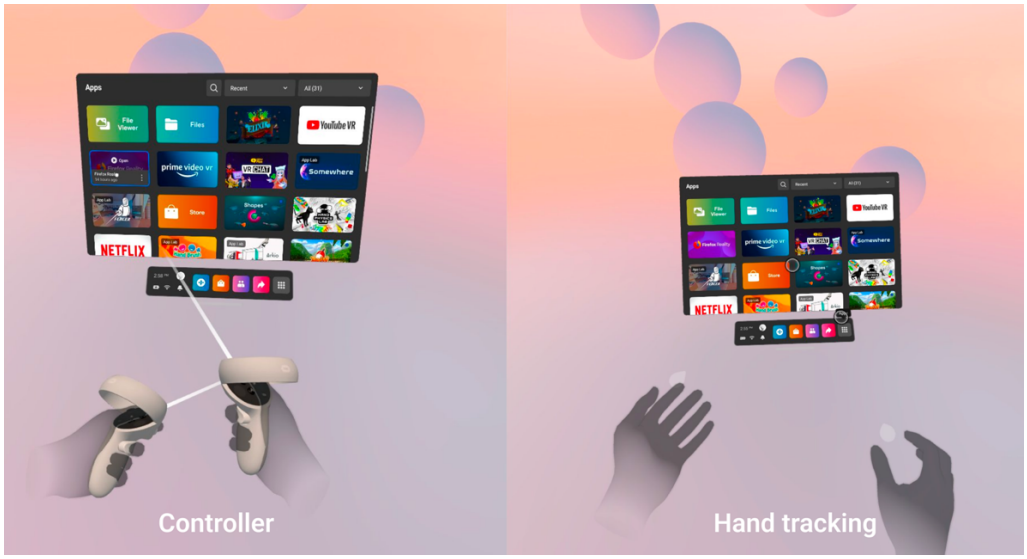

Figure 1. Physical VR controller and hand tracking in a VR environment

As can be seen in figure 1, to manipulate a VR menu using a controller, the user can either use a pointer projected from the controller or the buttons on the controller for navigating and selecting items within the menu. In the case of hand tracking, it requires the user to maintain their hands on the tracking field to be able to move the cursor projected from their hands and perform finger gestures for navigating and selecting. In some usage scenarios, hand tracking can be compromised when a finger is in shadow and cannot be seen by the tracking device, further leading to input errors thus finger gestures are usually limited to index finger pinching onto thumb and it must be performed clearly in front of the tracking device.

1.2 Objectives

The objective of this project is to propose an input device to be worn on a user's thumb which uses thumb movements as input data for controlling in XR, such device can operate independently without being tracked and is less likely to cause arm and hand fatigues, thus can be implemented in the future augmented reality and virtual reality systems. For such an approach, the input device must be capable of detecting thumb gestures and movements and processing them for user-desired input.

1.3 System Structure Overview

Figure 2 illustrates the system comprising an XR controller and an AR device, with the XR controller is based on an Adafruit Feather nRf52840 Sense board which takes the user's thumb gestures as input parameters for controlling an AR application on the AR device, in this project is a tablet. For input processing, a program is developed using Arduino to measure movement and contact data from a built-in inertial measurement unit (IMU) along with two force sensitive resistors connected to the Adafruit board's analog inputs, the measured data is further processed and converted to input states and is transmitted to the AR device over BLE. On the AR device, an AR application is developed using Unity and ARKit, and it takes received input states to manipulate AR elements in the application.

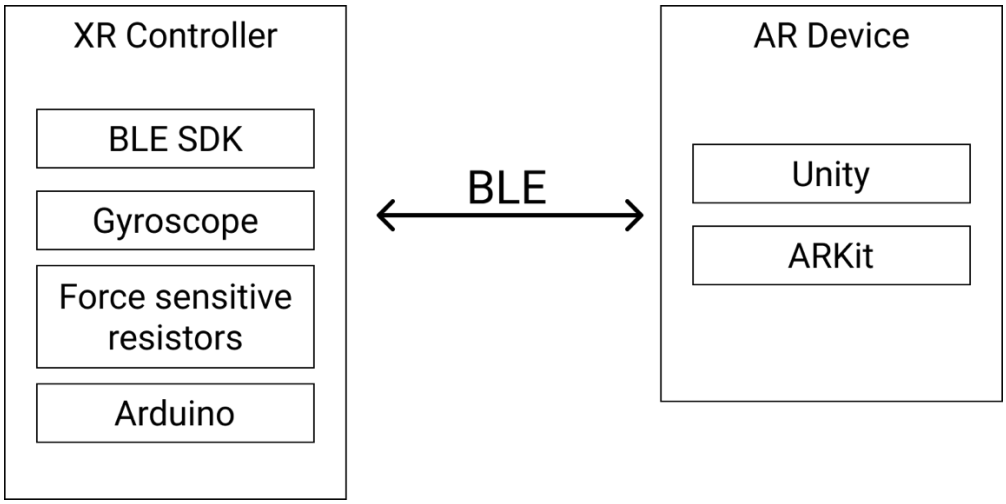

Figure 2. XR controller system diagram

2 THEORY AND BACKGROUND INFORMATION

Figure 3 shows the process diagram of the augmented reality system in this project, the process begins with device initialization including Bluetooth Low Energy (BLE) and sensors setup. Once the device initialization is finished, the XR controller starts advertising its service to other BLE devices around it. On the AR device side, the process begins with application initialization including AR scene generation and BLE setup, then it starts to scan for XR controller around, if the XR controller is found, a connection between the AR device and XR controller is established. After the connection has been made, the XR controller starts to measure and process data from the force resistors sensors and gyroscope for the user's input gestures. The input data is then notified to the AR device every 20ms, which is used for manipulating a menu in AR environment on the AR device.

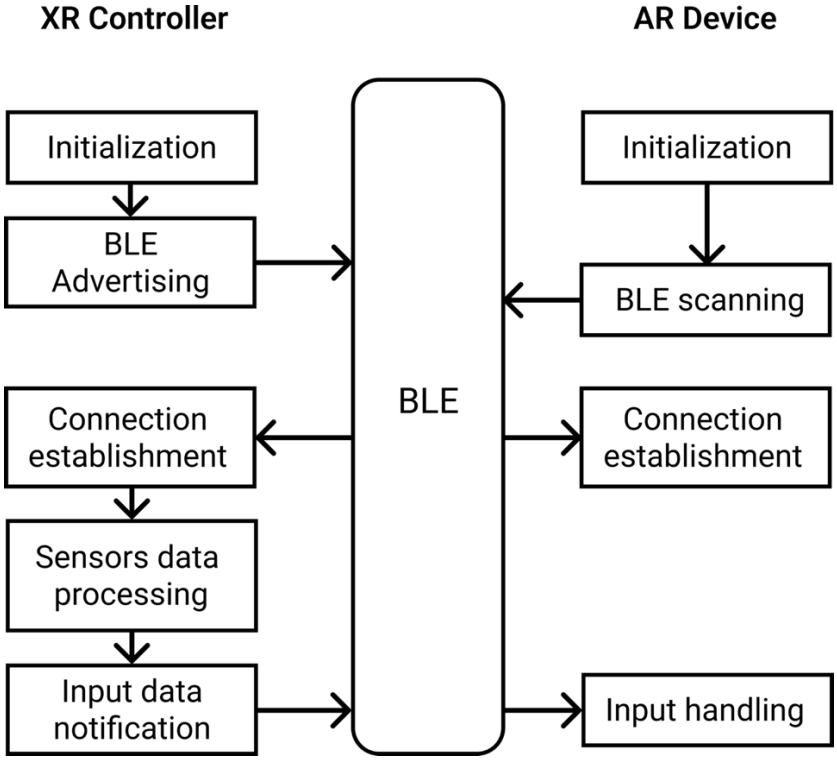

Figure 3. System process diagram

2.1 Augmented Reality

Augmented reality is referred to an experience in which the virtual information and objects are visually overlayed onto the image of the real world therefore users

are not isolated from reality and can still be able to interact with it. In this project, AR is chosen as the main approach for demonstrating the usability of the XR controller as it can be implemented without the need for a dedicated XR headset, instead, a smartphone or tablet can also play the role of an AR device.

Figure 4 shows an AR application is controlled using the controller developed in this project, in this application the AR device is a tablet that uses its camera for tracking the surrounding environment and overlaying AR elements onto it.

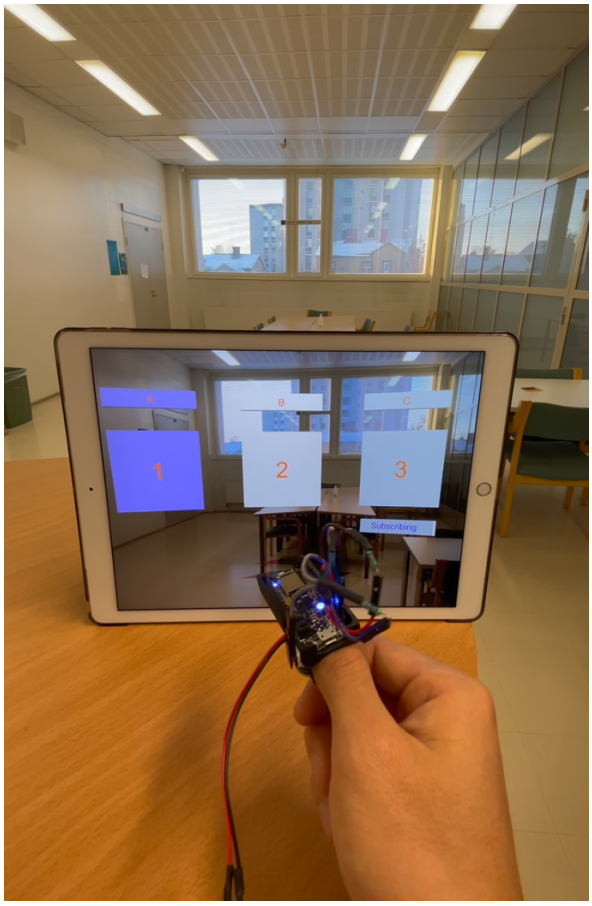

Figure 4. An AR application with the XR Controller

2.2 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a short-range radio communication stack developed by Bluetooth Special Interest Group (Bluetooth SIG) for operating at a very low power rate. BLE uses 40 channels in the 2.4Ghz unlicensed ISM frequency band, in which 37 channels are for data transmitting and 3 channels for advertising /1/. Considering the specifications of a thumb-mounted input device that is most likely

to be powered by a relatively small battery yet requires low latency wireless communication, BLE was chosen as the main wireless communication protocol in this project. Another reason that makes BLE appealing in this project is this communication stack has been widely adopted making it possible to implement in a wide range of mobile devices.

2.2.1 GAP

Generic Access Profile (GAP) is a layer of the BLE protocol stack responsible for BLE advertising and connections. GAP handles the access modes and procedures of a BLE device including device discovery, link establishment, link termination, initiation of security features, and device configuration. /2/

Device roles

GAP defines various roles for BLE devices, in which two key concepts are central devices and peripheral devices.

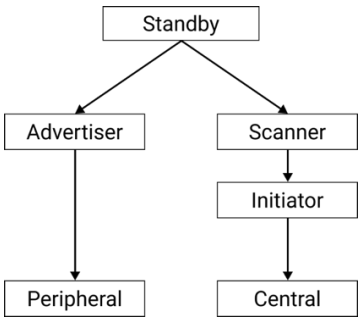

Figure 5. GAP state diagram

Figure 5 /2/ shows a state diagram defined by the GAP layer wherein:

- *Standby*: The device is in the idle state, during this state, there is no data transmission or reception. /2/
- *Advertiser*: the BLE device will transmit an advertising packet including device address and additional data such as device name and available services for letting other initiating devices within transmission range know that it is a connectible device. /2/

- *Scanner*: The device listens for advertising packets from advertisers, when the advertising packet is received the scanning device sends a scan request to the *advertiser*. The *advertiser* responds with a scan response. This process is called device discovery. The scanning device is aware of the advertising device and can initiate a connection with it. /2/
- *Initiator*: A peer device's address to connect is specified so that when the advertising packet is received with the matching address the initiator will send a connection request for establishing a connection with the advertising device. /2/
- *Peripheral/Central*: When a connection is formed, the device functions as a peripheral if the advertiser and a central if the initiator. /2/

Regarding this project, both the XR controller and the AR device begin with standby role before the controller becomes an advertiser and the AR device becomes a scanner. On the AR device side, being the scanner, it scans for nearby XR controller, once an available XR controller matched the name and UUID defined in the AR device program is found, the AR device send a connection request for establishing connection with the controller then becomes a central. Being the advertiser, the XR controller transmit its advertiser packets to nearby BLE device, once it receives a connection request from the AR device it becomes a peripheral.

2.2.2 GATT

While GAP handles the connection establishments between two BLE devices, Generic Attribute Profile (GATT) is responsible for data transceiving between peripheral and central devices using Services and Characteristics concepts. It makes use of a generic data protocol called the Attribute Protocol (ATT**)**, which is used to store Services, Characteristics, and related data in a simple lookup table using 16-bit IDs for each entry in the table. /3/

GATT Transactions

Server/client relationship is an important concept in GATT. In this relationship, a peripheral device plays the role of GATT server which holds the ATT lookup data,

service, and characteristic definitions and GATT Client is a central device that sends a request to the GATT server. All transactions are started by the GATT Client, which receives a response from the GATT Server. Figure 6 /3/ shows a connection diagram between a peripheral and a central device. When establishing a connection, the peripheral will suggest a 'Connection Interval' to the central device, following the suggestion, the central device will try to reconnect every connection interval to see if any new data is available, etc. However, the central device may not be able to reconnect at every connection interval when it's busy talking to another peripheral or lacking system resources. /3/

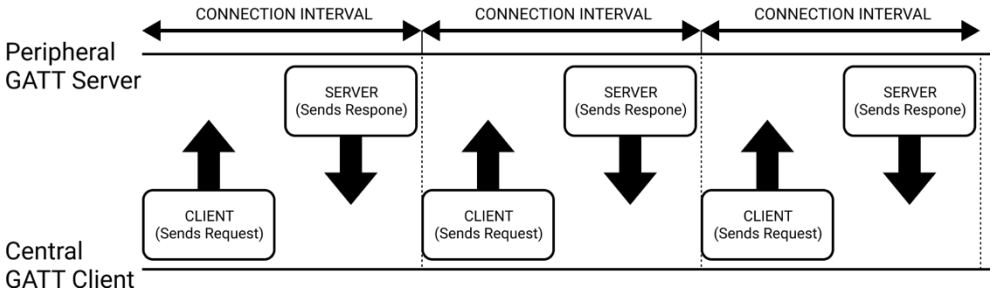

Figure 6. BLE connection diagram

Application throughput

To calculate the application throughput, first the protocol data unit (PDU) need to be clarified, a BLE PDU data packet comprises a header with the size of 2 bytes, a payload in our case is 1 bytes of data for the input state and a message integrity check (MIC) with the size of 4 bytes.

The theoretical application data throughput therefore can be calculated as:

$$Throughput = Data\ packet\ per\ second * Data\ per\ packet$$ /4/

Regarding the BLE communication between the XR controller and the AR device, the minimum connection interval is set to 20ms to comply with “Bluetooth Accessory Design Guidelines for Apple Products” /5/. In an ideal scenario, one packet with data size of 1 byte is transmitted per connection interval, therefore throughput equals 1000 * 7/20 = 350 bytes per second.

Services and Characteristics

GATT transactions in BLE are based on high-level, nested objects called *Profiles*, *Services,* and *Characteristics*, in which:

- *Profile* is a pre-defined collection of Services that has been compiled by either the Bluetooth SIG or by the peripheral designers. For example, Human Interface Device (HID) over GATT profile is the collection of services including HID service, Battery Service, Device Information Service, and Scan parameter service. /3/
- *Service* is a container to hold data as logic entities called characteristics. The service can have one or more characteristics, and each service distinguishes itself from other services using a unique numeric ID called universally unique identifier (UUID), which can be either 16-bit for officially adopted BLE Services or 128-bit for custom services to avoid collisions. /3/
- *Characteristic* is the lowest level concept in GATT transactions, which encapsulates a single data entity. Characteristics contain various parts including a type, a value, some properties, and some permissions. /3/

2.3 Adafruit Feather Sense nRF52840

In this thesis project, an Adafruit Feather nRF52840 Sense board is used, hereby is referred to as Feather Sense, is a BLE development board powered by nRf52840, a multi-protocol chipset from Nordic semiconductor which can be used as both a microcontroller and a Bluetooth Low Energy (BLE) interface. Feather Sense's firmware is based on the standard Nordic UART RX/TX connection profile for 'transparently' transmitting back and forth from BLE devices. This BLE development board also comes with a built-in inertial measurement unit (IMU), STMicroelectronics LSM6DS33, a MEMS-based gyroscope + accelerometer which is used in this project as the main approach for sensing user thumb's movement. To control LSM6DS33, the *Adafruit_LSM6DS33.h* library is used, this library includes custom classes for handling the built-in accelerometer and gyroscope on the board. For the general-purpose input/output (GPIO) handling, the

Feather Sense's firmware includes an adjustable successive-approximation analog-to-digital (ADC) which can be configured to convert data with up to 14-bit resolution (0-1023), and the reference voltage can be adjusted up to 3.6V internally. /6/

2.4 Force Sensors

Force sensitive resistor (FSR) is a type of variable resistor that decreases its electrical resistance in response to physical force or pressure is apply to the force sensing area on it. In this thesis project, two force resistors Ohmite FSR05 were used, according to the manufacturer /7/, the Ohmite FSR05 is capable of sensing a pressure as low as 30g which makes it suitable for sensing the contact when the user applies thumb to a surface. To read the output from the FSRs, a 10-bit analog to digital converter (ADC) is used to convert the output voltage from the analog inputs, in which the FSRs are connected, to the level of force applied.

2.5 Arduino

2.5.1 Development environment

By default, the Feather Sense board can be programmed using Arduino IDE or CircuitPython, regarding this thesis project, Arduino is chosen as the main development environment to upload programs and communicate with the Feather Sense through its native USB port. An Arduino wrapper library developed by Adafruit comes with full control over how the board behaves, including the ability to define and manipulate customized GATT services and characteristics or change the way that the device advertises itself. /6/

2.5.2 Adafruit Bluefruit nRF52 library

Communication over BLE with Nordic Semiconductor nRF52 Series system on chip (SoC) requires the implementation of SoftDevice, a wireless protocol stack built for handling BLE communication. Adafruit nRF52 is a library built based on SoftDevice, which includes custom classes for handling BLE GAP and GATT. Regarding this project, the following classes are used:

- *Bluefruit* is the main entry point to the nRF52 API which exposes essential functions and classes for BLE communications such as peripheral and advertising configurations, connection status, GATT services, and characteristics on the BLE device. /6/
- *BLEService* is a wrapper class for BLE GATT service records and can be used to define custom service definitions or acts as the base class for any service helper class. /6/
- *BLECharacteristic* is a wrapper class for a BLE GATT characteristic record, which can be used to define custom characteristics, or acts as the base class for any characteristic helper classes. /6/

2.6 Unity

Unity is a platform for creating and operating interactive, real-time 3D content with built-in features such as 3D rendering, plane, and collision detection, build tools, etc. With multi-platform support, games and applications developed by Unity can be built to run on different devices. /8/

Figure 7 shows a hierarchy of the Unity application developed in this project consisting of a scene, in which further comprises four gameObjects, a concept in Unity for every object in a game or application. The GameObjects including:

- *AR Session* is a gameObject from AR Foundation framework which controls the lifecycle of an AR experience by enabling or disabling AR on the AR device. /9/
- *AR Session origin* is also a gameObject from AR Foundation framework to transform AR objects into their final pose (position & orientation) and scale in the Unity Scene. This gameObject helps with interaction and manipulation with transformed objects. During the *AR Session,* the AR device provides its data in session space, an unscaled space relative to the beginning of the *AR session*. *AR Session Origin* includes an *AR Camera*

that uses the device's camera and image analysis to track specific points in the world and uses these points to build a map of its environment. /9/

- *Input Handler* is a gameObject includes a script from the plug-in Bluetooth LE for iOS, tvOS and Android and an input processor for handling received inputs.
- *Input Demo* is a gameObject which includes visual AR components arranged as a menu, this gameObject is controlled by the *Input Handler* gameObject.

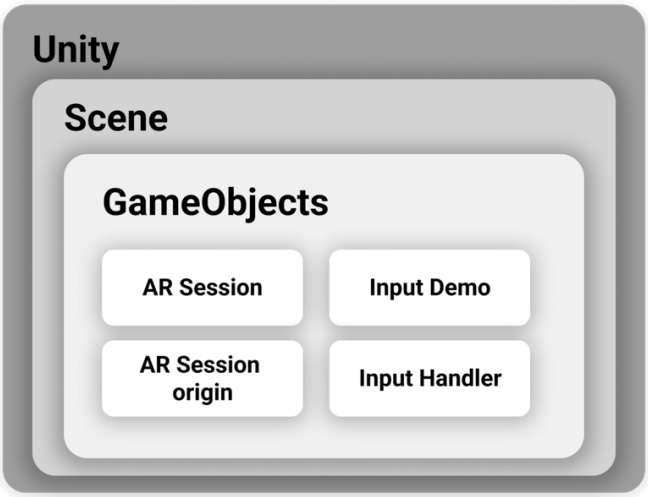

Figure 7. The Unity AR app's structure in this project

2.6.1 Script

An important element in Unity is the script, which is attached to a GameObject and consists of one or more event functions. Event function is a concept wherein Unity passes control, in response to an event in gameplay, to a script intermittently by calling a certain function that is declared inside the script, after the function has finished executing, the control is passed back to Unity. In this project, on the AR device side, a customized script for BLE communication is attached to a GameObject and is used for establishing a connection with the XR controller, this script is also responsible for handling received input states from the controller. /10/

In this thesis project, a customized script is used for handling input from the XR controller over BLE is attached to the Input Handler GameObject. This script is responsible for handling BLE communication including scanning for BLE devices, sending notification requests, and processing received input data from the XR controller for manipulating the Input Demo GameObject.

2.6.2 AR Foundation

AR Foundation is a cross-platform framework for augmented reality development, consisting of features from common software development kit ARkit, ARCore, Hololens and Magic Leap for multi-platform augmented reality development purposes. For iOS AR applications development in Unity requires ARKit XR, a plug-in provided by Apple to enable ARKit support via Unity's multi-platform XR API. The ARKit XR Plugin implements the native iOS endpoints required for building Handheld AR apps using Unity's multi-platform XR API. However, this package doesn't expose any public scripting interface of its own, instead, scripts, prefabs, and assets provided by AR Foundation are used as the basis for AR applications. Regarding this thesis project, customized GameObjects from AR Foundation are used for creating an AR session with environment tracking and overlayed AR elements. /9/

2.6.3 Bluetooth LE for iOS, tvOS and Android

By default, Unity does not support BLE communication for iOS and Android builds, *Bluetooth LE for iOS, tvOS and Android,* hereby is referred to as BLE plug-in, is a plugin developed by Shatalmic, LLC to overcome the issue. The plug-in consists of several parts including Objective-C codes, which communicate with BLE devices from iOS using CoreBluetooth framework, C# scripts which provide Unity-Plugin interface and helper methods to make calls into the Objective-C codes thus communicate with BLE devices, and an editor preprocessor for dealing with BLE usage permission on iOS devices. Regarding this project scope, the *BluetoothHardwareInterface* and *BluetoothDeviceScript* classes from the BLE plug-in are used. The *BluetoothHardwareInterface* is a class that contains static

methods to make calls into the Objective-C code. The *BluetoothDeviceScript* is used to receive messages passed back to Unity from the Objective-C code. /11/

3 IMPLEMENTATION

In this chapter, we talk about the XR controller and how on-board gyroscope and FSR sensors are used, then we will describe how BLE communication between Feather Sense board and the Unity AR application, at the end of this chapter we focus on AR application development in Unity.

3.1 XR Controller device

3.1.1 Development Kit Setup

Figure 8 shows a case has been made for housing the Feather Sense and 2 FSRs, which can be mounted on a user's thumb. The mock-up also shows the placement of two touch zones in which each touch zone comprises one FSR and is connected to the GPIO pin A0 on the Feather Sense for the first touch zone, and A1 for the second touch zone.

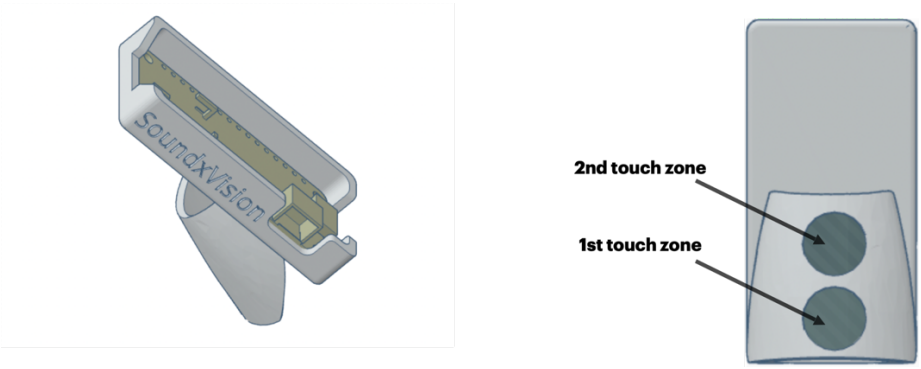

Figure 8. Mock-up of the prototype

3.1.2 Sensors configurations

Contact detection

Regarding this thesis project, two Ohmite FSR05 FSRs are used to measure the applied force on 2 force sensing zones with a voltage divider circuit, comprising two FSRs and 2 pull-up resistors, is implemented as the figure 9 /12/.

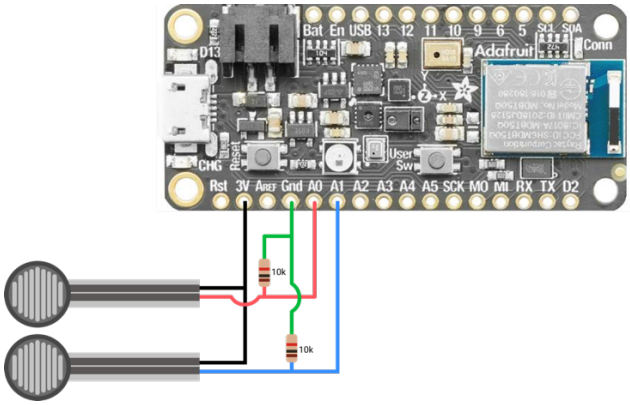

Figure 9. Force sensitive resistors set up

Gyroscope

To scope with fast thumb swiping, the measurement range of angular rate of the LSM6DS33 is set up to 500 degrees per second with the data update frequency of 208Hz.

3.1.3 Sensors data processing

Force levels

As two FSRs are connected to the GPIO pin A0 and A1, their outputs can be read by the ADC input of the microcontroller from the Feather Sense using the code in figure 10. When a force is applied to either the first or second force sensing zone, the function *analogRead()* returns a number between 0 and 1023 that is proportional to the amount of the force applied to that force sensing zone.

```
164    int AIN0 = analogRead(A0); // first force sensor
165    int AIN1 = analogRead(A1); // second force sensor
```

Figure 10. Reading force sensors output

The force applied output is further divided into three levels of touch: *non-contact* with value range 0-50, *normal touch* in 51-600 range, and *pressing* for the value greater than 600 as shown in figure 11.

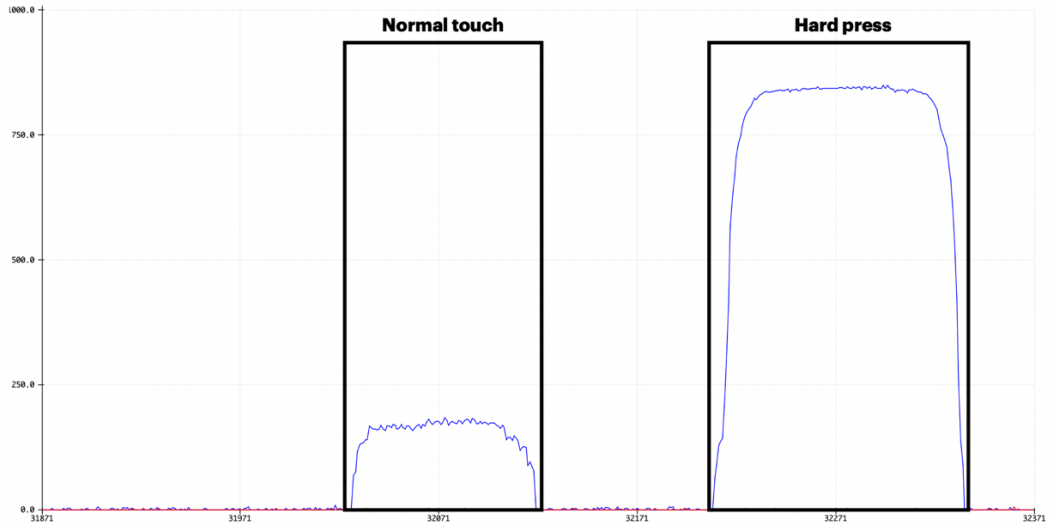

Figure 11. Force levels

Angular velocity

Figure 12 /12/ shows the gyroscope axes of the Feather Sense, regarding the thumb movements, the rotation along the Z-axis (yawn) is used to calculate the angle of thumb rotation.

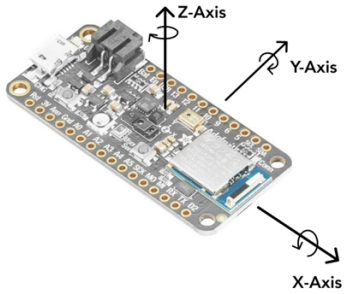

Figure 12. 3D axes of Feather Sense

The gyroscope on LSM6DS33 outputs the rate of change of the angular position over time in radian/s, for calculating convenience the output is converted to degree per millisecond. For rotation of the thumb, angular velocity is integrated as:

$$\theta(t) = \int_0^t \omega(t)dt$$

$\theta(t)$: $rotation\ in\ degree$

$\omega(t)$: angular velocity in degree per millisecond

dt: $time\ difference\ in\ millisecond$

Figure 13 shows the code implemented for horizontal swiping sensing based on z axis angular velocity reading. In this code snippet:

A swiping interval of 1000ms is implemented using the countdown timer *timer1*, during one interval, *zAxisRotation* accumulates the rotation resulted from the integration of angular velocity on z axis *event.gyro.z* (in radians per second, further is converted to degrees per second by multiplying with *SENSORS_RADS_TO_DPS*) over one gyroscope data event. The duration of the gyroscope data event is determined by the subtraction of TimerRef1, which is the timestamp from the previous gyroscope data event and *millis()*, a function that returns the number of milliseconds passed since Arduino board began running. Once the accumulated rotation reached SWIPING_ANGLE constant or its negative value or when the timer1 passed 0, the function moves to the next swiping interval.

```
219   if (timer1 > 0) {
220     zAxisRotation += event.gyro.z * SENSORS_RADS_TO_DPS * (millis() - timerRef1) / 1000;
221     timer1 -= millis() - timerRef1;
222     timerRef1 = millis();
223     if (zAxisRotation >= SWIPING_ANGLE) {
224       state = FIRST_ZONE_LEFT;
225       zAxisRotation = 0;
226     }
227     else if (zAxisRotation <= -SWIPING_ANGLE) {
228       state = FIRST_ZONE_RIGHT;
229       zAxisRotation = 0;
230     }
231   }
232   else {
233     timer1 = SWIPING_UPDATE_INTERVAL;
234     zAxisRotation = 0;
235   }
```

Figure 13. Code snippet for rotation processing

3.1.4 Gestures recognition

Touch states

Figure 14 shows the process of gesture sensing which is further listed as 17 states of input in Figure 15, in which each state holds a value from 0 to 16. Each state can be described as a single keystroke sent to a central device over BLE. The *Idling* state is the starting point of the input processing process in which there is no contact from either of the force sensitive resistors. Once a contact is detected, the program processes the ADC data for identifying on which force sensitive resistor the contact has occurred and the force level of the event. In the case of *Normal Touch,* the program uses the angular velocity data outputted from the gyroscope for calculating traveled angle for detecting swiping horizontal direction. In the case of *Pressing,* the program starts with a timer of 250ms, if the user presses and holds for a duration longer than 250ms the state will be switched to *Press Down*, in case the user stops pressing before the timer reach 0 the state will *be Quick Press*. *Touch release* occurs when the user releases the device off the touching surface, once the program reaches the state of *Touch Release*, a function is called for *up*, *down* swiping detection based on previous and next touch events, in case no touch event occurs after *Touch Release,* the state will be switched back to *Idling*.

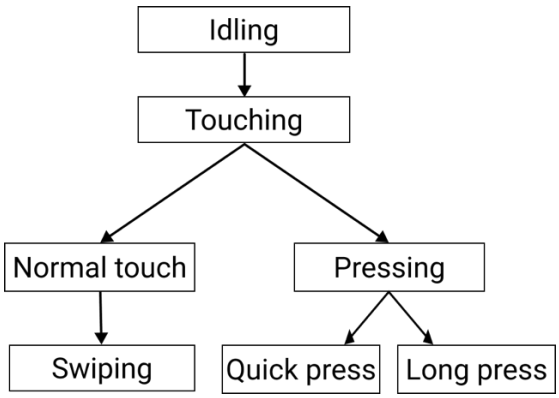

Figure 14. Gesture recognition diagram

```
18  enum TouchState
19  {
20    IDLING,
21    TOUCHING,
22    FIRST_ZONE_TOUCHING,
23    FIRST_ZONE_LEFT,
24    FIRST_ZONE_RIGHT,
25    FIRST_ZONE_START_PRESS,
26    FIRST_ZONE_QUICK_PRESS,
27    FIRST_ZONE_PRESS_DOWN,
28    SECOND_ZONE_TOUCHING,
29    SECOND_ZONE_LEFT,
30    SECOND_ZONE_RIGHT,
31    SECOND_ZONE_START_PRESS,
32    SECOND_ZONE_QUICK_PRESS,
33    SECOND_ZONE_LONG_PRESS_DOWN,
34    TOUCH_RELEASE,
35    UP,
36    DOWN,
37  };
```

Figure 15. Implemented touch states

Swiping left/right

Swiping can be defined as a thumb moving along a finger from the tip to at least the second phalange of the same finger or in the opposite direction, this movement can be measured as angular rotation along the saddle joint of the thumb. For demonstration purposes, the rotation angle measured as shown in figure 16 from my hand with the value of 16° is used, this value may vary on different users' hands. Figure 17 illustrates the detailed process of thumb rotation sensing, by maintaining contact to either of the FSRs with normal touch level of force will start the swiping interval with the duration of 1000ms, during an interval once the calculated rotation reaches 16° or -16° the state will be set to the left or right accordingly. After the state is switched to either left or right, a packet is sent to the central to inform about the swiping interaction, then the state is set back to *Touching* for further gestures sensing.

Figure 16. Thumb movement along the index finger

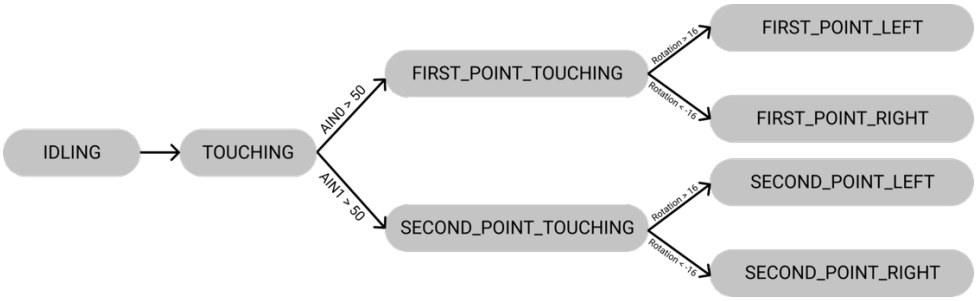

Figure 17. Swiping left/right process diagram

Swiping up/down

Figure 18 illustrates the output of two FSRs as a user swiping a finger on the sensing zones in down and up directions, as can be seen in figure 18 the force applied moves from one FSR to another depending on which direction the user swiped.

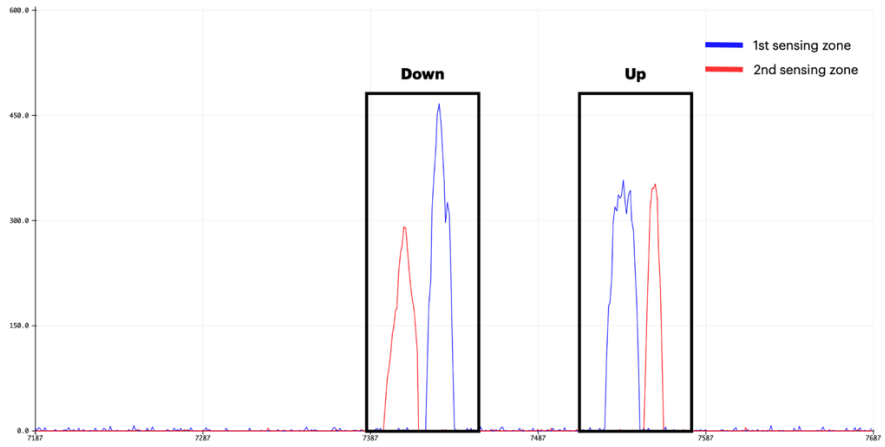

Figure 18. Level of force applied output for swiping down and up

Based on the output of the FSRs, figure 19 illustrates the diagram of swiping up/down sensing implementation, which is different from swiping left/right as it occurs on more than one FSR, once the force applied drops to below normal touch level on one FSR, the state is set to Touch Release which a countdown timer of 100ms is activated, during this duration if the force level on the other FSR raises to normal touch level the state is set to up/down depends on the previous state before touch release, in the case of *FIRST_POINT_TOUCHING*, the state will be set to *DOWN* and *UP* if the previous state before *TOUCH_RELEASE* is *SECOND_POINT_TOUCHING*.

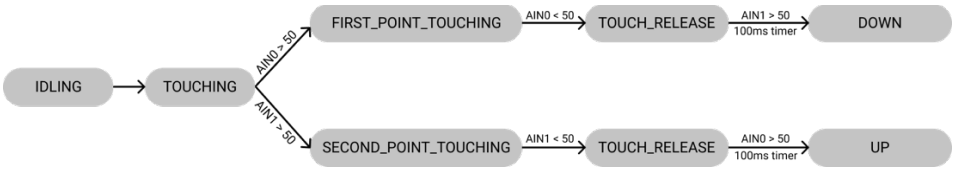

Figure 19. Process diagram for swiping up and down

Pressing

Figure 20 shows the output from an FSR when a user performs pressing gestures. Pressing can be defined as the user applying a force greater than a normal touch

and is defined in the program, based on the duration of a press event, it can further be divided into 2 types of press: *Quick Press* and *Long Press*. The difference between a *Quick Press* and a *Long Press* as can be seen in figure 20 is mostly in their durations, with the event of quick press occurring in a shorter duration.

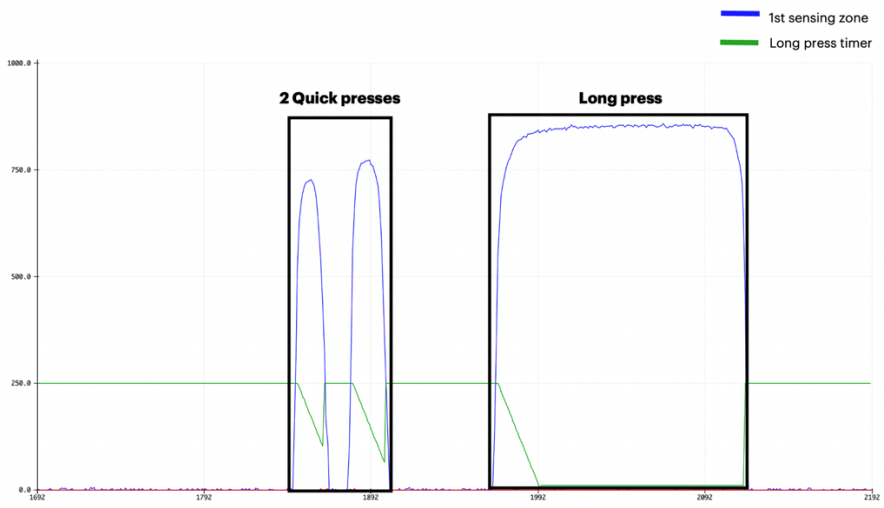

Figure 20. Quick press and long press from an FSR's output

Figure 21 shows a process diagram of *Long Press* and *Quick Press* sensing, the process begins when the force applied on either FSR is greater than pressing the level defined earlier, the state is set to *FIRST_POINT_START_PRESS* or *SECOND_POINT_START_PRESS* depending on which FSR the event has occurred, at either of these state a countdown timer of 250ms is activated, during this duration if the force applied is maintained at pressing level passes the zero point of the timer the state is switched to *Long Press*, otherwise the state is switched to *Quick Press*.

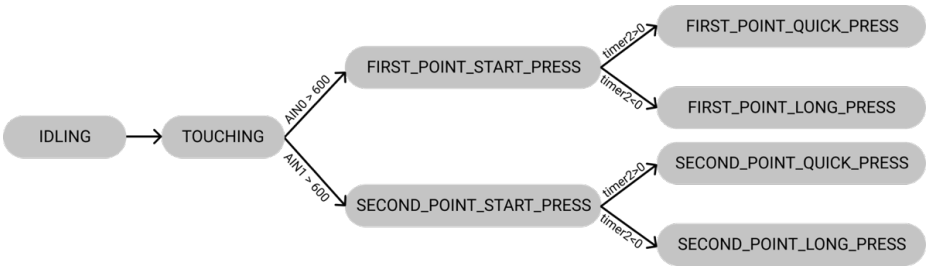

Figure 21. Process diagram for *Quick Press* and *Long Press* sensing

3.1.5 BLE Configurations

Bluefruit.h library is imported to the Arduino sketch for BLE configurations, it includes Bluefruit class with peripheral profile and methods for setting up connection and advertisement. For Apple devices compatibility, the connection interval is set to the minimum of 20ms and a maximum of 30ms to scope with Apple BLE requirement /5/, in which a peripheral must have a minimum connection interval of 20ms. Figure 22 shows the BLE profile structure on the XR controller, it includes an input service that further comprises a characteristic storing the input state data. Once the XR controller received a notification request from a central device, the XR controller will start notifying its value continuously, in this case, the characteristic holds the input state as one byte of data.

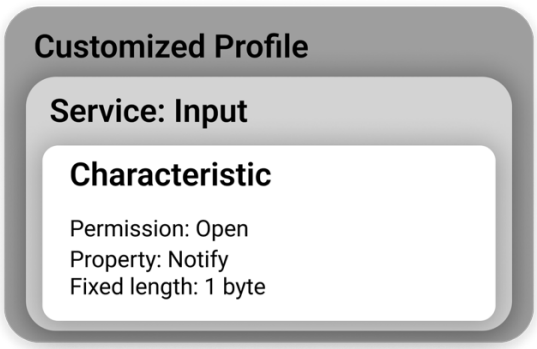

Figure 22. BLE Profile overview

3.1.6 BLE Advertising

For the XR Controller to be discoverable to the AR device, we need to configure BLE advertisement on the Feather Sense. During the advertisement process, advertising packets are sent as a fixed interval, also known as advertising interval, in our case the controller is set to advertise at 20ms minimum interval as recommended in Apple's general advertising guidelines /5/. The advertising packet also includes a service UUID for further scanning on the AR device.

3.1.7 BLE Communication

LightBlue is used for testing the BLE communication from the XR controller, LightBlue is an application for BLE development with a BLE profile that can be set to either peripheral or central, for connection testing with the XR controller, the central profile is enabled. Figure 23 shows LightBlue is used for scanning nearby peripherals, in which the second result is the XR controller with the name "XRController".

Figure 23. Peripheral list in LightBlue

By choosing the XR controller from the list in figure 23, a connection (link) is established between LightBlue and the XR controller. Once a link is established, LightBlue can send a notify request for a characteristic to subscribe to value change notifications. Once the notification for the characteristic is enabled, a stream of input data will be sent to the central device, which is displayed on LightBlue as HEX values.

3.2 Augmented reality device

In Unity, an AR application is developed to use with the XR controller, this application is built using AR Foundation's gameObjects for capturing environment, adjusting, and tracking the position of AR components on captured image.

3.2.1 Input Demonstration

Figure 24 show the AR menu in Unity for input testing, the menu includes a top bar that is controlled by swiping left/right on the first sensing zone and a grid of cells below it to be controlled by swiping up/down/left/right and pressing on the

second force sensing zone. This menu is controlled by the *Input Demo* gameObject which comprise a script which takes the input state received from the XR controller as a parameter for further handling swiping, pressing on the menu in AR.

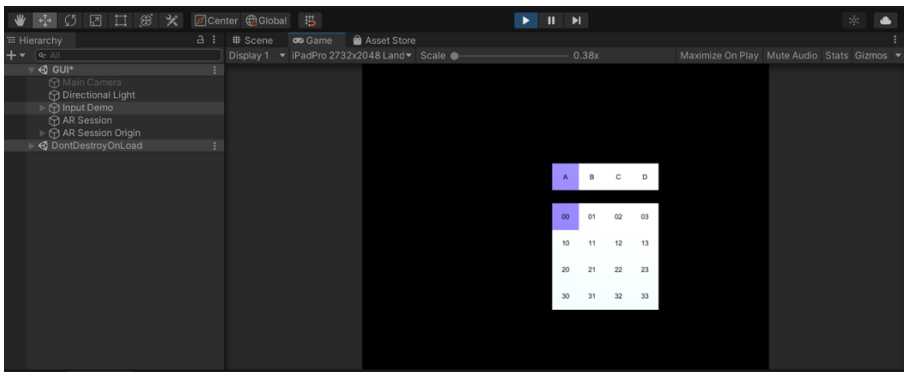

Figure 24. Gameplay during runtime

3.2.2 BLE Communications

Figure 25 is a diagram for handling BLE communication on the AR device side, which is executed one per frame during the AR session.

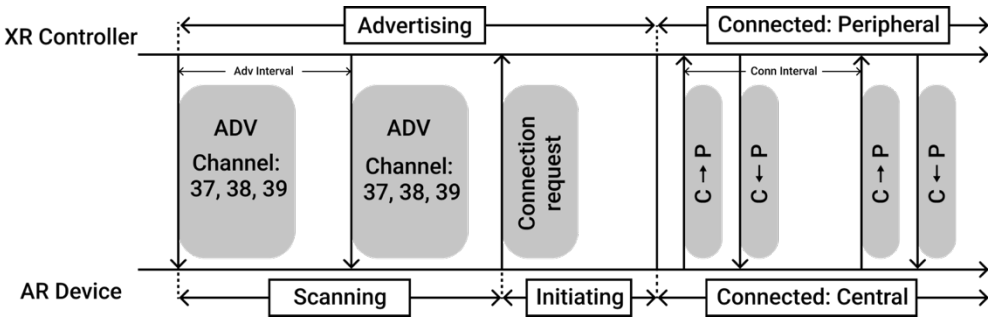

Figure 25. Process diagram for BLE handling

Figure 25/13/ illustrates the procedure for BLE communication in which the AR device establish a connection to the XR controller. On the XR controller, it advertises over BLE advertising channel 37, 38, 39 at 20ms interval with advertising time of 30 seconds. To distinguish from other BLE advertiser, the advertising packet of the XR controller includes an UUID for the input service, this service UUID will be used later by the AR device when scanning for the XR controller. On the AR device side, a script for BLE handling is attached on the *Input*

Handler gameObject, this script defined the XR controller name and its service UUID for further scanning. To scan for the XR controller, the AR device listen to the BLE advertising channels 37, 38, 39 for peripheral with defined name and service UUID, this is done by the *ScanForPeripheralsWithServices*() method from the class *BluetoothLEHardwareInterface* which takes service list with services' UUIDs and return name and physical address of the discovered peripherals. Once the matched XR controller is found, the AR device sends a connection request to the XR controller for establishing the link, the method *ConnectToPeripheral()* uses the device address found earlier for establishing a connection (link) to the matching peripheral. Once the connection is established, the AR device sends a notification request to notify value change on the input characteristic UUID using the method *SubscribeToService()*. The XR controller then starts to notify the states change in response to user input at the connection interval of 20ms, subscribing to this data the AR device makes visual changes on the *Input Demo* gameObject based on the user input. If there's no notification received on the AR device within 5 seconds, the method *DisconnectPeripheral()* is called to terminate the BLE connection, after the BLE link has been terminated and the AR device will start the scanning phase again.

3.2.3 Build for iOS

Unity converts the game/application to an XCode project which is written in Objective-C. Once the XCode project is created, the AR application can be ported to the iOS device or an iOS simulator using the build feature on XCode.

3.3 Evaluation

After the AR application was successfully installed on an iPad, a test for gesture sensing and BLE communication was conducted. For the testing purpose, instead of using a micro-USB port on the Feather sense for powering, the controller is powered by an external lithium-polymer battery pack as can be seen in figure 26.

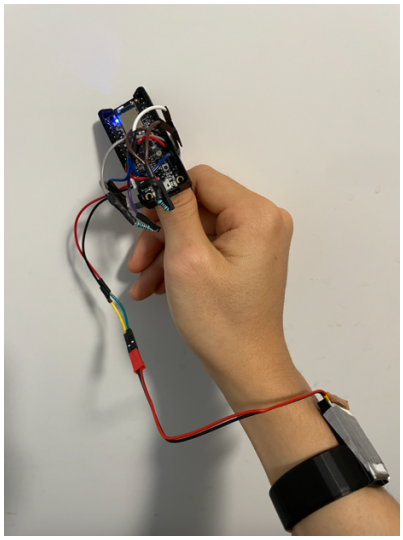

Figure 26. The XR controller mounted on a thumb

The testing includes running the AR application on an iPad and swiping vertically and horizontally on the force sensing zones, pressing with multiple quick presses and long presses on the controller. Figure 27 shows the applications running in an AR environment with the controller connected.

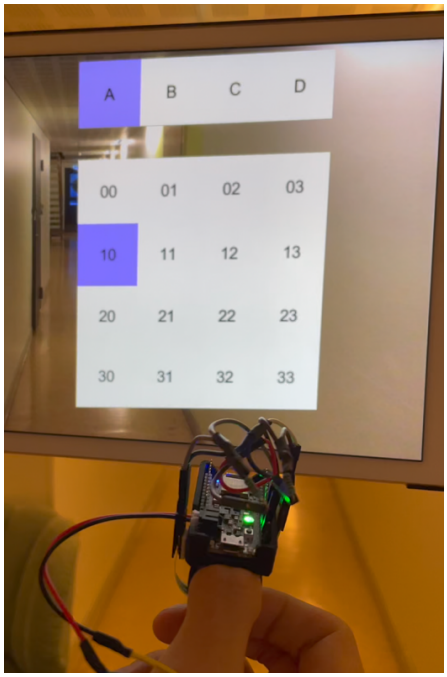

Figure 27. Working demo

The connection between the controller and the iPad has been made successfully, the controller was able to manipulate the *Input Demo* GameObject in the application.

4 CONCLUSION AND FUTURE WORKS

4.1 Conclusion

Regarding the thesis scope, an input solution for controlling in AR/VR using thumb movements as an input parameter was proposed. To demonstrate the usability and performance of the proposed solution, a prototype based on the BLE development board Adafruit Feather nRF52840 Sense and two AR applications were built. The prototype detects user inputs by processing thumb movements data from an external array of force sensing resistors and a built-in gyroscope. The user input is then sent from the prototype to an iPad with two AR applications installed over BLE. From the testing, the user was able to navigate within the AR applications in response to the thumb input gestures.

In conclusion, the thesis project has shown that a thumb mounted device can work as an input device in AR/VR environments, with the relatively small form factor yet responsive, it can be considered for future AR/VR systems.

4.2 Future Works

From my own testing, the gestures of pressing, swiping to left and to right are more responsive than swiping up/down, this could be caused by the gap between the two FSRs, to overcome this problem more FSRs with smaller sizes could be used.

User comfort can be improved with a smaller form factor prototype, for example with an external IMU circuit mounted on user's thumb connected to the development board on user's wrist will help reduce the size of the device on the user's thumb significantly.

For better usability, the XR controller can be used with a hand tracking system for tracking the position and orientation of the XR controller thus providing more control in the AR/VR environment.

REFERENCES

/1/ Bluetooth SIG. Bluetooth Technology Overview. Accessed 20 December 2021. https:www.bluetooth.com/learn-about-bluetooth/tech-overview/

/2/ Texas Instruments. Generic Access Profile (GAP). Accessed 20 December 2021. https://software-dl.ti.com/simplelink/esd/simplelink_cc2640r2_sdk/3.20.00.21/exports/docs/blestack/ble_user_guide/html/ble-stack-3.x/gap.html

/3/ Kevin, T. 2014. Introduction to Bluetooth Low Energy: GATT. Accessed 21 December 2021. https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt

/4/ Mohammad, A. Bluetooth 5 speed: How to achieve maximum throughput for your BLE application. Accessed 21 January 2022. https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/

/5/ Apple. Using the correct Bluetooth LE Advertising and Connection Parameter for a stable connection. Accessed 3 January 2022. https://developer.apple.com/library/archive/qa/qa1931/_index.html

/6/ Adafruit. Feather nRF52840 Sense.Adafruit Feather nRF52840 Sense. Accessed 28 December 2021. https://learn.adafruit.com/adafruit-feather-sense/

/7/ Ohmite. Ohmite FSR series. Accessed 28 December 2021. https://www.ohmite.com/assets/docs/res_fsr.pdf

/8/ Unity. Homepage. Accessed 28 December 2021. https://unity.com

/9/ Unity. UnityEngine.XR.ARFoundation. Accessed 7 January 2022. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.1/api/UnityEngine.XR.ARFoundation.html

/10/ Unity. Unity Manual, Scripting. Accessed 3 January 2022. https://docs.unity3d.com/Manual/ScriptingSection.html

/11/ Shatalmic, LLC. Unity Bluetooth LE Plugin for iOS. Plugin Overview, 5-5. Accessed 4 January 2022.

/12/ Adafruit. Feather nRF52840 Sense. Accessed 28 December 2021. https://www.adafruit.com/product/4516

/13/ Sarkar, Sopan & Liu, Jianqing & Jovanov, Emil. (2019). A Robust Algorithm for Sniffing BLE Long-Lived Connections in Real-time.

Accessed 23 Feburary 2022.
https://www.researchgate.net/publication/334783317_A_Robust_Algorithm_for_Sniffing_BLE_Long-Lived_Connections_in_Real-time