Ashton Billings, Hamza Lutfi, and Aditya Nebhrajani

ECE 445 Proposal

Spring 2025

TA: Jason Zhang

February 14, 2025

1 Introduction

1.1 Problem

Despite the recent breakthroughs in VR technologies and experiences, it's clear that the technology is still in its infancy. Our project will explore one area lacking in the industry, good and relatively cheap hand tracking. As of now, most flagship VR devices still rely on handheld controllers, with only the more expensive products such as the Valve Index having hand-tracking capabilities.

1.2 Solution

We plan to develop a relatively cheap and accurate hand-tracking system. To confirm the accuracy of our system, we need something to control with this hand tracker. We have opted to control a virtual hand using the Unity game engine OpenXR SDK. Our project can be broken up into three subsystems: hardware, firmware, and software. The hardware will be a PCB board attached to a glove, and sensors will be arranged in such a way as to retrieve finger location data. The exact sensor we use is something we will explore, more on this in the hardware section. The firmware will take in the data from these sensors, process it, and send it to our computer. Exactly what will be processed in the glove vs on the computer will be determined later. Finally, Unity will take in this data, and use it to control a virtual hand in the game engine.

1.3 Visual Aid

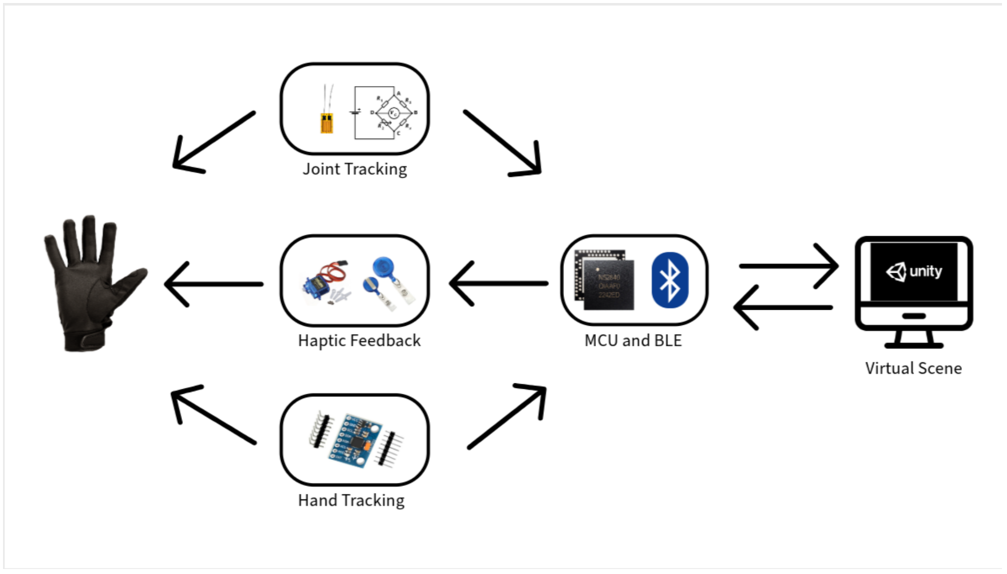

Figure 1: Visual-Aid

1.4 High-level Requirements

1. We can pick up virtual objects with our gloves, with them properly locking in place for haptic feedback.

2. The gloves have a latency of under 1 second. We can test this by moving the glove in a predictable way, and timing how long it takes for Unity to react. We can do this by having a timer in our firmware and software and comparing the two.

3. The accuracy of the gloves is within a reasonable range. We can test this by predictably moving the glove into a certain position, say bending the index finger in relative to the palm in by 30 degrees, and seeing how much Unity moves the virtual hand in response. I'd say a reasonable range is that the fingers track within +/-15 degrees.

2 Design

2.1 Block Diagram

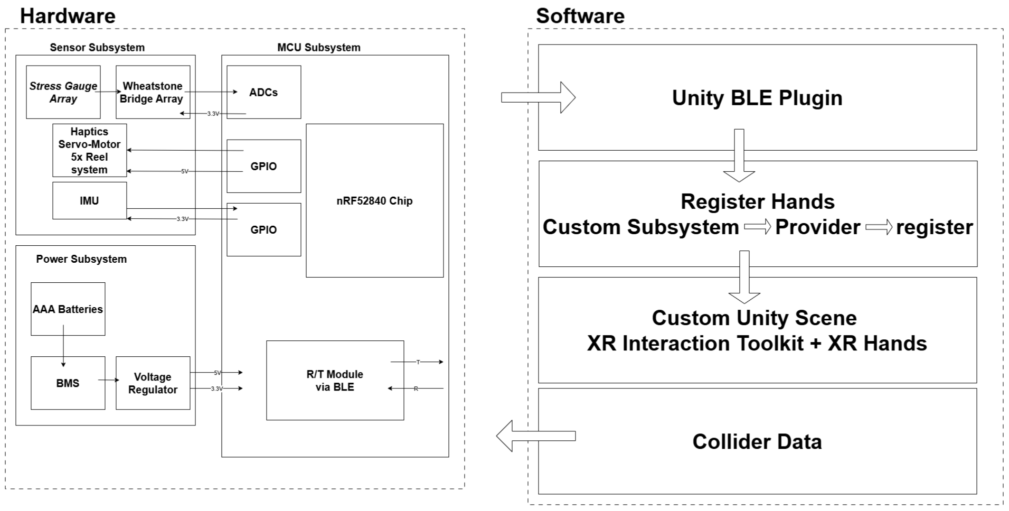

Figure 2: Visual-Aid

2.2 Subsystem Overview

Here we split our system into two subsystems, hardware and software. We further break down our hardware subsystem into the following blocks: sensors subsystem, power susbstem, and the MCU subsystem.

Power Subsystem - Since we are using BLE, our desire here is that our gloves work wirelessly. As such, we need to battery power these gloves, hence the power subsystem. This system will take in AAA batteries, and using a BMS + voltage regulator create steady 3.3V and 5V lines to power all of our systems.

Sensor Subsystem - Our sensor subsystem has three parts: stress gauge sensor array + Wheatstone bridge array for joint tracking, a servo motor + reel system for hand tracking, and an IMU system for general hand tracking. By connecting our stress gauge sensors to a Wheatstone bridge, we can turn the resistance changes of this sensor due to it stretching to voltages that the ADC on our MCU can read in. The servo motor + reel system is what will be used for haptics. In Unity, when two objects with colliders are colliding, you can read in this data. We will use this to our advantage to have haptic feedback by stopping our fingers in place as they collide with objects by having our MCU take in this data, and communicate it with our haptics system via GPIO pins. Our haptics system consists of a servo motor + reel system. We will be taking retractable badge reels and attaching these reels to each finger. Once an object collides with our finger, a servo motor mounted on our glove will hold the reel in place, stopping our fingers.

IMU Subsystem - Finally, we have our IMU system. Although our joints are being tracked, our device needs a set reference point for where our hands are. We will have a IMU mounted in the glove to give a reference of where our hand is, using sensor fusion algorithms to get an accurate position. Our IMU will communicate with our MCU via its GPIO pins.

MCU Subsystem - The jobs of our MCU subsystem will be taking in data from our sensors and sending them via BLE to our software, as well as taking in data from the software to control the haptics. It will also be responsible for computing the IMU sensor fusion algorithm for general hand tracking.

Software Subsystem - Our software will take in this sent data and convert it to work within the XRHands Unity package. This is done by creating a custom XRHandSubsystem provider, which takes in the raw sensor data and formats it in such a way as that the XRHands unity package can be used. With this package, we can control virtual hands in a virtual Unity scene which can be deployed to a VR headset such as the Oculus Quest 2. This scene will contact virtual objects we can interact with to test our gloves.

2.3 Subsystem Requirements

Power Subsystem - Our power system needs to be onboard our glove system as to initiate the wireless nature of our gloves. It needs to provide consistent and steady power as not only will it be powering our MCU, but it will be providing the power to our strain sensors. If there is noise on the power line, this will cause noise on our sensors.

Sensor Subsystem - Our sensors also need to be properly mounted onto our gloves in such a way as to allow for freeform hand movement. The strain sensors need to be configured with a wheatstone bridge as to get valid voltage data from them. This means picking resistor values near our strain sensor's nominal resistor value. Given the fact we are using a reel + motor system for haptics that will attach these reels to our fingers, we need to make sure it is arranged in such a way as to not get in the way of our strain sensor joint system.

IMU Subsystem - Our IMU subsystem needs to be compact enough to fit within the rest of the subsystems as well as be far enough away from EMI sources since we will be using sensor fusion algorithms utilizing things like the magnetometer data.

MCU Subsystem - Our MCU subsystem needs to read our data, conjoin it, and send it to our PC fast enough to not experience latency issues.

Software Subsystem - Our software subsystem needs to be able to take in this data and convert it to usable XRHands data as to not experience latency issues. The software will also need to properly deploy to a VR headset for testing.

2.4 Tolerance Analysis

Power Subsystem - This power subsystem needs to keep a steady voltage as it will not only provide power to most other subsystems, but things like the strain sensor's noise will directly be effected by the noise on the power line. It will experience noise due to things like the haptics system activating, as this will draw current quickly from the power line due to the motors becoming active, so decoupling capacitors will need to be heavily used.

Sensor Subsystem - The strain sensors we will be using are typically used on hard materials like metals and concrete, so there use here in fabric will need to be carefully tested. If a steady enough voltage from flexing fabric cannot be used with these sensors, a different finger sensing method may need to be used instead. As for the haptics system, this will most likely take up the most room on the glove, so getting it all to work and fit within each other will need to be properly considered.

IMU Subsystem - IMUs are prone to be inaccurate, hence why we plan to use sensor fusions algorithms here to accomplish this. These algorithms rely on the other sensor data available to us in IMUs such as the magnetometer and gyroscope. If there is enough EMI and noise on the system, the sensor fusion algorithms we would reasonably be able to implement may not be sufficient, and other tracking methods may need to be employed.

MCU Subsystem - Our MCU data needs to read in all of our sensor data, package it up, and send it our PC. It also needs to receive collider data from the PC and control the motors from this. Finally, it will also need to run our sensor fusion algorithm in order to get position data. It being able to do all three of these in low enough latency will need to be properly considered.

Software Subsystem - Our software needs to properly be able to deploy to a VR headset and take in this sensor data and use it in low enough latency. It also needs to contain enough interactables as to test our gloves.