WENTWORTH INSTITUTE OF TECHNOLOGY

ELECTROMECHANICAL ENGINEERING

FINAL REPORT

# Kinesthetic feedback device for realizing boundary contact through a virtual environment

*Authors:*
Andrew ZUCKER[1*]
Owen MEDEIROS[2*]
Jared TALAMINI[3]

*Supervisor:*
Dr. James MCCUSKER

April 24, 2019

[1]azucker55@gmail.com [2]medeirosowen@gmail.com [3]jaredtalamini@gmail.com
*These authors contributed equally to this work.

**Abstract**

Virtual reality is growing rapidly and has demonstrated uses in the fields of rehabilitation, professional training, education, research, and recreation, thanks to its immersive quality. Virtual reality is able to stimulate audio and visual senses, but touch remains largely underdeveloped. We present a bi-directional slot mechanism that actively limits the range of motion (ROM) of a user's lower arm, but does not affect motion within these limits. By controlling these limits, the device provides kinesthetic feedback to simulate virtual interactions without additional impedance to the user's movement. These interactions are fostered using HTC VIVE tracking components and BLDC actuators. In developing such a device, we have laid the framework for other single degree of freedom joints and provided initial steps towards the design of a fully immersive experience.

# Contents

# 1 Problem Definition

Since its inception in 1968, virtual reality (VR) has demonstrated an ability to surround the user with virtual information [1]. Virtual settings provide great opportunity in the fields of rehabilitation [2, 3], professional training [4, 5], recreation [6], and education [7]. The quality of immersion directly relates to the realism of the application [5]. Immersive virtual settings have been achieved by stimulating the user's audio and visual senses, but touch has been largely underdeveloped. Tactile devices that simulate textures, vibrations, and contact interactions provide a sense of immersion [8–13]. However, devices that enable large scale interactions have not been achieved. By developing devices that enable large scale interaction, the next steps in achieving a high-fidelity immersive experience can be realized.

# 2 Background research

In the pursuit of developing a high-fidelity immersive experience, initial research was conducted into physical human robot interaction (pHRI). A review by Losey *et al* discusses the advances of pHRI and its development in areas like telerobotics, exoskeletons, and robotic limbs for amputees [14]. Here, the intent of these devices is to improve the user's strength and endurance and such designs are directly coupled to the joints of a user. In coupling to a joint, external control or detection of the user's intent is necessary for actuating a limb, and this disqualifies the devices as immersive. This inability to move the joint without applying a force to the control device is a term we define as *position lock*; it is the main obstacle for a high-fidelity immersive device. To create such a device, experience and background knowledge is needed in a wide breadth of technical areas. In this section, the basic understanding needed for the development of an immersive device is outlined.

## 2.1 Kinematics

To avoid *position lock*, an understanding of human kinematic modeling is required. The body can be modeled using single, two, or three degree of freedom joints. Modeling the motion of these joints requires mathematical understanding of spatial translations and rotations. Spatial translations are straight forward compared to spatial rotations which can be accomplished using two methods.

### 2.1.1 Euler Angles

In the first method, axial rotations are defined by Euler angles offset from a global coordinate system [15]. This method is limited by *gimbal lock*, a scenario where the rotation of two axes are aligned and a degree of freedom is compromised. A graphical representation of gimbal lock is shown in Figure 1.

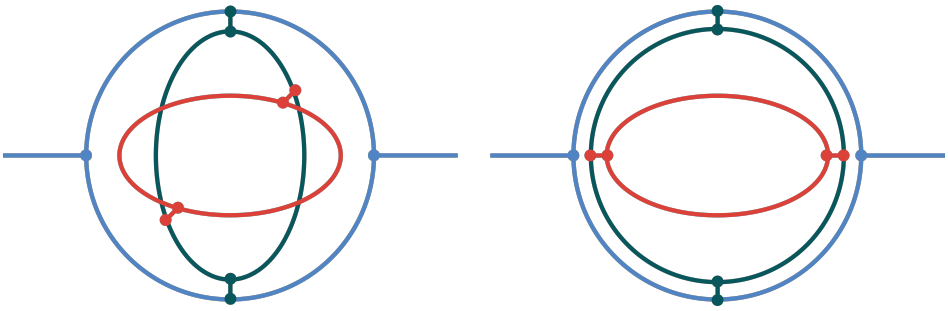


Figure 1: Gimbal schematic representing free rotation and gimbal lock. Left: Three freely rotating axes within Euclidean space. Right: Two axes aligned, reducing rotational degrees of freedom, defined as gimbal lock.

### 2.1.2 Rotation Matrices

In the second method, linear algebra is utilized to perform rotations in Euclidean space. A general rotation matrix is defined as orthagonal with determinant 1. To perform a basic rotation about the $x$, $y$, or $z$ axis, the following matrices can be applied to a coordinate system.

$$\mathbf{R_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{1}$$

$$\mathbf{R_y}(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{2}$$

$$\mathbf{R_z}(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Performing multiple rotations is as simple as multiplying more matrices, resulting in a combined matrix. Although, rotation matrix multiplication is non-commutative, and matrices must be multiplied in the correct order to achieve the desired rotation.

### 2.1.3 Quaternions

In the third method, quaternion rotation defines a spatial rotation by a vector and a rotation about that vector. Quaternions extend the complex number system to three dimensions and include a rotation scalar, forming a four dimensional vector [16]. Operating in four dimensions allows for three dimensional rotations to be performed without gimbal lock. The definition of a quaternion can be seen in Equation 4 where $\theta$ is the angle of rotation and $u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$ is the unit rotation vector $\mathbf{v}$.

$$\mathbf{q} = e^{\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} \tag{4}$$

$$= \cos\frac{\theta}{2} + (u_x\,\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2} \tag{5}$$

$$= w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{6}$$

In order to multiply two quaternions together, it must be noted that the method is noncommutative. Using quaternion mathematics, an indefinite series of rotations can be converted into a single rotation about an axis. To perform a rotation on a vector $\mathbf{p}$, one must consider the vector $\mathbf{p}$ to be a quaternion with a rotation component equal to zero. Let $\mathbf{p}$ be an arbitrary vector that is to be later rotated by $\mathbf{q}$:

$$\mathbf{p} = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

The rotation of $\mathbf{p}$ about $\mathbf{q}$ is achieved by *conjugating* $\mathbf{p}$ by $\mathbf{q}$, by using the Hamilton product.

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \tag{7}$$

This rotation method can be applied to multiple degrees using $\mathbf{p}'$ as the new vector to be rotated by another quaternion.

## 2.2 Virtual Tracking

Accurately and precisely tracking an object's movement is the foundation of interacting with a virtual environment. To understand how an audio-visual platform determines the relative position of the user, we focused on the theory employed by the HTC VIVE. The VIVE components rely on infrared (IR) photo-diodes and base stations. The base stations produce IR laser plane sweeps that cause the photo-diodes to assert their output low, shown in Figure 2. The timing of the output pulse train encodes the position of the sensor relative to the base stations [17].

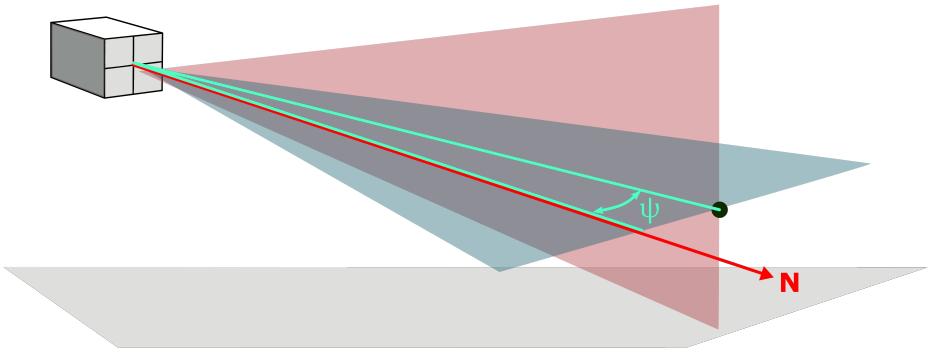



Figure 2: Depiction of HTC VIVE IR light laser plane sweeps. One of two base stations are shown in the upper left corner and the black circle represents the photo-diode sensor. The vertical and horizontal plane sweeps are shown in transparent red and blue, respectively. The vector **N** is normal to the face of the base station and the azimuthal offset $\psi$ is shown while the polar offset $\phi$ is zero.

The timing of the sweep pulse indicates the angular offset from the base stations' normal vector, $\mathbf{N}$, which is determined from the room setup. The angular offset is calculated from the difference pulse time, $t$ and the center of the sweep $4000\mu s$, divided by the total sweep time.

$$\theta = \frac{\pi(t - 4000)}{8333} \tag{8}$$

The angular offsets are referred to as the *polar* ($\phi$) and *azimuthal* ($\psi$) angles. With these angles, a local unit vector corresponding to the location of the sensor can be produced.

$$\hat{i} = \sin\phi\cos\psi \tag{9}$$

$$\hat{j} = \cos\phi \tag{10}$$

$$\hat{k} = \sin\phi\sin\psi \tag{11}$$

$$\mathbf{P}_l = \langle \hat{i_1}, \hat{j_1}, \hat{k_1} \rangle \qquad\qquad \mathbf{Q}_l = \langle \hat{i_2}, \hat{j_2}, \hat{k_2} \rangle$$

Multiplying local unit vectors $\mathbf{P}_l$ and $\mathbf{Q}_l$ by the respective base station's rotation matrix converts the unit vectors to global coordinates. Using these global unit vectors and the base stations' positions, $P_0$ and $Q_0$, a line from the base stations to the sensor, P(s) and Q(t) can be formed.

$$\mathbf{P_g} = \mathbf{M_1 P} \tag{12}$$

$$\mathbf{Q_g} = \mathbf{M_2 Q} \tag{13}$$

$$\mathbf{P(s)} = P_0 + s\mathbf{P_g} \tag{14}$$

$$\mathbf{Q(t)} = Q_0 + t\mathbf{Q_g} \tag{15}$$

Solving the intersection of the two lines provides the complete location of the sensor.

## 2.3 Brushless DC Motors

Brushless DC (BLDC) motors are permanent magnet 3-phase motors that utilize an air gap to complete a magnetic field from the stator to the rotor. Each phase within a BLDC motor has a specific phase inductance and resistance. Understanding this impedance allows the motor controller to utilize back-EMF voltage for feedback. Two important parameters in motor selection are the *velocity constant,* $K_v$, whose units are radians per Volt-second,

$$K_v = \frac{\omega_{noLoad}}{V_{Peak}}, \text{rad[Vs]}^{-1} \tag{16}$$

and the *torque constant,* $K_\tau$, whose units are Newton-meters per Ampere.

$$K_\tau = \frac{60}{2\pi \cdot K_v}, \text{NmA}^{-1} \tag{17}$$

These constants are instrumental in selecting a motor that will both have enough speed and torque for an application. The torque constant is derived from the velocity constant, which is given by the manufacturer of the motor. A BLDC motor is operated by applying a PWM signal to an electronic speed controller (ESC) that generates a 3-phase sinusoid signal as an output to the motor. ESC's normally operate on open loop control where speed is proportional to PWM pulsewidth. A closed loop signal can provide feedback about the motor to improve its performance. An example of closed loop control is field oriented control (FOC) which monitors motor phase currents to generate more precise and efficient commutation. FOC uses current sensing to dynamically control the current vector, shown as a grey vector on the left side of Figure 3, and is produced by the magnetic field. The output of the control loops that manipulate the current vector components drives a PWM controller that in turn adjusts the induced magnetic field. Another example of closed loop control is with an encoder, where angular ticks are counted as a parameter to measure the position of the rotor. Encoders are discussed further in Section 2.5.

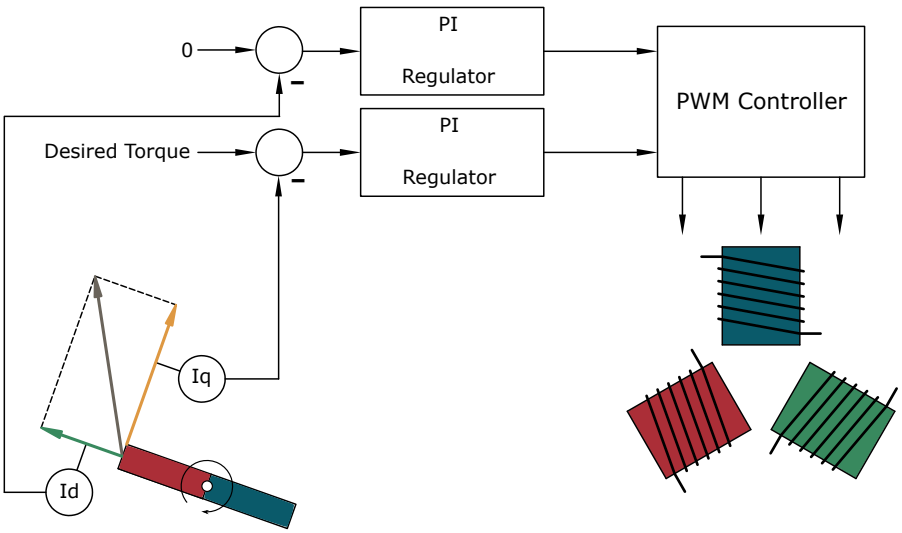



Figure 3: The torque vector components are depicted by the yellow arrow, $I_q$, and the green arrow, $I_d$. The vectors are shown as the feedback to two PI regulators. These vectors are shown radiating from the rotor. The stator is shown as three coils on the right side of the figure.

## 2.4 Velocity Control

A useful method for commutating BLDCs in a smooth manner is trapezoidal velocity control. This method directly measures an angular velocity that the motor is moving at, and controls this value with a series of

linear interpolations. As a result, the movement is equivalent to the integral of these interpolations. At spin-up and spin-down, the motor's position experiences $x^2$ change in position, and during movement in between these limits, a constant-linear movement. This relationship can be seen in Figure 4.

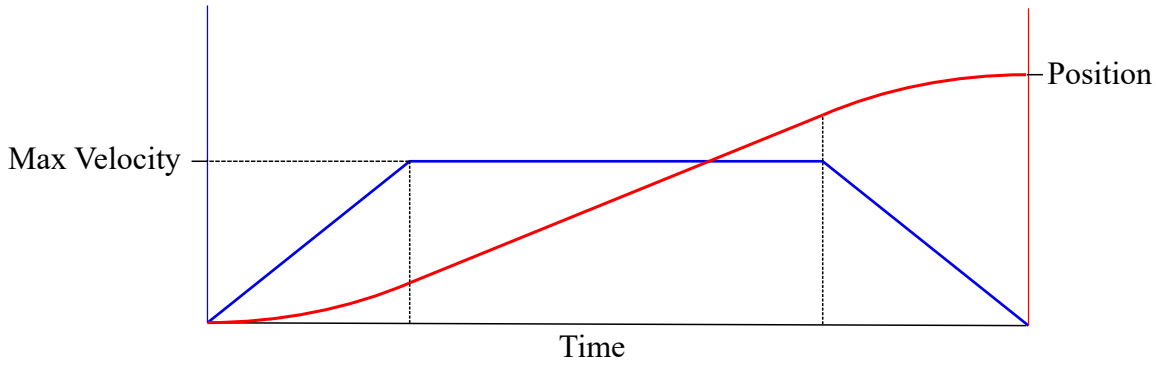



Figure 4: The blue line represents the trapezoidal velocity curve, while the red line represents the piecewise polynomial position curve. The units of velocity are encoder counts per second.

## 2.5 Encoders

Measuring the rotational position of a motor is necessary for any closed loop design. In order to accurately track the angular position of a motor, an encoder or hall effect sensor can be used. There are two main types of encoder architectures, incremental and absolute. The visual difference between the two can be seen in Figure 5. Encoders can read the encoder disc in many different ways, of the most common are optical, capacitive, and magnetic. Optical encoders use a transparent disc with opaque ticks and a photo diode to detect position. Capacitive encoders use a high frequency signal and measure a modulated change as the capacitive disc rotates, then on board electronics convert this signal into rotary pulses. Magnetic encoders work similarly to optical encoders but at a lower resolution due to the size of the alternating magnetic field areas. Incremental encoders utilize an index that determines relative position from a starting index (requiring a homing sequence), while absolute encoders show a unique index at every position. Incremental encoders are typically of a quadrature structure, meaning that the output has two channels reading the encoder disc. The two channels in the quadrature encoder are offset in phase, giving four readings per encoder tick. Hall effect sensors output a voltage proportional to a change in magnetic field [18], and this is used to track when a motor phase is being powered. These have less resolution than an encoder because the hall effect sensor is limited by the number of poles in the motor.

## 2.6 Hardware Communication

A common form of communication from computer to microcontroller is serial communication. This communication protocol sends one byte at a time at a given transmission rate known as the baud rate. The baud rate is the formal name given to the rate at which symbols are transferred over the connection. A symbol is any signaling event, such as the change from 0 to 1 in binary. In digital systems such as a microcontroller, the baud rate is equivalent to bits per second. Typically, the baud rate ranges from 9600 to 115200 baud.

## 2.7 Virtual Rendering Engines

Foremost in every VR experience is the computer generated environment. A popular platform for developing both 2D and 3D environments is the Unity engine. Unity can be used in conjunction with VR devices such as the HTC VIVE to create virtual environments. It accepts scripting in the complete languages of C#

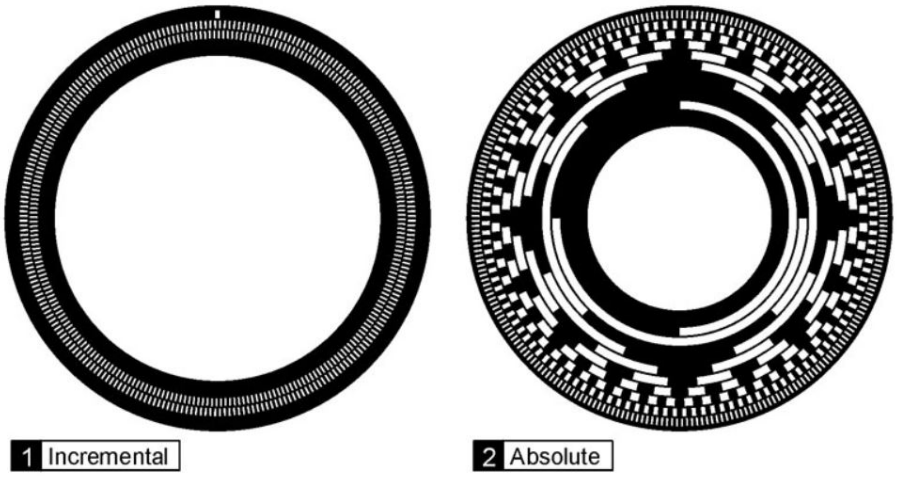



Figure 5: On the left figure, the index on the incremental encoder can be seen. The right figure shows an absolute encoder with a unique representation at every position. Reprinted from: [19]

or Java in order to add capability to objects within the game. With scripts, all parameters of an object can be controlled, such as position, weight, or color. The engine is capable of communicating with external devices through USB, COM ports, and TCP/IP networks, which is useful for integrating hardware.

# 3 Objectives

With the intent of designing a high-fidelity immersive experience, the primary objective is to create a non-invasive wearable device that provides immersive, kinesthetic feedback to a user. Creating such a device that encapsulates the user's entire body is ideal but is a herculean undertaking. In this work, we will be focused on providing mechanical feedback to a simple, single degree of freedom joint. The closest single degree of freedom joint to current virtual manipulators is the elbow.
In order to qualify high-fidelity immersion, the following specifications must be achieved.

1. The device shall dynamically limit the user's ROM without impeding the user's desire to move within this range (*position lock*).

2. The virtual representation of the arm will produce a usable parameter for the positioning of the motors with respect to virtual objects.

3. The device shall only limit the user's ROM when in contact with a virtual object.

4. The device shall simulate contact within the resolution of human anthropometry [20].

5. In order to simulate a stiff surface, the equivalent spring constant of the device should fall within $283 \pm 132$ Ncm$^{-1}$ [20].

6. The device should be capable of actuating up to 10 pounds.

The design requirements focus on the underlying needs to generate immersion and provide metrics that would be useful in forceful applications.

# 4 Design

Developing a high fidelity immersive device combines mechanical, electrical, and software control systems. The quality of this device is dependent on the design of each system and how they are able to engage with one another. In this section, we describe how *position lock* is overcome, how the actuators are controlled, how the device is packaged as a wearable device, and how the virtual environment is created.

## 4.1 Mechanical

In this section, we describe the considerations made in the mechanical design. Here, we address the problem of *position lock* and how the device will later be packaged on a user. These details provide a foundation for device manufacturing, as discussed in Section 5.

### 4.1.1 Bi-directional slot mechanism

The component that limits a user's ROM is the bi-directional slot mechanism. This mechanism must be designed such that it can be easily worn by a user. A rehabilitative arm brace provides an ergonomic solution in which to affix the other components of the device. This mechanism overcomes *position lock* and consists of two independent discs, each of which control one end point in a user's range of motion. This effect is outlined in Figure 6. The discs are concentric about the elbow's axis of rotation and the slot in each disc aligns with a pin that is affixed to the lower portion of the arm brace. The slot is designed to allow the pin to travel freely through a 170° arc. This ensures that the disc will not impede the user as they move their lower arm through their natural range of motion of 150° [21]. By rotating this slot, the ROM can be adjusted, thus simulating contact while eliminating the concern of *position lock*.

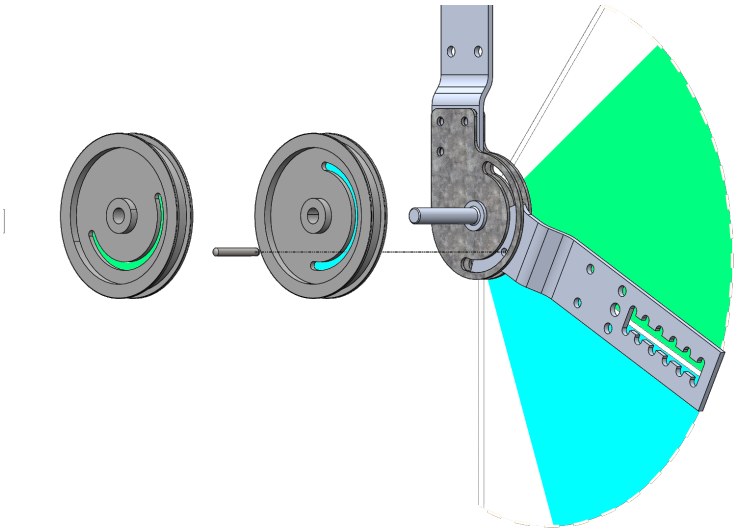


Figure 6: An exploded view of the bi-directional slot mechanism is depicted. Each disc is mounted concentrically about the axis of rotation, i.e. the elbow joint. The pin is fixed to the lower arm. The discs are able to rotate in either direction to constrain the range of motion of the pin.

### 4.1.2 Device packaging

The design of a bi-directional slot mechanism requires many electromechanical components. Configuring these components in an ergonomic package is non-trivial. The focus of our design is to mount each component as close to the shoulder as possible. This reduces the moment caused by each component. As mentioned, the base of this design is a modified rehabilitative arm brace. It is upon this brace that

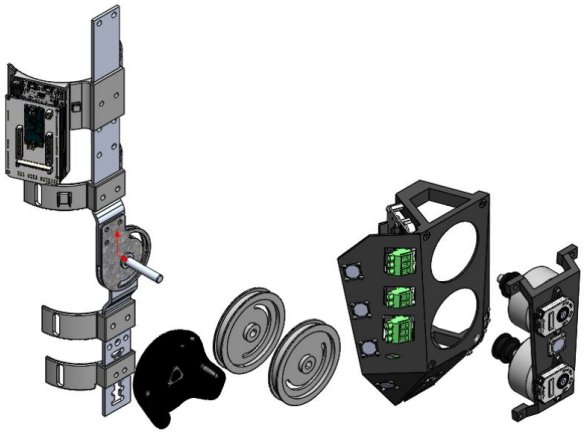


Figure 7: An exploded view of the device is depicted. From left to right: arm brace, VIVE Tracker, bi-directional slot mechanism, electronics mount, and actuator mount.

the remaining components are mounted. The upper most strap was chosen as the dedicated mounting location for the micro-controllers. A multi-purpose mounting structure was designed to house the motor controller, tracking sensors, and shields the user from moving parts. This mounting structure was stylized to accentuate the user's arm contour while providing increased visibility of the IR tracking sensors to the base stations and discrete mounting locations for other components. Following the same criteria for ergonomics and a low-profile design, the lower sensor cover was designed with similar features to the multi-purpose mounting structure. The lower sensor cover extends past the bi-directional slot mechanism to locate wiring harnesses away from the moving parts. A motor mount was designed to install directly on top of the multi-purpose mounting structure and to space the motors within millimeters of the arm brace. The motor mount provides mounting locations for an IR tracking sensor, two encoders, and two motors. Altogether, the designed packaging safely houses all necessary components for the bi-directional slot mechanism, as shown exploded in Figure 7.

## 4.2 Electrical

The bi-directional slot mechanism provides a unique solution for overcoming *position lock*. This solution remains incomplete without the ability to actuate each disc. In this section, the actuator design and the methods that define the conversion of virtual information to physical sensation are discussed.

### 4.2.1 Motor selection

The most practical way to actuate the discs is through the use of a motor. With a myriad of motor specifications, selecting a motor is a critical step in the design process. Considering a wearable device, the size and weight are the highest priorities for motor specifications. Large motors that project perpendicularly from a user's arm create a noticeable imbalance and this decreases comfort due to the fluidity of the skin.

In addition to concerns of comfort, reducing the device's response time is dependent on the motor's angular velocity. The speed at which the motor can control the range of motion and the user's lower arm position needs to achieve an average of at least 25.6 $\mathrm{rads}^{-1}$ [22]. Considering the design constraints, a BLDC motor (GARTT ML 5215 340KV) was selected.

Having selected a motor, controlling the torque and position is required for creating force feedback in a virtual environment. While velocity-based feedback controllers are available for high-speed applications, this control method is not suited for high-precision position control. To allow for position control, a motor

controller (ODrive v3.6) was chosen due to its high level of control over all parameters of up to two motors. A single ODrive can control up to two motors with closed loop feedback, and measures all parameters of the motors to be known and edited, thus fulfilling all of the needs for a motor controller for two actuators in a single package. The controller is configured using a Python library and actuates the BLDC motors using data from the virtual environment.

### 4.2.2 Interaction control

Within a virtual environment, the user is engulfed by audio and visual information. This information is only useful if it can be used to rotate the bi-directional slot mechanism to simulate the sensation of contact with virtual objects. To simulate objects, it is necessary to understand the kinematic state of the arm relative to the virtual environment. By understanding the relationship between the virtual environment and the arm, the discs are adjusted to control the ROM. At the limits of the ROM, the interaction between the lower arm pin and the bi-directional slot mechanism can provide additional information about the object. An example of an interaction is shown in Figure 8. These methods outline how the device can provide physical feedback from a virtual environment.

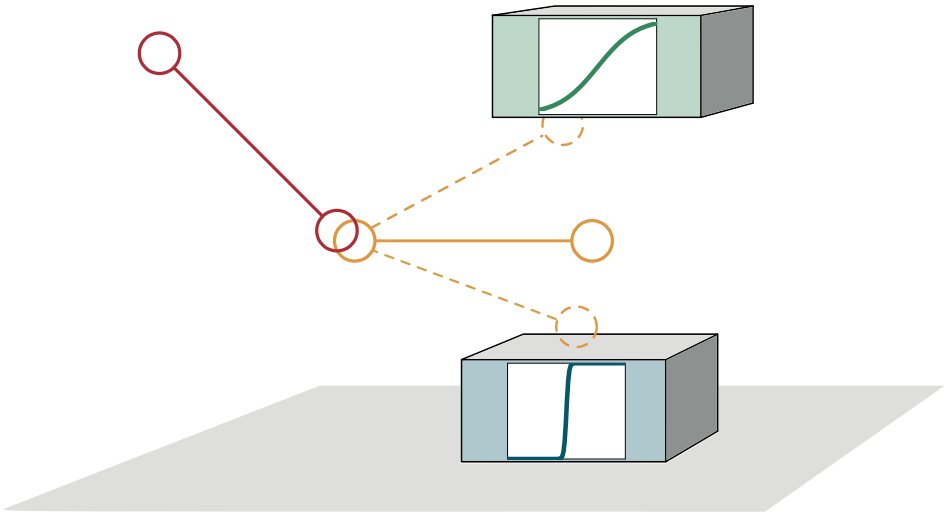


Figure 8: The virtual representation of the upper arm is depicted in solid red lines and the lower arm is solid yellow lines. The blue and green blocks represent two virtual objects. Interactions with the blocks by rotating the elbow are depicted by dotted yellow lines.

### 4.2.3 Communications

The data gathered from a virtual environment must be communicated to the the user. This data needs to be transmitted to the actuators in the form of encoder counts. The overall data flow consists of encoder counts calculated in Unity, to a microcontroller, and finally from a microcontroller to the ODrive motor controller. In Unity, external communication can be accomplished using standard C# COM ports. A COM port was utilized to send data between the Unity environment and a microcontroller then communicates over a separate RX/TX with the motor controller. The data that is shared between these systems is simply the desired encoder counts that each actuator should be positioned at. The COM port operates at 115200 baud allowing for fast data transfer. Consequently, the encoder positions are updated once per frame (30 frames per second), an update rate of 0.5 ms. The baud rate of the connection is faster than the frame update, allowing the position to be updated in real-time.

## 4.3 Software

The design of the device relies on the information encoded within the virtual environment. The link between the physical device and the virtual representation still needs to be established. We create this link by developing a virtual model of the device, tracking this model in virtual space, and communicating relevant information to control the actuators.

### 4.3.1 Virtual tracking

Critical to all virtual representations is the ability to track physical devices in the virtual environment. Tracking our device in the virtual environment is necessary for determining when the user interacts with the environment. The method for tracking our device in space is adapted from the HTC VIVE platform. The accuracy and precision of virtual tracking are dependent on an understanding of the orientation of the base stations and the object being tracked. The location and rotation of the base stations is determined by using the built in functions of the SteamVR API running within Unity. After setup, TS3633-CM1 IR sensors attached to the device output a low voltage pulse train signal that is used to calculate the sensor's position in the space. By reverse engineering the tracking method used by the HTC VIVE, we are able to track a custom controller in the virtual environment.

The results of the sensor test outlined in the Appendix concluded that the authors' attempt at virtual tracking was not reliable; a design choice was made to purchase an HTC VIVE Tracker. Integrating this Tracker provided reliable and precise spatial location of the lower arm, allowing accurate and stable joint limits to be calculated.

### 4.3.2 Virtual environment

The integrated HTC Tracker provides accurate position and rotation in Unity. However, this information is not a complete picture for generating contact. The arm was modeled in virtual space to understand the physical location of the arm. This is done in Unity by using built-in *joints*. In order to model the arm in a simple way, the HTC VIVE HMD was used as a secondary data point. This was done because the tracking data for the HMD is already well understood. In understanding the head's position, and the position below the elbow where the Tracker is mounted, the arm's complete position can be inferred. The upper arm is not directly tracked in this model, but is instead assumed by the location of the elbow and the head. This model works because the elbow only has one degree of freedom from the upper arm, and can only move in a well-defined manner. The movement of the arm can be estimated using virtual representations of the joints that physically exist in the arm. At the elbow, a hinge joint is used, simulating a single rotational axis about which the lower arm rotates. By knowing the Tracker's exact position on the arm, the hinge joint's position can be calculated. At the shoulder, a spring joint is used to force an upper arm model to be constrained between a collar-bone-like object (derived from the head's location), and the elbow's location. The spring joint allows its spring, object mass, and damping constants to be set in order to control the behavior of the multi-degree of freedom joint. By designing the arm model in this way, Unity calculates the inverse kinematics natively. Once the model of the arm is well defined in Unity, the interactions it can have need to be defined. In Unity, physics collisions are understood through the use of mesh colliders. In this context, mesh colliders are interchangeable with the terminology of hitboxes, both of which are non-rendered objects that simply trigger a change in an algorithm. Once the hitboxes are set for each interactable object, including the arm, collision algorithms can be created. These algorithms understand when two hitboxes collide, and a programmed outcome can be activated. It is the activation of the programmed outcomes that are considered interactions. By recognizing which hitboxes are in contact with other objects, the range of motion can be calculated in real-time.

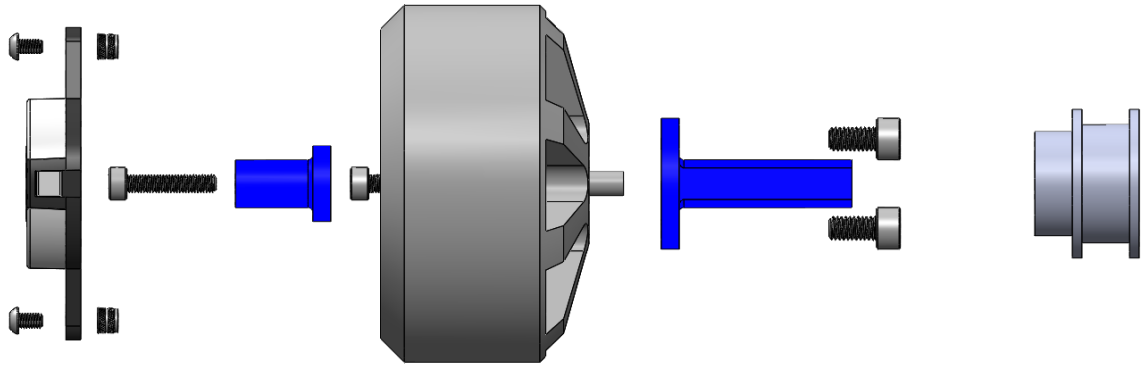


Figure 9: Shown is an exploded view of the actuator design. From left to right, the encoder with mounting hardware, the encoder shaft (highlighted in blue), the motor and mounting hardware, the pulley shaft (highlighted in blue) and mounting hardware, and the 32 tooth pulley.

# 5 Methods

*Position lock*, actuator control, device packaging, and the virtual environment have all been designed to produce a high-fidelity immersive device. In this section the methods by which the mechanical, electrical, and control systems were manufactured, assembled, and developed are described.

## 5.1 Mechanical

Producing a high-fidelity immersive device on a budget requires purchasing and manufacturing many mechanical components. With the bi-directional slot mechanism being the focal point of the device, the authors selected an acetal pulley suited for a GT2 fiber-reinforced belt. This pulley features 120 teeth and was chosen for this amount of teeth, and reasonably small diameter of 75.90mm. This allows the pulleys to fit approximately within the contour of a standard user's arm at the elbow joint. The selected pulley has a hub with a brass bushing pressed into it. To decrease the required shaft length of the bi-directional slot mechanism, the hubs were machined flush with the flange. The slot that is the basis of the bi-directional slot mechanism is directly machined into the face of the 120 tooth pulleys.
In order to drive the bi-directional slot mechanism's pulleys, drive pulleys must be mounted on the motors. The authors selected 32 tooth pulleys for this purpose. The 32 tooth pulleys were selected to generate the highest gear ratio possible with no additional gearing. The resulting gear ratio is 3.75x mechanical advantage at the bi-directional slot mechanism. Both pulleys were constrained to an 8mm shaft size based on the very first motor choice. This design choice was carried through to the final presentation of the prototype, although smaller motors were used. All of these components are shown highlighted in a bright blue color, in Figure 9.

To ensure the pulleys would fit upon the smaller motors, adapter shafts were designed and machined out of 6061 Aluminum. These shafts were made to fit upon the previously selected Gartt branded BLDCs. In order to install encoders on the BLDCs, additional encoder shafts were designed and machined to fasten to the back of the BLDCs. These shafts were turned from 6061 Aluminum. With these two engineered shafts, the actuators can be assembled.

The pulleys of the bi-directional slot mechanism are mounted through bushings on a stationary shaft, made of 6061 Aluminum, that is indirectly mounted through mounting plates to the upper arm section of the brace. The only relative motion between the pulleys needs to be the intentional motion generated by the actuators, therefore the shaft must be mounted to the upper arm that the actuators are mounted upon. Since the pulleys and lower arm rotate about the same axis, the shaft was designed with two different

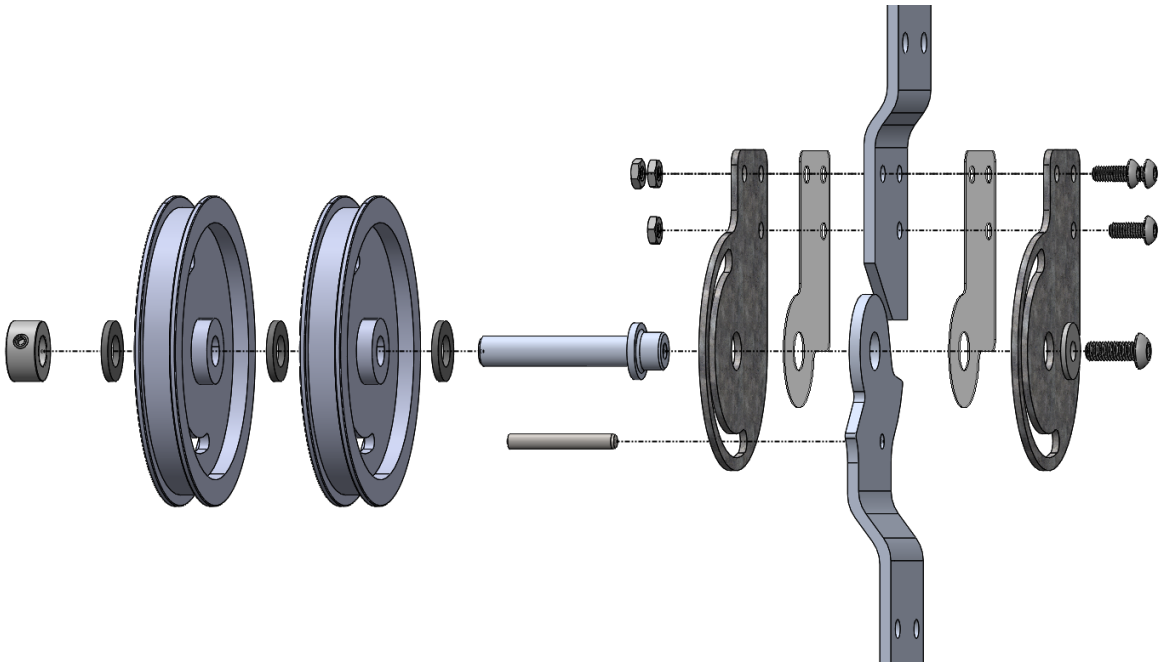


Figure 10: Detailed exploded view of the bi-directional slot mechanism is shown. Along the main rotational axis, from left to right, a shaft collar, spacing washers, the bi-directional slot mechanism, and the main shaft. The two arm brace segments are sandwiched between two low-friction spacers, fastened together by two slotted plates and accompanying hardware. The ROM limiting pin (shown underneath the main shaft) is pressed into the lower arm section and is limited between the bi-directional slot mechanism.

diameters to provide a surface for all of these components to rotate about. The shaft was designed in such a way that when its fastening screw is tightened, the shaft does not bind and create resistance in the rotational joint.

The mounting plates that union the upper arm and the main rotational shaft were based on the mounting plates that the original arm brace was manufactured with. The modified mounting plates were machined with a slot larger than the bi-directional slot mechanism, and are intended for use as an emergency hard stop. The lower limit of the plates are placed at such an angle that the hard stop would be reached before the arm could be hyper-extended. Borrowing a material choice from the original arm brace, these mounting plates were manufactured out of low-carbon, galvanized steel. The upper arm was redesigned and manufactured out of 6061 aluminum bar. Its design was based upon the original arm brace, and was modified to be longer and to facilitate all necessary mounting points for the structural covers. The lower arm was modified from the original arm brace. A bored hole was made to create an interference fit for the ROM pin in the lower arm. Additionally, holes were re-sized to allow standardization of mounting hardware. Other than these two modifications, the lower arm segment remains as purchased. The mounting plates, upper and lower arm segments, and bi-directional slot mechanism are shown in Figure 10. All structural covers including the multi-purpose mounting surface, the lower sensor cover, and the motor mount were additively manufactured on a Markforged X7.

In order to mount every component to the 3D-printed covers and modified arm brace, fasteners are needed. To fasten screws into the 3D-printed components, heat-set brass threaded inserts were used. These are easier to install and more durable than threading the 3D-printed components or using press-fit brass thread inserts. This alternative requires a significant amount of force and precise tolerances to press the inserts into the plastic to ensure they function as intended. Standard metric hardware was used throughout the design, ranging from thread diameters of M2.5 to M5. Screw diameters were chosen based on available stock, existing holes (such as the Arduino), and ease of assembly with minimal tools. Based on these actions, each component has a defined location in which it is mounted.

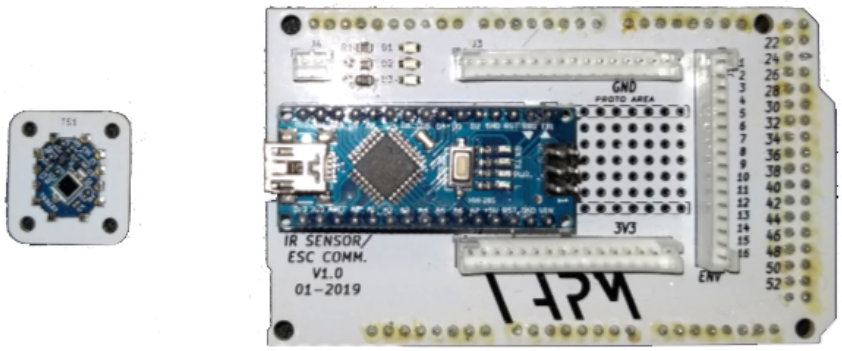



Figure 11: Left: IR sensor breakout board with three-pin JST connector surface-mounted on the back. Right: Arduino Due shield with connectors for wiring harnesses, Arduino Nano pin headers, three monitoring LEDs with current limiting resistors, and two-pin JST for serial communication with ODrive.

## 5.2 Electrical

After locating each component, we can begin to build the necessary electrical architecture. There are many challenges when wiring a large electrical structure. In this section, we discuss the methods used to facilitate simple connections and the components used to connect larger electronics.

### 5.2.1 Printed Circuit Boards

To facilitate simple connections between electrical components such as the IR sensors and the Arduino Due, two printed circuit boards (PCBs) were created. PCBs are instrumental in creating a clean, finished product that eliminates the need for non-permanent methods of electrical connections such as breadboards. The first PCB that can be designed is a breakout board for the IR sensors. The IR sensors have three important connections, 3.3V, signal, and ground; these signals were broken-out onto a larger PCB with mounting holes and a surface mount three-pin JST (Japan Solderless Terminals) connector to facilitate wire harnessing once all sensors are in place. The power pin is placed in the middle of the three-pin connector to provide protection against incorrect pinning. With this, the signal and ground pins can be reversed with minimal damage to the electrical system. To facilitate connections to the Arduino Due, a second PCB is designed. This PCB, referred to as the Arduino Due shield, collects the connections from all of the sensor breakout boards and routes them to the respective signal pins on the Arduino. The three signals from each board are routed into 16-pin connectors to facilitate wire harnessing. Each sensor's signal is connected to two digital pins, due to the fact that the Arduino tracking software needs to understand both rising and falling pulse lengths, and a single signal pin is only capable of reading either rising or falling pulses, and is not capable of reading both simultaneously on a single pin. In addition, the Arduino Due shield has pin headers for an Arduino Nano, which is the microcontroller that communicates with the ODrive. The Arduino Nano is completely disconnected, except for the ground pins, from the Arduino Due, but the shield provides a convenient mounting location for the Arduino Nano. Mounted above the Arduino Nano, are three surface mount monitoring LEDs (with series current-limiting surface mount resistors), and a two-pin JST that connects to the harness for the ODrive. Both of these PCBs are seen in Figure 11. These PCBs provide hard-wired, reliable traces for all electrical signals, to minimize external wiring and improve overall aesthetic.

### 5.2.2 Hardware

With the necessary electronic boards and surface mount connectors in place, wiring harnesses can be built and installed to complete our electrical architecture. In building each harness, a great deal of care must be put into ensuring each cable is properly connected and of the proper length. From the Arduino shield, all cables were terminated with JST style terminals and installed into the mating connectors. The harness from the shield to the array of sensors on the lower arm included an intermediate disconnect to allow this assembly to be removed with little effort. Communication between the shield, as well as the encoders, to the ODrive required DuPont terminals. The three phase outputs on the ODrive use Phoenix Contact screw terminal blocks. Each cable was terminated with wire ferrules to provide a more reliable connection. Reliable harnessing was achieved by using the correct crimping tool and routing each cable in a manner that would not interfere with the user's motion.

After mounting and locating each component, they must all be powered. The IR sensors all require 3.3V which can be supplied from the Arduino Due. The Arduino Due gets power from a USB connection to the computer. The Arduino Nano also derives its 5V power from a USB connection to the computer. The ODrive supplies the power needed to actuate the motors, it draws this power from a COTS AC/DC converter rated at 24VDC/15A. This converter provides more than enough power for the BLDC motors, due to the current transformation that is intrinsic to a BLDC.

The selected motors are Gartt ML5215 340KV BLDC machines. Given the velocity constant of the specified motors, the maximum speed at which the motors can physically rotate is equivalent to 8160 $\mathrm{rads}^{-1}$. Dividing by the gear ratio of 3.75, discussed in the Mechanical section, the bi-directional slot mechanism is capable of rotating at 2176 $\mathrm{rads}^{-1}$. Deriving the torque constant from the velocity constant, the theoretical torque constant is equivalent to 0.028 , $\mathrm{NmA}^{-1}$. Multiplying by the current limit set by the ODrive, 40A, 1.12 $\mathrm{Nm}$ is achieved at the motors' shafts. Multiplying the theoretical torque value by the gear ratio of 3.75, results in a torque of 4.2 $\mathrm{Nm}$ at the bi-directional slot mechanism.

## 5.3 Software

With the mechanical and electrical structures designed, the actuators must have an input to generate the physical ROM limits. This input is instrumental in creating the feeling of immersion. The following sections describe how the tracking within the virtual environment creates the input to the actuators. A flow-chart representation of this software is provided in the Appendix, Section 10.1.

### 5.3.1 Virtual tracking

With the electrical architecture completed, each tracking sensor is now producing a signal to be interpreted. Using an algorithm, this signal of high and low voltages must be converted to spacial coordinates. The high-low output of each sensor was connected to two interrupts on the Arduino. One interrupt was set to monitor rising edge events and the other, falling edge events. It is possible to set an interrupt pin so that it monitors both rising and falling events but this greatly complicated the calculation of time between events. With the freedom of numerous interrupt pins, the simpler, two pin solution was used. As previously mentioned, the timing between interrupts encodes the spacial position of the sensor between the base stations. The algorithm then takes the timing, or pulse width, and determines which pulse corresponds to the vertical and horizontal plane sweeps. The angular offset of each sensor is calculated from the pulse width using Equation 8. If the angular position appears to be incorrect, it may be necessary to invert the angular value but subtracting it from 180°. The Arduino then takes the four angular offsets (two for each base station) of each sensor and sends the information via serial communication to Unity.

In Unity, the first step is to split the transmitted string from the Arduino. Data per sensor is split at a delimiter (":") and stored in an array. Each item in the array is then checked with a search pattern (regular

expression). If the data matches, it is further split by a second delimiter (",") into the individual angular offsets and converted from a string to a float. With the angular offsets in the form of a float, we can begin to convert this information from a local to global context. The vertical and horizontal angular offsets are converted to Cartesian coordinates using Equations 9-11. The Cartesian vector is then rotated using three rotation matrices derived from the base station Euler angles. As matrix multiplication is noncommutative, correct order of multiplication for this application is Y-X-Z. After matrix multiplication, the vector is offset with respect to the orientation of the base station. The location of each sensor is then solved as the intersection of two lines using Equations 14-15. This process was repeated for each string of values in the array, corresponding to each sensor.

### 5.3.2 Virtual environment

The previous section described how the location of each sensor was calculated. The location of each sensor then needs to be attached to a corresponding Unity game object. The script that determines location is attached to a game object and is indexed to the corresponding sensor number. Figure 12 shows the game objects whos' location is determined by the tracking algorithm. The right half of this figure shows their actual position relative to the device. A single sensor is highlighted (blue) to show its corresponding location on the device. At this point in our development, we elected to use the HTC VIVE Tracker. Further discussion of this decision is explained in Section 6.

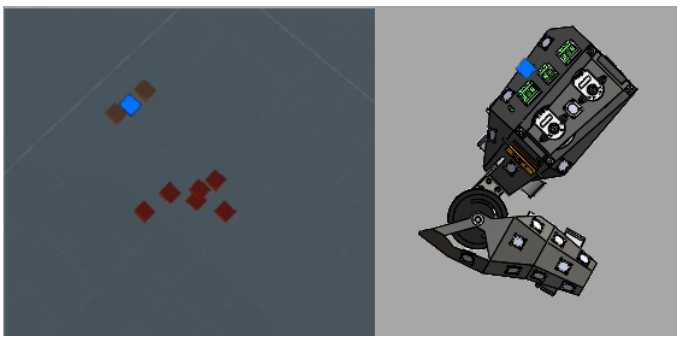


Figure 12: Sensor tracking in the virtual environment compared to the orientation of the physical device. One sensor is highlighted to show its position on the physical device.

As discussed in Section 4.1.2 the HTC VIVE Tracker was affixed to the lower arm. The location of the Tracker relative to the device served as the foundation for modeling the user's arm in Unity. Game objects (rectangular prisms) were created to represent (1:1 scale) the user's upper and lower arm. The location and rotation of the lower arm is determined using the design choices outlined in section 4.3.2. Tuning the parameters in the shoulder spring joint created an accurate representation of the upper arm. The following parameters (arbitrary units) where chosen: spring constant 50k, damping coefficient 1.5k, angular drag 5. The mass of the upper arm was also set to 0.5. These methods provided an accurate representation of the user's arm.

By accurately representing the user's arm with the VIVE components we can start to investigate how their arm is interacting with other virtual objects. Our bi-directional slot mechanism is interested in controlling the range of motion which is less dependent on the location of the lower arm but is defined by the position of the upper arm. Therefore, an array of hitboxes were modeled around the upper arm and can be viewed in Figure 13. The array consists for 41 hitboxes that are the width and length of the average person's arm. Each hitbox is offset 3.75° from the previous hitbox to cover the entire range of motion. This method, is able to understand the correct range of motion in any orientation of the arm and but does not provide the virtual understanding needed for translating this information to the actuators.

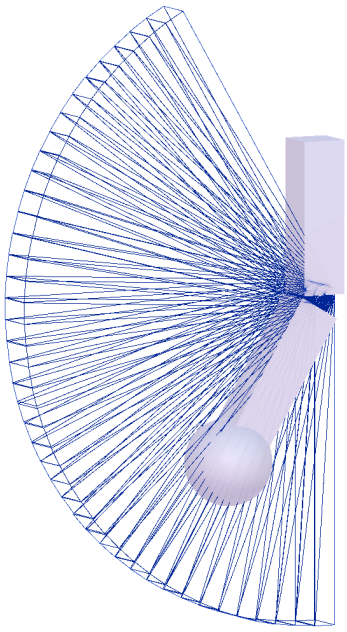


Figure 13: Virtual model of the user's arm. The array of 41 hitboxes fan out to cover the elbow's range of motion.

To extract the range of motion from the hitboxes, and thus actuate the bi-directional slot mechanism, three things need to be understood. That being: the lower bound, the upper bound, and the location of the arm within the array of hitboxes. Starting with the arm's current hitbox, each subsequent hitbox above and below the arm was checked. By starting at the arm's current location of the algorithm is able to avoid problematic scenarios where contact does not enter the entire upper or lower bounds. If an object is present, the algorithm will extract the particular hitbox number(s) for determining the upper and/or lower bounds.

Determining which hitbox number is active is not enough to actuate each slot mechanism. To resolve a useful parameter, the hitbox number is multiplied by the angular resolution to determine what angle the actuator needs to move to. This angle is multiplied by an encoder constant and the gear ratio to translate the angle into a value the actuators can understand. This information is sent from Unity to the Arduino Nano as a string. The string consists of a set number of integers followed by an indicator for which actuator should be updated. The explicit format of this data is as follows: "$(-)00000a(-)00000b$", where the five zeros can be any number 0-9, and *a* or *b* represents each actuator. The encoders are capable of 8192 encoder counts per revolution of the motor, but given the gear ratio of the actuators, numbers higher than 8192 can be expected. Negative values are optional and occasionally used to move the motors in a clockwise direction. The microcontroller parses these values per actuator, formats the data and sends the data to the motor controller over a software serial connection. This communication structure was chosen because the motor controller has an open-source library for microcontrollers in order to send parameters to the motor controller.

### 5.3.3 ODrive

The parameters from the microcontroller are sent to the ODrive motor controller. The motor controller's ability to correctly actuate the bi-directional slot mechanism is dependent on how each axis is configured. This configuration is performed over a Python terminal using a manufacturer provided interface. Key parameters that are necessary in this configuration are motor pole pairs, maximum motor current, maximum motor velocity, the characteristics of the encoders being used, acceleration and decceleration limits, and control method. The motor pole pairs for the selected motors was counted as 11 pairs. This setting

will prevent the configuration from properly controlling the motor if set incorrectly. The maximum motor current was set to 40A to prevent burning the motor coils and the maximum motor velocity was limited to 20000 counts per second to prevent extreme overshoot. The controller will return a fault if the encoder parameters are not set correctly; each encoder must be set to 8192 counts per revolution.

When errors were encountered with either axis on the controller, the Python tool was utilized to learn where the fault happened, and to clear the fault and return the axis to operational status. Common faults that were encountered included the controller missing the encoder index, and a controller error caused by moving a motor too quickly. Every fault will immediately halt the motor, signalling that a fault has occurred. The common encoder fault was resolved by re-calibrating the encoder index offset. The over-velocity fault was resolved by decreasing the acceleration/decceleration limits, forcing the motors to move into position more slowly. Any other fault can be tracked to a specific target within the controller, and resolved accordingly. By properly setting these parameters, reliable motion control was achieved.

# 6 Evaluation

In this work, we developed a device that simulates contact with virtual objects by limiting the user's range of motion. The following evaluation explores the result of each method used. The details provided in this section discuss the device's ability to meet the design objectives and how the tracking algorithm and immersion can be improved. To verify that Design Objective 1 was met, the device was placed on a test fixture and the lower arm was moved freely, while virtual objects varied the range of motion through both types of restriction: *shift* and *tighten*. In order to verify Design Objective 1, it was implied that Design Objective 2 already be verified. The success of Design Objective 2 was qualified by the ability to physically position the two motors independently by moving a virtual object into a hitbox of the virtual representation of the arm. Design Objective 3 was verified by wearing the device, as shown in Figure 14, and moving in and out of virtual objects that would cause a change in the ROM. When a virtual object is not within the user's natural ROM, the user has a maximum of 88.5% (135°) of their natural ROM as defined by an average taken from the Man-Systems Integration Standards from NASA. The ROM is limited by either: a constriction due to strap placement or a collision between the HTC VIVE Tracker and the upper structure.

Figure 14: A user testing virtual boundary contact within the test environment.

Design Objective 4 was not met because the resolution of human anthropometry at the elbow joint is described as 2° [20], while the device has a resolution of 3.75°. The resolution of the device is an artifact of the quantity of hitboxes and can be further resolved by simply adding more hitboxes. Design Objective 5 was quantified by measuring the device's stiffness by performing a spring constant experiment. The

test placed a known mass on the lower arm, while keeping the upper arm fixed upon a static fixture. By measuring the linear displacement, the linear equivalent spring constant of the joint was calculated. This yielded a single motor's spring constant through the belt and pulley system, measuring 15.2 $\mathrm{Ncm}^{-1}$. The spring constant should measure at least 151 $\mathrm{Ncm}^{-1}$ in order to simulate a stiff surface [20]. The evaluation of this device extends past our design objectives. Here, we consider the actions that would improve the spatial tracking and immersive quality of the device. The tracking algorithm was able to locate the relative position of a sensor within 7.5%, or a maximum precision of $1.2\,\mathrm{mm}$. This value is acceptable for VR applications but our algorithm was unreliable and a commercial device was implemented. However, steps can be made to greatly improve the reliability. The solution lacks any logic that is able to process incorrect or incomplete sensor data. Incorrect sensor data would come from optical distortions i.e. reflections, or communication faults. Incomplete sensor data is the result of poor base station visibility and this prevents the sensor location from updating. Adding conditions that can process such events would make the algorithm more robust. If this functionality was included, the device reliability would still fall short of the commercial product. To match the reliability of the commercial device, more sensors need to be added and their placement must be optimized. This would require developing end-use PCBs for use with the sensors that are embedded on the TS3633-CM1 modules. Integrating an accelerometer/gyroscope would further increase reliability to that of a commercial device. With these improvements, we would have a more reliable foundation for our own tracking system.

Primary to this evaluation is the device's ability to foster an immersive experience. The considerations involved in achieving an immersive experience extend beyond the stated design objectives. The first consideration for achieving an immersive experience is the feeling of contact. The initial prototype allows the user to feel contact, but when the belt split under tension the feeling of immersive is lost. A more immersive experience can be generated by improving the rigidity of the motor mount to prevent flexure. Flexure can be reduced by designing a rigid motor mount and this would allow the GT2 belts to be installed under greater tension and prevent belt slippage. Removing the concern of belt slippage would enable the user to apply more force onto objects in the virtual environment that they can interact with and enhance the feeling of contact. Immersion is also drastically improved by creating a device that accomplishes kinethestic feedback, while remaining as small and lightweight as possible. The current device weighs about 1.755kg, of which 8 percent is hardware. Ideally, hardware would be eliminated and the device would be redesigned for snap fit assembly and tool-free maintenance.

The results of the evaluation show that the primary objectives that generate a feeling of contact were met, although two of the objectives that qualify forceful applications failed. The device is able to limit the user's ROM in accordance with virtual objects, thus simulating boundary contact.

# 7 Future work

There are two considerable paths for the future of the device. The first path is to streamline the presented prototype into a more integrated and lightweight device for the arm. Streamlining can be achieved by removing all non-essential hardware through more integrated design, improving the sensor layout to obtain better spatial coverage, and refining the actuators to achieve desired stiffness while saving weight. The second path is to develop and refine a generic joint-mounted device that provides similar performance to the presented system in a lightweight modular package. This modular design would be redesigned around smaller actuators. This would prevent the device from fulfilling Design Objective 6. By abandoning this objective the device would be better suited for the primary goal of furthering immersion. Similar to the device presented in this work the next iteration would include an integrated tracking system, lightweight and high speed actuators with a large gear reduction, and the novel slotted ROM limiting mechanism.

# 8 Conclusion

We presented a device that furthers the physical immersion in virtual environments. It is shown that a bi-directional slot mechanism provides effective simulation of virtual boundaries without affecting natural motion. By evaluating the design, the requirements for a single degree of freedom device should be limited to simulating boundary contact. This work stands as a foundation for the future of immersive technology.

# 9 References


[1] Sutherland, I. E., 1968. "A head-mounted three dimensional display". In Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS '68 (Fall, part I), ACM, pp. 757–764.

[2] Belda-Lois, J.-M., Mena-del Horno, S., Bermejo-Bosch, I., Moreno, J. C., Pons, J. L., Farina, D., Iosa, M., Molinari, M., Tamburella, F., Ramos, A., et al., 2011. "Rehabilitation of gait after stroke: a review towards a top-down approach". *Journal of neuroengineering and rehabilitation,* **8**(1), p. 66.

[3] Bortole, M., Venkatakrishnan, A., Zhu, F., Moreno, J. C., Francisco, G. E., Pons, J. L., and Contreras-Vidal, J. L., 2015. "The h2 robotic exoskeleton for gait rehabilitation after stroke: early findings from a clinical study". *Journal of neuroengineering and rehabilitation,* **12**(1), p. 54.

[4] Seymour, N. E., Gallagher, A. G., Roman, S. A., O'brien, M. K., Bansal, V. K., Andersen, D. K., and Satava, R. M., 2002. "Virtual reality training improves operating room performance: results of a randomized, double-blinded study". *Annals of surgery,* **236**(4), p. 458.

[5] Bowman, D. A., and McMahan, R. P., 2007. "Virtual reality: how much immersion is enough?". *Computer,* **40**(7).

[6] Zyda, M., 2005. "From visual simulation to virtual reality to games". *Computer,* **38**(9), pp. 25–32.

[7] Pantelidis, V. S., 1993. "Virtual reality in the classroom". *Educational Technology,* **33**(4), pp. 23–27.

[8] Tsai, H.-R., and Rekimoto, J., 2018. "Elasticvr: Providing multi-level active and passive force feedback in virtual reality using elasticity". In Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, ACM, p. D300.

[9] Benko, H., Holz, C., Sinclair, M., and Ofek, E., 2016. "Normaltouch and texturetouch: High-fidelity 3d haptic shape rendering on handheld virtual reality controllers". In Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST '16, ACM, pp. 717–728.

[10] Jadhav, S., Kannanda, V., Kang, B., Tolley, M. T., and Schulze, J. P., 2017. "Soft robotic glove for kinesthetic haptic feedback in virtual reality environments". *Electronic Imaging,* **2017**(3), pp. 19–24.

[11] Rietzler, M., Geiselhart, F., Frommel, J., and Rukzio, E., 2018. "Conveying the perception of kinesthetic feedback in virtual reality using state-of-the-art hardware". In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, ACM, p. 460.

[12] Okamura, A. M., 2009. "Haptic feedback in robot-assisted minimally invasive surgery". *Current opinion in urology,* **19**(1), p. 102.

[13] Just, M., Stapley, P. J., Ros, M., Naghdy, F., and Stirling, D., 2016. "Effects of reintroducing haptic feedback to virtual-reality systems on movement profiles when reaching to virtual targets".

[14] Losey, D. P., McDonald, C. G., Battaglia, E., and O'Malley, M. K., 2018. "A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction". *Applied Mechanics Reviews,* **70**(1), p. 010804.

[15] Weisstein, E. W., 2003. "Rotation matrix".

[16] Weisstein, E. W., 2004. "Quaternion".

[17] Quiñones, D. R., Lopes, G., Kim, D., Honnet, C., Moratal, D., and Kampff, A., 2018. "Hive tracker: a tiny, low-cost, and scalable device for sub-millimetric 3d positioning". *Augmented Human,* **9**.

[18] Hall, E. H., 1879. "On a new action of the magnet on electric currents". *American Journal of Mathematics,* **2**(3), pp. 287–292.

[19] Gonzalez, C., 2017. What's the difference between absolute and incremental encoders?

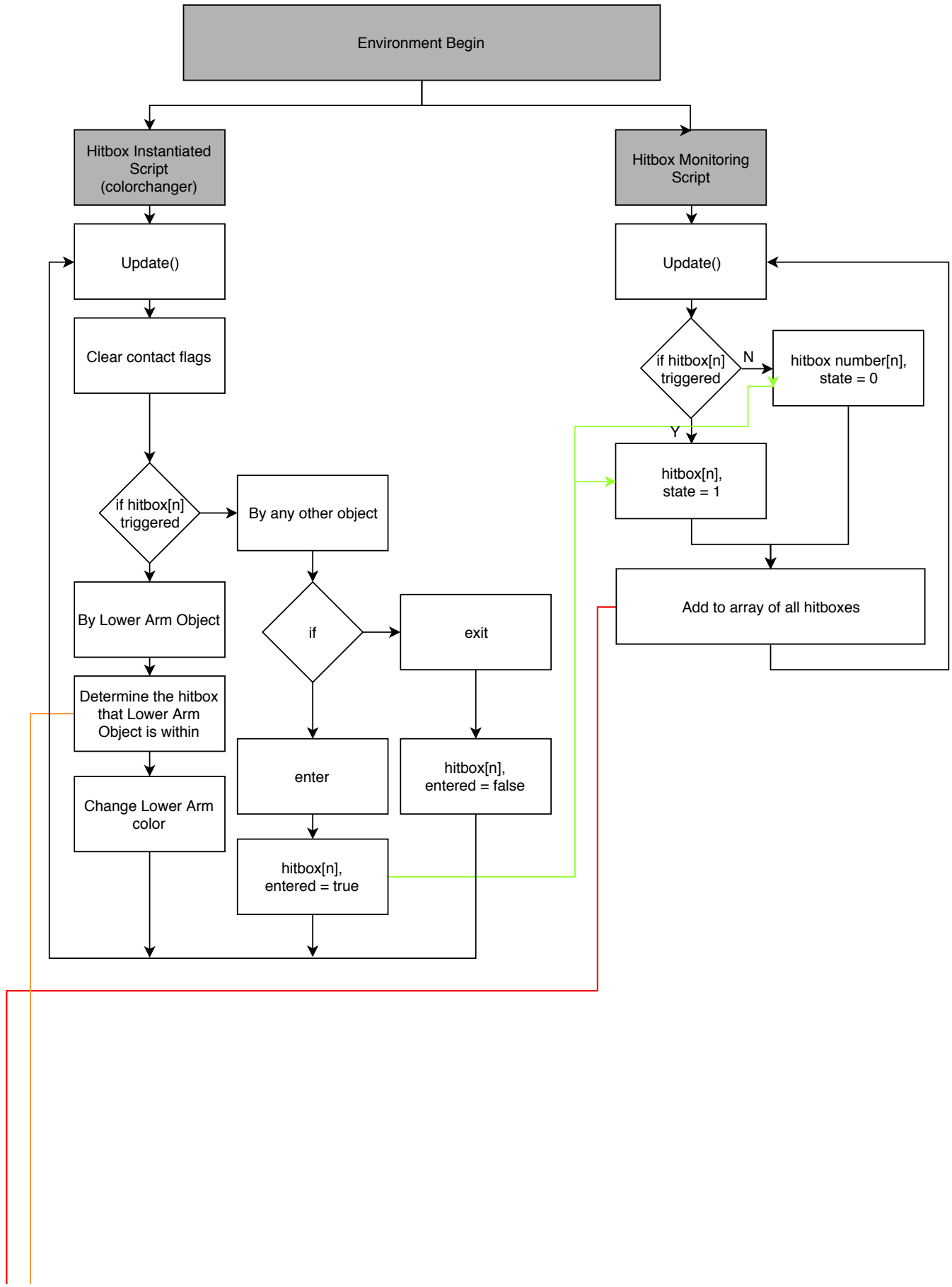
[20] Tan, H. Z., Srinivasan, M. A., Eberman, B., and Cheng, B. "Human factors for the design of force-reflecting haptic interfaces".

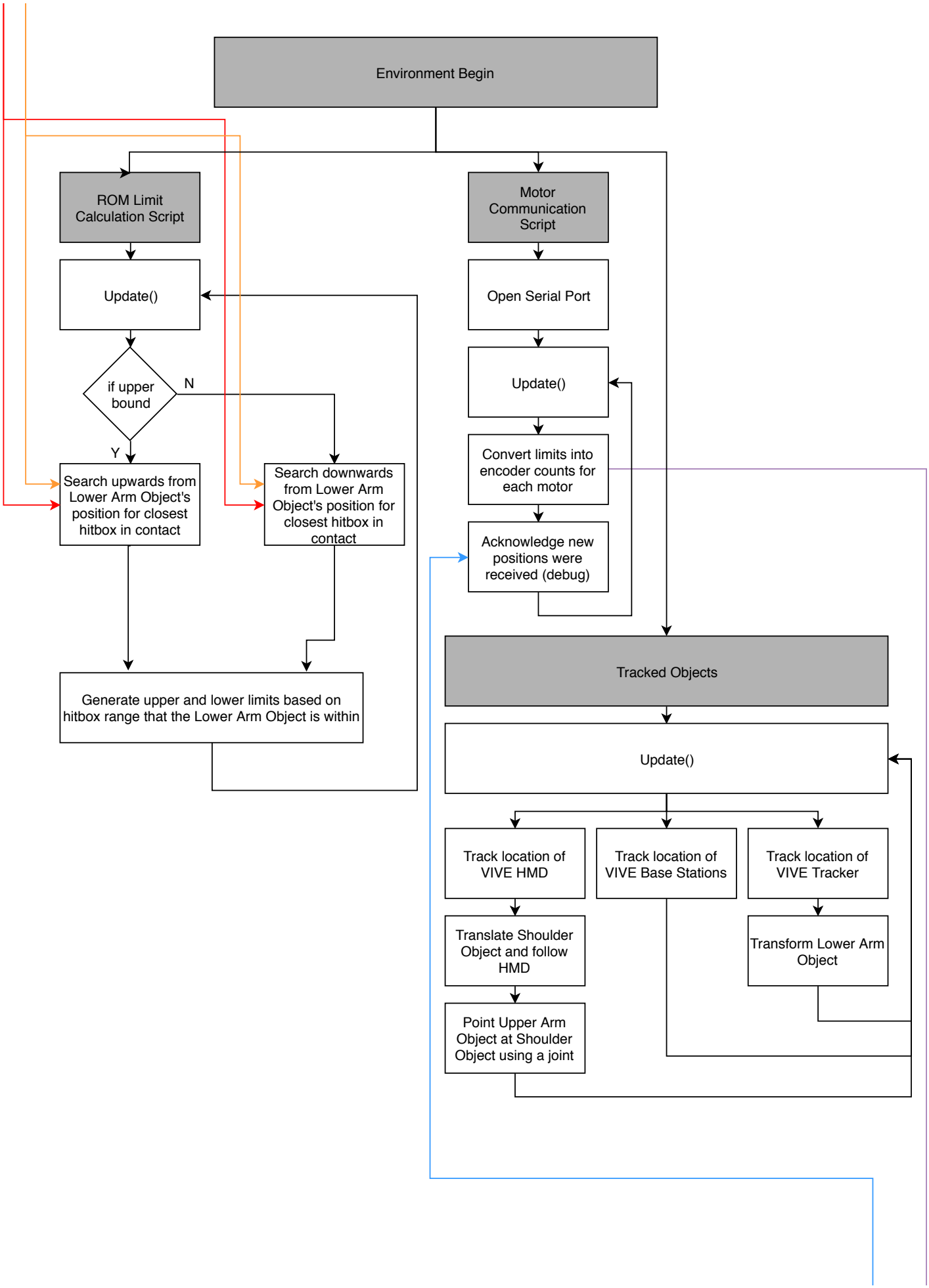
[21] Klopčar, N., and Lenarčič, J., 2005. "Kinematic model for determination of human arm reachable workspace". *Meccanica,* **40**(2), pp. 203–219.

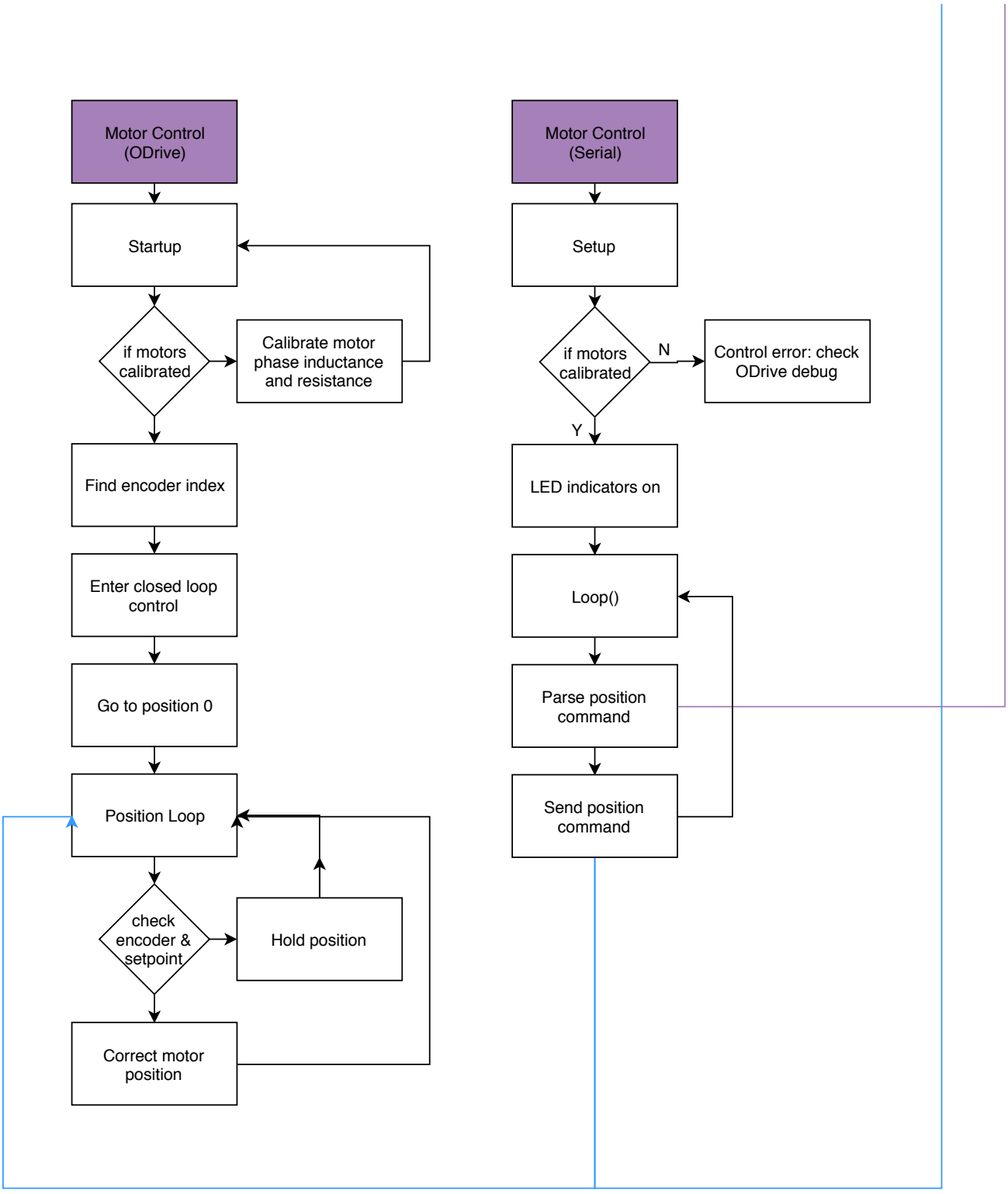
[22] Jessop, D. M., and Pain, M. T., 2016. "Maximum velocities in flexion and extension actions for sport". *Journal of human kinetics,* **50**(1), pp. 37–44.

# 10 Appendix

## 10.1 Software Architecture


Environment Begin
Hitbox Instantiated Script (colorchanger)
Hitbox Monitoring Script
Update()
Update()
Clear contact flags
if hitbox[n] triggered
N
hitbox number[n], state = 0
Y
hitbox[n], state = 1
if hitbox[n] triggered
By any other object
By Lower Arm Object
if
exit
Add to array of all hitboxes
Determine the hitbox that Lower Arm Object is within
enter
hitbox[n], entered = false
Change Lower Arm color
hitbox[n], entered = true


Environment Begin
ROM Limit Calculation Script
Update()
if upper bound
N
Y
Search upwards from Lower Arm Object's position for closest hitbox in contact
Search downwards from Lower Arm Object's position for closest hitbox in contact
Generate upper and lower limits based on hitbox range that the Lower Arm Object is within
Motor Communication Script
Open Serial Port
Update()
Convert limits into encoder counts for each motor
Acknowledge new positions were received (debug)
Tracked Objects
Update()
Track location of VIVE HMD
Track location of VIVE Base Stations
Track location of VIVE Tracker
Translate Shoulder Object and follow HMD
Transform Lower Arm Object
Point Upper Arm Object at Shoulder Object using a joint


Motor Control (ODrive)
Startup
if motors calibrated
Calibrate motor phase inductance and resistance
Find encoder index
Enter closed loop control
Go to position 0
Position Loop
check encoder & setpoint
Hold position
Correct motor position
Motor Control (Serial)
Setup
if motors calibrated
N
Control error: check ODrive debug
Y
LED indicators on
Loop()
Parse position command
Send position command

## 10.2 Sensor Test

# Measuring Accuracy of Tracking Algorithm

Andrew Zucker, Owen Medeiros, Jared Talamini

## Purpose

Infrared sensors are used in the VIVE system to conduct spatial tracking. The most important parameters of spatial tracking is accuracy. In order to test the accuracy and precision of the custom tracking algorithm developed by the authors, a quantitative test is performed.

## Background

The VIVE system uses two infrared light emitters known as base stations to generate pulses and infrared laser sweeps used in the spatial tracking algorithm. Each base station consists of two spinning motors that sweep an IR laser across the room in the X and Y directions respectively, and a timing flash consisting of IR LED's. An example of the pulse train generated by the base stations can be seen in Figure 15. The pulses that occur at a regular frequency are called the timing pulse. Both of the base stations flood the room with IR light in order to generate a distinct and regular pattern for the tracking algorithm to base its timing upon. The shorter pulses that happen between these cycles of timing pulses are the location sweeps. There are four location sweeps per cycle, each separated by a timing pulse sequence. Given that the base stations operate at constant frequency and angular velocity, the location in the room can be interpreted by a tracking algorithm.

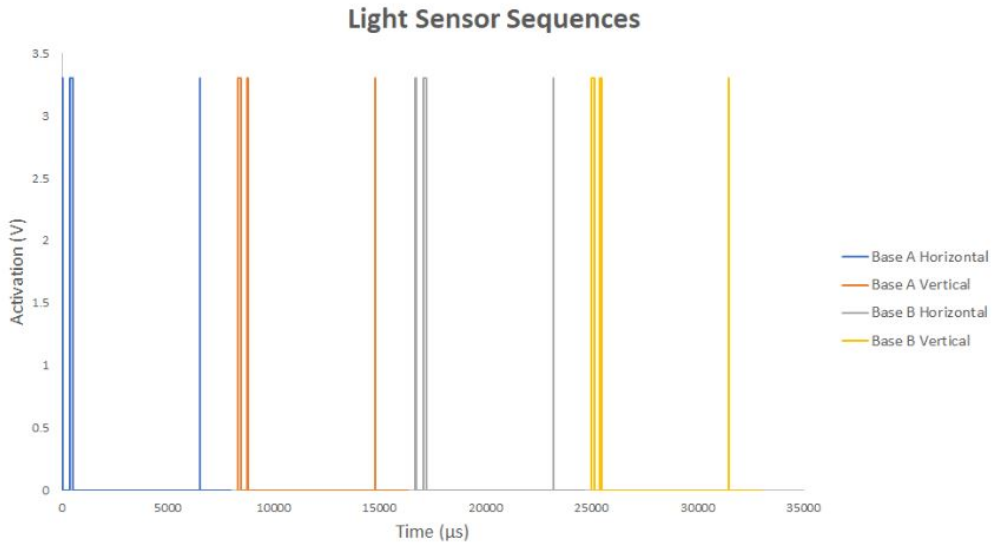



Figure 15: Depicted is an example of four sequences of data generated by two base stations and captured by a TS3633-CM1 module. Starting each sequence is two timing pulses of a pre-specified length, the length corresponding to a specific base station. The short pulse before the next sequence is the laser plane sweep location. It is to be noted that the actual output of the sensors are the inverse of this graph, as the Light-to-Digital converters are active-low sensors.

In order to understand a 3-dimensional location of a sensor, four data points are needed: X and Y spherical location from both base stations. The tracking algorithm must understand which base station is currently sweeping the room, and in which direction in order to interpret the spatial position of a given sensor. The distance between the timing pulses and the laser plane sweep is the angular location of the sensor in the given plane of the sweep. This can be converted into Cartesian coordinates using a simple transformation. It is worth stating that as this is a timing based system, data processing on the data input side is instrumental in generating accurate results.

# Materials

- 2 VIVE Base Stations, VIVE HMD
- PC with SteamVR installed, and Unity environment with tracking algorithm
- 2 or more TS3633-CR1 Light-to-Digital Converters
- Breadboard and hookup wires
- Microcontroller capable of high frequency interrupts and firmware to decode pulse train
- Calipers

# Methods

1. Set up the VIVE system in the room that testing is to be performed
2. Download SteamVR application
3. Perform "Room Setup" within the SteamVR menu such that the orientation and position of each base station can be understood within the Unity environment
4. Measure distance between two TS3633-CM1 modules placed on a breadboard
5. Measure virtual distance between two modules within Unity environment while moving throughout the designated VIVE play area

The virtual distance was measured at each of the points detailed in Figure 16.

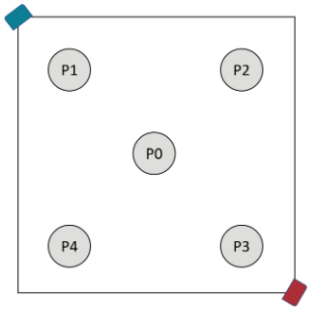



Figure 16: The test was performed in the center of the tracking area, and about each corner of the room that the base stations were installed in.

# Results

| Position | Virtual Distance | Actual Distance | Measured Error |
|---|---|---|---|
| P0 | 10.7 cm | 10.2 cm | 3.1% |
| P1 | 11.9 cm | 10.2 cm | 14.6% |
| P2 | 10.0 cm | 10.2 cm | 2.0% |
| P3 | 8.5 cm | 10.2 cm | 18.0% |
| P4 | 10.2 cm | 10.2 cm | 0% |

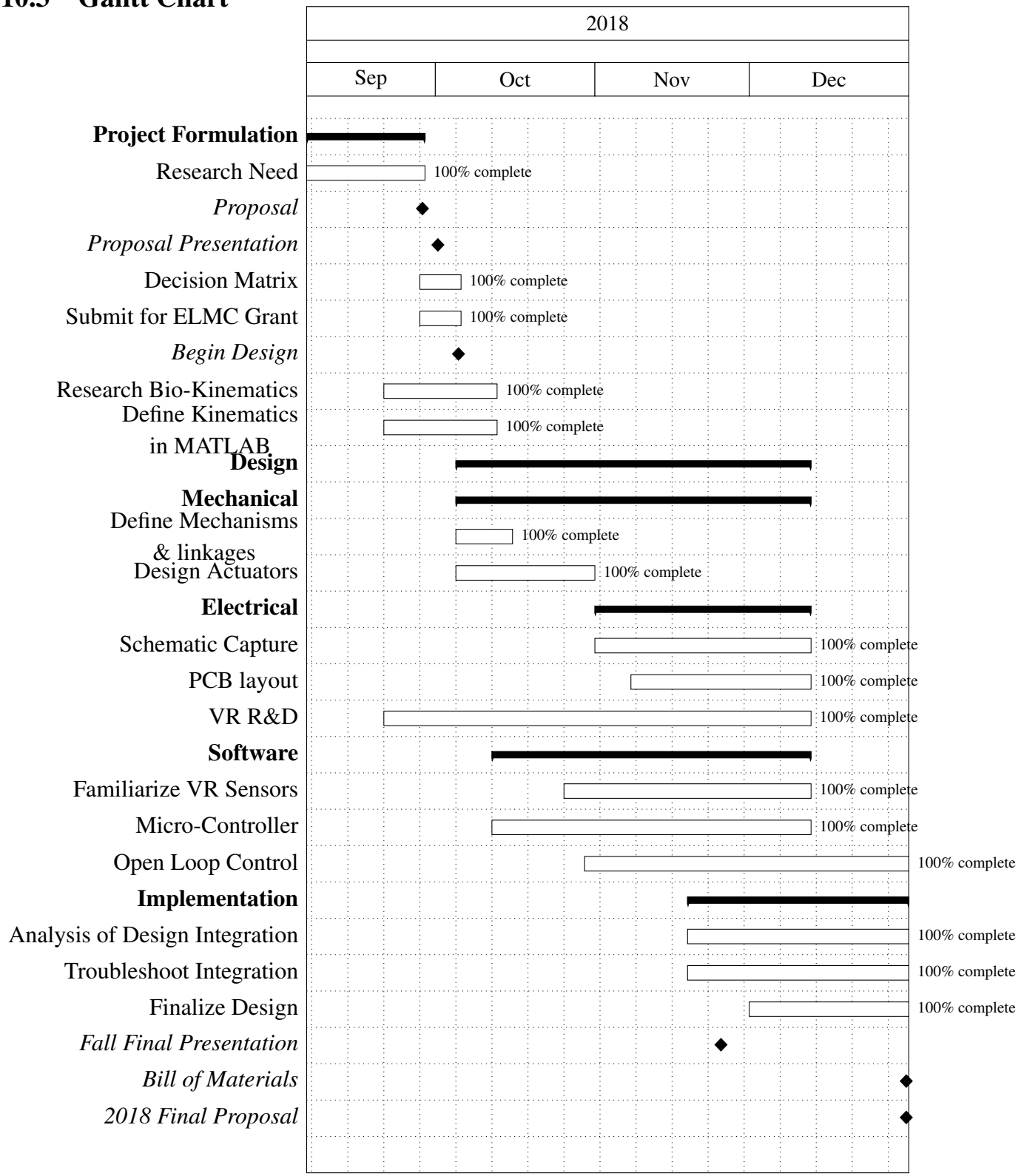
# Analysis of Results

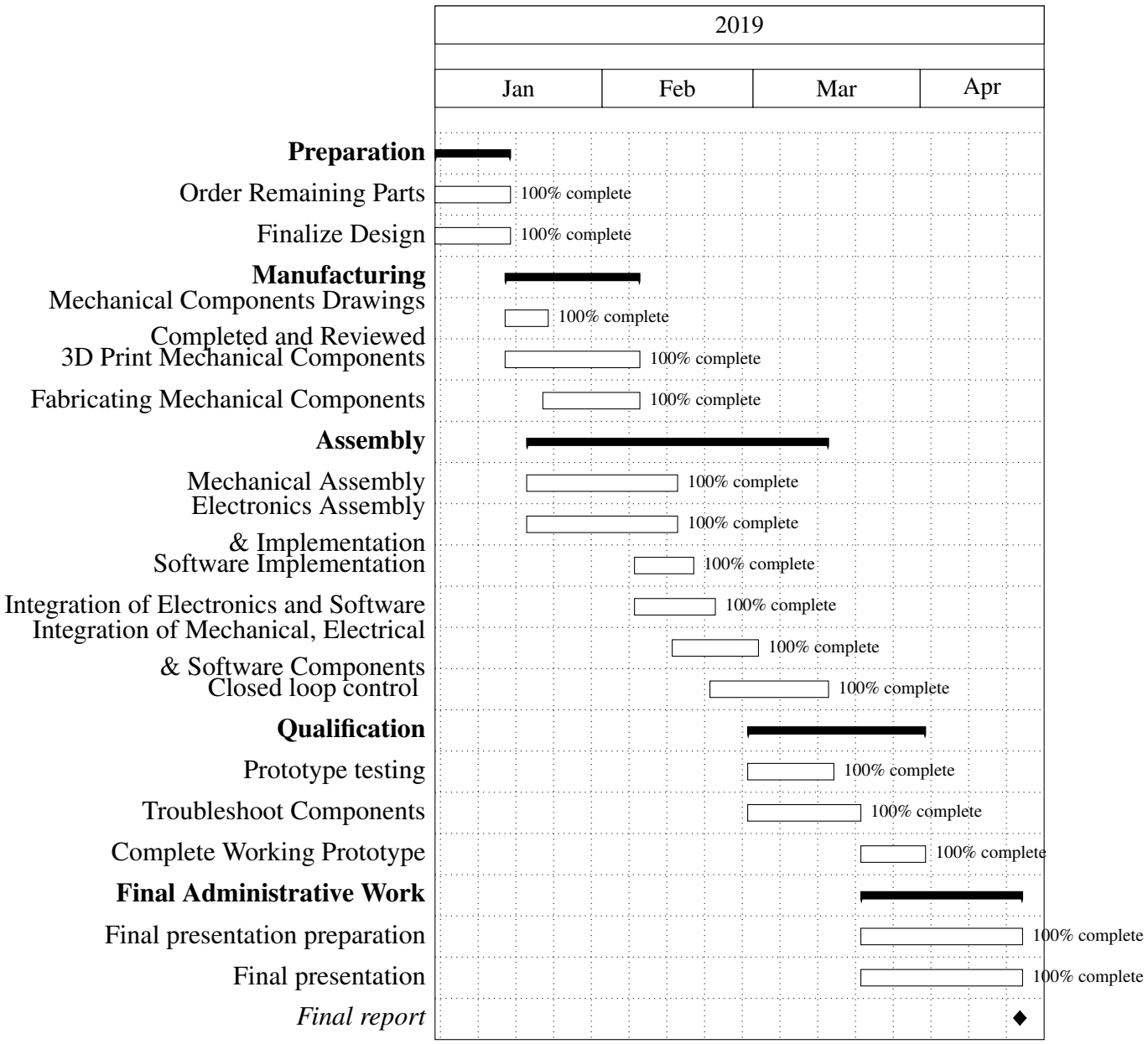
Within the center of the tracking zone, measured distance error is minimized. While closer to either base station, the accuracy of the measured distance deviates up to 18%.

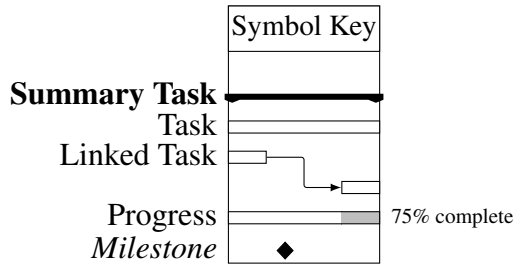
# Conclusion

While tracking using a custom algorithm can be quite accurate, this number fluctuates greatly as the sensors are moved throughout the tracking space and is unreliable in generating accurate virtual locations.

## 10.3 Gantt Chart




2019
Jan
Feb
Mar
Apr
Preparation
Order Remaining Parts
100% complete
Finalize Design
100% complete
Manufacturing
Mechanical Components Drawings
100% complete
Completed and Reviewed
3D Print Mechanical Components
100% complete
Fabricating Mechanical Components
100% complete
Assembly
Mechanical Assembly
100% complete
Electronics Assembly
100% complete
& Implementation
Software Implementation
100% complete
Integration of Electronics and Software
100% complete
Integration of Mechanical, Electrical
100% complete
& Software Components
Closed loop control
100% complete
Qualification
Prototype testing
100% complete
Troubleshoot Components
100% complete
Complete Working Prototype
100% complete
Final Administrative Work
Final presentation preparation
100% complete
Final presentation
100% complete
Final report


Symbol Key
Summary Task
Task
Linked Task
Progress
75% complete
Milestone

## 10.4 BOM

| Part Number | Quantity | Price/unit | Extended Price | Fees/Tax | Link | Total Price | Total | $1,630.80 | Remaining | $369.20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3M Strips | 1 | $7.26 | $7.26 | | Target | $7.26 | | | | |
| VESC + necessary wiring accessories | 1 | $242.92 | $242.92 | | https://diyelectricskateboard | $242.92 | | | | |
| ELECTRIC SKATEBOARD MOTOR 6355 190KV | 2 | $60.00 | $120.00 | $5.00 | https://diyelectricskateboard | $125.00 | | | | |
| Elbow Brace | 1 | $65.79 | $65.79 | | https://www.amazon.com/Or | $65.79 | | | | |
| GT2 Belt Stock | 1 | $8.99 | $8.99 | | https://www.amazon.com/Mc | $8.99 | | | | |
| Belt Clamp | 1 | $7.99 | $7.99 | | https://www.amazon.com/Mc | $7.99 | | | | |
| Belt Pulley 120 tooth, 8mm bore | 4 | $14.29 | $57.16 | | http://shop.sdp-si.com/catal | $57.16 | | | | |
| Belt Pulley 32 tooth, 8mm bore | 4 | $4.09 | $16.36 | | http://shop.sdp-si.com/catal | $16.36 | | | | |
| Arduino Due | 1 | $37.44 | $37.44 | | https://www.amazon.com/gp | $37.44 | | | | |
| S3B-PH-SM4-TB(LF)(SN) | 20 | $0.66 | $11.72 | | https://www.arrow.com/en/p | $11.72 | | | | |
| B16B-PH-K-S(LF)(SN) | 6 | $0.48 | $2.90 | | https://www.arrow.com/en/p | $2.90 | | | | |
| SPH-001T-P0.5L | 200 | $0.07 | $14.40 | | https://www.arrow.com/en/p | $14.40 | | | | |
| B2B-PH-K-S(LF)(SN | 5 | $0.15 | $0.76 | | https://www.arrow.com/en/p | $0.76 | | | | |
| Shipping | | | | $1.63 | | $1.63 | | | | |
| BNTECHGO 22 Gauge Silicone Wire Spool White 250 feet | 1 | $44.98 | $44.98 | | https://www.amazon.com/Bl | $44.98 | | | | |
| T.R.U. EL-766AW White General Purpose Electrical Tape (pack of 2) | 1 | $7.49 | $7.49 | | https://www.amazon.com/EL | $7.49 | | | | |
| 100pcs Male Header Pins | 1 | $6.99 | $6.99 | | https://www.amazon.com/He | $6.99 | | | | |
| Mini Nano V3.0 ATmega328P Microcontroller Board USB Cable for Arduino | 1 | $8.29 | $8.29 | | https://www.amazon.com/AT | $8.29 | | | | |
| 10ft mini usb | 1 | $7.99 | $7.99 | | https://www.amazon.com/UC | $7.99 | | | | |
| Shipping | | | | | | $0.00 | | | | |
| W54035ASV7, W54035ASV8 | 1 | $39.00 | $39.00 | | https://drive.google.com/ope | $39.00 | | | | |
| ODrive V3.5 **Options: 24V, with connectors** | 1 | $129.00 | $129.00 | | https://odriverobotics.com/sh | $129.00 | | | | |
| CUI AMT102-V | 2 | $39.00 | $78.00 | | https://odriverobotics.com/sh | $78.00 | | | | |
| Shipping | 1 | $12.48 | $12.48 | | https://drive.google.com/ope | $12.48 | | | | |
| JST PHR-3 | 25 | $0.10 | $1.50 | | https://www.digikey.com/pro | $1.50 | | | | |
| JST PHR-16 | 6 | $0.40 | $2.40 | | https://www.digikey.com/pro | $2.40 | | | | |
| JST PHR-2 | 5 | $0.10 | $0.50 | | https://www.digikey.com/pro | $0.50 | | | | |
| LTST-C171GKT | 10 | 0.215 | $2.15 | | https://www.digikey.com/pro | $2.15 | | | | |
| LTST-C171KSKT | 10 | $0.24 | $2.44 | | https://www.digikey.com/pro | $2.44 | | | | |
| ESR10EZPJ221 | 10 | $0.08 | $0.83 | | https://www.digikey.com/pro | $0.83 | | | | |
| Shipping | 1 | | $4.10 | $0.61 | | $4.71 | | | | |
| Miscellaneous Hardware for 3D printer prototype | 1 | $6.67 | $6.67 | $0.00 | n/a | $6.67 | | | | |
| Black PLA Filament | 2 | $19.99 | $39.98 | $2.50 | https://www.amazon.com/gp | $42.48 | | | | |
| ML5215 | 3 | $45.14 | $135.42 | $0.00 | https://www.aliexpress.com/ | $135.42 | | | | |
| 91239A144 | 1 | $10.16 | $10.16 | | https://www.mcmaster.com/9 | $10.16 | | | | |
| 91239A230 | 1 | $12.63 | $12.63 | | https://www.mcmaster.com/9 | $12.63 | | | | |
| 90576A103 | 1 | $4.18 | $4.18 | | https://www.mcmaster.com/9 | $4.18 | | | | |
| 90576A104 | 1 | $4.50 | $4.50 | | https://www.mcmaster.com/9 | $4.50 | | | | |
| 93625A102 | 1 | $10.25 | $10.25 | | https://www.mcmaster.com/9 | $10.25 | | | | |
| 91239A755 | 1 | $8.21 | $8.21 | | https://www.mcmaster.com/9 | $8.21 | | | | |
| 8975K579 | 1 | $4.42 | $4.42 | | https://www.mcmaster.com/8 | $4.42 | | | | |
| Shipping | 1 | $14.36 | $14.36 | | | $14.36 | | | | |
| B0748ZY323 | 1 | $99.00 | $99.00 | $7.24 | https://www.amazon.com/VI | $106.24 | | | | |
| 91595A179 | 1 | $8.06 | $8.06 | | https://www.mcmaster.com/9 | $8.06 | | | | |
| 8943K116 | 1 | $17.92 | $17.92 | | https://www.mcmaster.com/8 | $17.92 | | | | |
| 57485K67 | 1 | $2.32 | $2.32 | | https://www.mcmaster.com/9 | $2.32 | | | | |
| 91239A140 | 1 | $10.16 | $10.16 | | https://www.mcmaster.com/9 | $10.16 | | | | |
| 94180A361 | 1 | $12.00 | $12.00 | | https://www.mcmaster.com/9 | $12.00 | | | | |
| 90695A035 | 1 | $2.15 | $2.15 | | https://www.mcmaster.com/9 | $2.15 | | | | |
| Shipping + tax | 1 | $11.38 | $11.38 | | | $11.38 | | | | |
| 91545a470 | 1 | $3.44 | $3.44 | | https://www.mcmaster.com/9 | $3.44 | | | | |
| 91239A753 | 1 | $8.77 | $8.77 | | https://www.mcmaster.com/9 | $8.77 | | | | |
| 91239A111 | 1 | $7.38 | $7.38 | | https://www.mcmaster.com/9 | $7.38 | | | | |
| 91290A109 | 1 | $10.01 | $10.01 | | https://www.mcmaster.com/9 | $10.01 | | | | |
| 90695A031 | 1 | $7.34 | $7.34 | | https://www.mcmaster.com/9 | $7.34 | | | | |
| 94180A321 | 1 | $10.82 | $10.82 | | https://www.mcmaster.com/9 | $10.82 | | | | |
| 94180A333 | 1 | $16.15 | $16.15 | | https://www.mcmaster.com/9 | $16.15 | | | | |
| Shipping | 1 | $6.73 | $6.73 | | | $6.73 | | | | |
| 91255A540 | 1 | $9.14 | $9.14 | | https://www.mcmaster.com/9 | $9.14 | | | | |
| 91239A224 | 1 | $8.51 | $8.51 | | https://www.mcmaster.com/9 | $8.51 | | | | |
| Shipping | 1 | $6.71 | $6.71 | | | $6.71 | | | | |
| MP Select Plus Build Sheet Bed 200x200mm | 2 | $17.98 | $35.96 | $2.24 | https://www.amazon.com/gp | $38.20 | | | | |
| MK10 M7 0.4mm Extruder Nozzle Brass | 1 | $5.99 | $5.99 | | https://www.amazon.com/gp | $5.99 | | | | |
| MK10 Nozzles M7 0.4mm Threaded Stainless Steel Nozzle | 1 | $5.99 | $5.99 | | https://www.amazon.com/gp | $5.99 | | | | |
| AVAWO AW1434 | 1 | $18.96 | $18.96 | | https://www.amazon.com/gp | $18.96 | | | | |
| BNTECHGO SW14A400008-P | 1 | $13.98 | $13.98 | | https://www.amazon.com/gp | $13.98 | | | | |
| H3S16AWG | 1 | $9.99 | $9.99 | | https://www.amazon.com/gp | $9.99 | | | | |
| 91239A755 | 2 | 8.21 | 16.42 | 7.88 | | 24.3 | | | | |
| lowerPuckCover | 1 | 11.17 | 11.17 | | | 11.17 | | | | |
| lowerArmSensorCover | 1 | $18.38 | $18.38 | | | $18.38 | | | | |
| beltClampCover | 1 | $0.60 | $0.60 | | | $0.60 | | | | |
| motorMount1 | 1 | $7.60 | $7.60 | | | $7.60 | | | | |
| 14292020 | 1 | $7.59 | $7.59 | $0.47 | | $8.06 | | | | |