March 16, 2015

Andrew H. Rawicz
School of Engineering Science
Simon Fraser University
V5A 1S6

Re: ENSC 305W/440W Design Specification – LumenX^3: Projected Mobile Computer

Dear Dr. Rawicz,

I am writing in regards to the course requirements of ENSC 305W/440W. Enclosed with this letter is ObelXTech's Design Specification document for LumenX^3. We are in the process of designing a screen-less portable computing device that projects the user interface of the Windows operating system onto the surface on which the LumenX^3 is placed. Users can interact with the device by touching the surface where the screen is displayed for maximum portability and interactivity.

The following documentation details the design specifications of our innovative new product, including a detailed overview and description of our modular subsystem design. Technical design considerations are written to support the functional requirements of the overall product. In addition, an extensive test plan consisting of various test cases are included for product evaluation. Through attentive design considerations of LumenX^3 and its functional requirements, we believe this document will guide us through a successful implementation and proof-of-concept delivery.

Our diverse team consists of five talented senior engineering students, ranging in concentrations from Computer Engineering, Engineering Physics, Electronics Engineering to Systems Engineering. It consists of Carmen Tang, Davin Mok, Gary Yu, Herman Mak and Michael Ng.

I appreciate your time in reviewing our Design Specification for the LumenX^3**.** Should you have any further comments or questions, please feel free to contact me by phone at 778-995-7858 or email at yuguoy@sfu.ca.

Sincerely,

Gary (Guo) Yu
Chief Executive Officer
ObelXTech

Design Specification

LumenX^3

Version 2.2.4

March 16, 2015

Team Members:

Carmen Tang

Davin Mok

Gary (Guo) Yu

Herman Mak

Him Wai (Michael) Ng

OBELXTECH

CONFIDENTIAL

Abstract

This document details the design specifications for the LumenX^3 proof-of-concept model. The purpose is to give the reader a detailed look at each of the three subsystems from hardware to software. This will include design approaches and justifications explaining how our design choices meet the functional requirements for our proof-of-concept model (marked with P1) in the functional specification document.

The LumenX^3 proof-of-concept model will meet the following major requirements:

- Projection of an angle-corrected screen onto the surface on which the device is placed
- Interaction with the device through single finger taps
- Seamless integration of all hardware and software, invisible to the end-user
- All hardware components are enclosed within a rigid shell while keeping overall device size to a minimum to maximize portability

Each of the functional requirements described in the document “Functional Specification – LumenX^3: Projected Mobile Computer” [1] will be addressed by design specifications in their respective system subsections. Hardware choices will be justified by examining a set of specific system performance requirements and motivations behind our design choices are explored.

The final section of the document provides a set of preliminary test procedures to verify the functionality of the LumenX^3 proof-of-concept model. The test plan consists of sections examining the rising mechanism, touch detection, and projection user steps and expected behavior.

Table of Contents

List of Figures

List of Tables

Glossary

Bitstream	A sequence of digital bits
Capacitive touchscreen	A touchscreen that takes in user input by relying on the electrical properties of the human body.
COM	Communication port, a serial port interface on modern computers
Framerate	The frequency at which an imaging device produces consecutive images.
HDMI	High-Definition Multimedia Interface – a standard for connecting high-definition video devices
Homography	The relationship between two images of the same planar surface, usually used in computation of camera motions in computer vision.
IEEE	Insitute of Electrical and Electronics Engineers, the world's largest association of technical professionals
I/O	Input-Output
IR	Light that processes wavelengths ranging from 700nm to 1mm
LED	Light-emitting diode: a pn-junction diode which emits light when activated
Lumen	The SI derived unit of luminous flux
Microcontroller	A small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
Microsoft Windows 8.1	An operation system made by Microsoft, the most popular OS on x86/x64 devices, first reached general availability on October 17, 2013
Multi-threaded program	A program that can serve more than one user at a time and to manage multiple simultaneous requests without the need to have multiple copies of the programs running within the computer.
Object Oriented design	An approach to software design in which the software is design to be a system of interacting objects
OS	Operating System, software that manages computer hardware and software resources.
Perspective correction	The process of correcting a distorted user viewpoint through the use of computer software and/or mechanical devices.
PWM	Pulse-width modulation, a technique used to encode a

	message into a pulsing signal
RAM	Random-access memory, a form of computer data storage.
Regression testing	A type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system.
SoC	System On A Chip, an integrated circuit that integrates all components of a computer onto a single chip.
USB	Universal Serial Bus, an input/output interface standard for data transmission between electronic devices
User Interface (UI)	The space where interactions between humans and machines occur.
Wi-Fi	A wireless networking standard based on radio wave communication to provide Internet and local area network connections
x86/x64	32 bit and 64 bit computer architectures.

1.0 Introduction

The LumenX^3 (pronounced "lumen-ex-cubed") is the next addition to everyone's smart device portfolio. Designed with portability and collaboration in mind, the device will utilize projection to display a screen that is not limited by the size of the device and employ hand tracking techniques to give users a whole new interactive experience. Our aim is to create a durable lightweight device that promotes interactivity and collaboration among groups by using the surface of the projected screen as a plane to take in touch gesture inputs.

This design specification document will outline the following:

- The LumenX^3 product and its modular subsystem design
- An overview of the proof of concept model and its comprehensive product features
- Technical details of system design to support the functional requirements
- A set of detailed test plans to examine proper functionality

1.1 Scope

This document describes the design details of the LumenX^3 and explains how the design meets the functional requirements as described in the Functional Specifications. The design specification includes all requirements for a proof-of-concept system and additional stretch goals as time permits. As we are focusing on the proof-of-concept system, only design considerations pertaining to the functional requirements marked P1 or P2 will be explicitly discussed. These technical details will be used as reference throughout the design and implementation phase and will be referred to in future documents.

1.2 Intended Audience

The design specification document is intended to be used by all members of ObelXTech. Design engineers shall refer to the specifications as overall design guidelines to ensure all requirements are met in the final product. Test engineers shall use this document to implement the test plan and to ensure correct functionality of the LumenX^3.

2.0 System Overview

The LumenX^3 will be composed of three subsystems that include: the Core subsystem, the Projection subsystem, and the Touch Gesture Recognition subsystem. The Core subsystem will be responsible for all of the device's computing needs and will also be providing system status information to the users. Additionally, the Core subsystem will serve as a central hub where all subsystems are connected. The Projection subsystem will be responsible for providing the user interface, and the Touch Gesture Recognition subsystem will be responsible for retrieving user input in the form of touch gestures. The layout of these subsystems within the LumenX^3 system is shown in Figure 1.

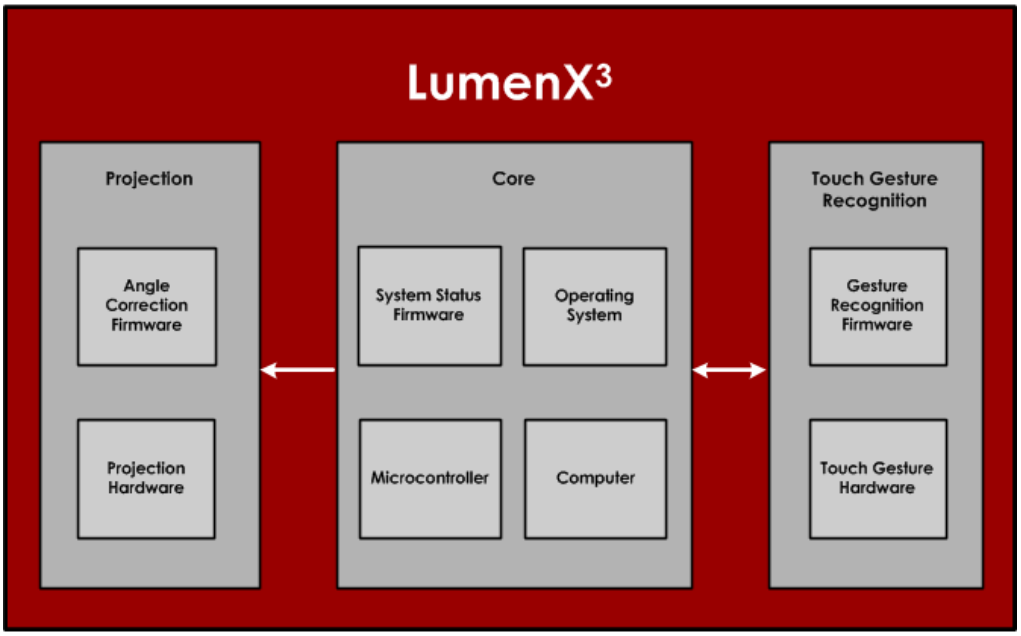


Figure 1 – Subsystem layout for the LumenX^3 system

The 3 subsystems will operate using the following hardware components:

- Core subsystem: MeegoPad T01 [2], Arduino Uno [3], and LED's
- Projection subsystem: AAXA P3 Pico Projector [4]
- Touch Gesture subsystem: Leap Motion Controller [5]

The MeegoPad T01 computer will be outputting a modified video stream that has been compensated for angle distortion via the HDMI port to the P3 Pico Projector. The Leap Motion Controller will be delivering touch gesture information to the MeegoPad T01 through a USB 2.0 connection. The Arduino Uno will drive a set of LEDs and receive instruction from the MeegoPad T01 through a USB Hub that is connected to the MeegoPad T01's USB port. As a result of the limited number of USB ports on our small computing unit, the USB Hub will remedy that restriction by providing additional USB ports allowing a keyboard and mouse to be connected for debugging use. Finally the MeegoPad T01 will be running **Microsoft Windows 8.1** [6] which natively supports an on-screen keyboard and a variety of touch gestures. A

detailed block diagram of how we will be connecting these specific hardware components is shown in Figure 2.

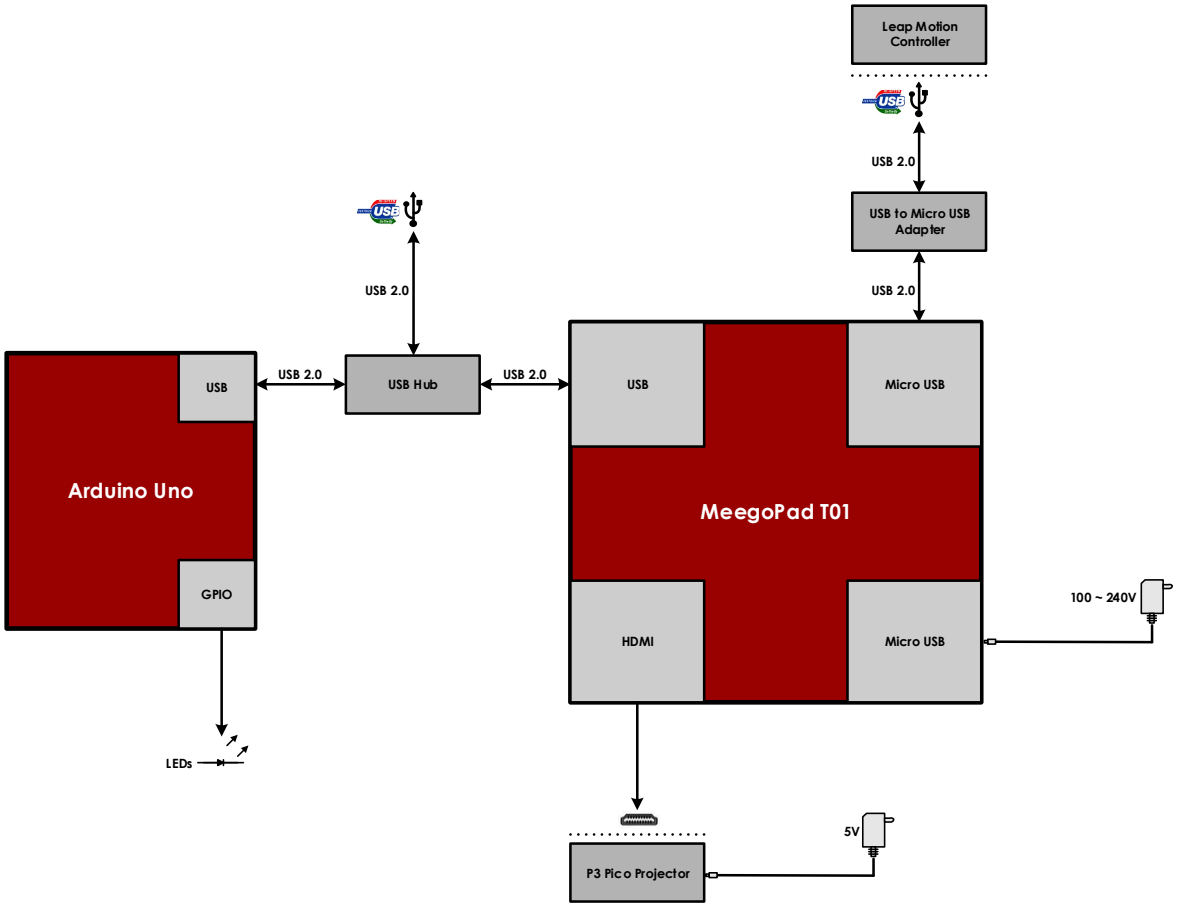


Figure 2 – Block diagram for the LumenX^3 system

3.0 User Experience

The LumenX^3 is designed from the ground up to provide a seamless user experience. By combining the best qualities from tablets and notebooks into the LumenX^3, users can experience the best of both worlds through our careful design and integration of hardware and software. The market has many products that focus on great user content consumption experiences for individuals, but very few design for sharing content among multiple users while still remains portable. The LumenX^3 fills this gap perfectly by replacing the traditional displaying unit with its sophisticated projection mechanism.

The casing is designed to be continuous and simplistic to provide the user an effective, smooth and intuitive way to operate the device. Underneath the clean, structured appearance is a set of selectively chosen components laid out in a specific manner. This ensures LumenX^3's stability for extended usage while maintaining a portable size. The same strict design influences extend beyond the unit itself into the surroundings as well. Careful placement of the power cable, status light, and side buttons means that things are never in the way yet will remain within arm's reaches. The projected screen is larger than most tablets' screen size for convenient single or multi-user content consumption. Finally, since it does not rely on capacitance or pressure to detect touches, users can use the LumenX^3 in greater number of scenarios. The innovative user experience is visualized in Figure 3.

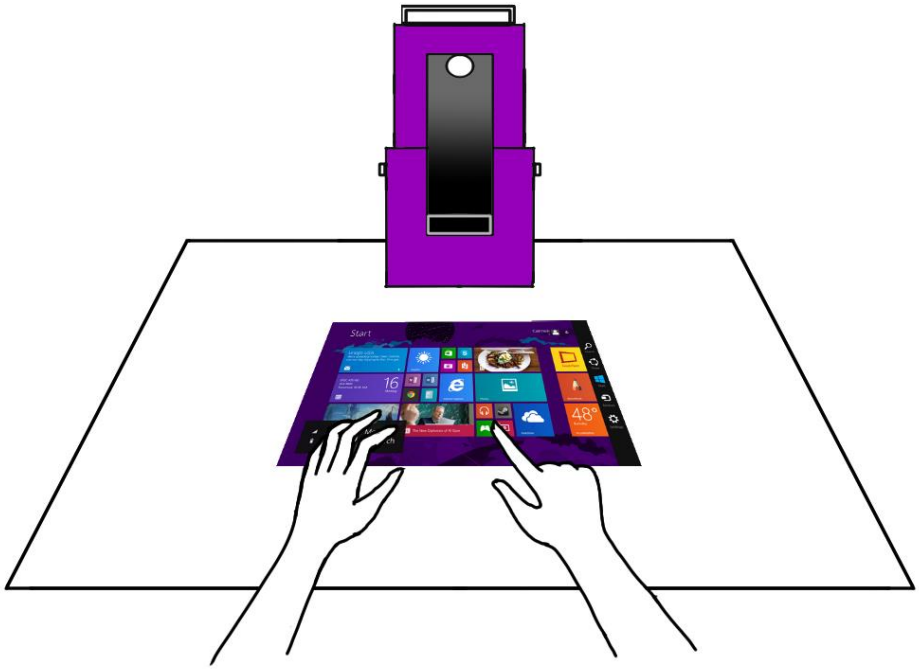


Figure 3 – First person view of user operating LumenX^3

4.0 Product Design

4.1 System Design

4.1.0 Core Subsystem Design

Computer Design

The Core Subsystem of the LumenX^3 is the heart of the experience for the customer. It must be capable of delivering exceptional performance to the end user while also acting as the integration center for all other subsystems. On immediate inspection, there are many components capable of powering the LumenX^3 but the MeegoPad T01 single board computer was chosen for a couple of important reasons. Firstly, the MeegoPad T01 is able to deliver the desktop-like utility shown in Table 1, while maintaining a super compact form factor. It has a powerful processor with quad core performance and low electricity consumption. Secondly, the MeegoPad T01's processor enables the LumenX^3 to run a full desktop Microsoft Windows 8.1 OS as required in our functional specifications. Other devices running ARM processors may offer higher frequencies and more processing cores, however they can only run extremely limited versions of any modern Operating System.

Table 1 – MeegoPad T01 Specifications [2]

Device	MeegoPad T01
SoC	Intel Atom Z3735F "Bay Trail"
Processor	Quad Core @ 1.33 GHz (Burst @ 1.83 GHz)
Storage	32 GB eMMC + microSD slot up to 64 GB
Memory	2 GB DDR3L-1333
Graphics	Intel HD Graphics
Connectivity	802.11 B/G/N Wi-Fi and Bluetooth 4.0
Power Supply	5V/2A
Miscellaneous	1x USB 2.0 Port, 1x micro USB 2.0 port, 1x HDMI port

We are using a medium-frequency quad core processor over the more common high-frequency dual core processor because the LumenX^3 is designed for content consumption. The additional processing cores allow users to run more programs concurrently without impeding system performance and responsiveness. Memory size is also a critical factor for the smooth running of Windows 8.1. Since it has 2GB of RAM, users can comfortably work on the LumenX^3 without experiencing system performance degradation. The MeegoPad T01's **SoC** has built in high speed Wi-Fi and Bluetooth connectivity so users of the LumenX^3 can connect with many wireless networks. In terms of expansion, the MeegoPad T01 offers a SD card slot that supports up to 64

GB. This means that we can add storage to the LumenX^3 if the user's storage requirements ever increase. Most importantly, the MeegoPad T01 comes with an integrated HDMI output port and support for our specified 720p high-resolution output for our projection subsystem. Finally with its built-in USB 2.0 ports, it can facilitate communications between our Arduino Uno status indication board, Leap Motion Controller and other end user peripherals.

Status Indication Design

The System Status Indicator consists of two parts: the first part is a Windows System Level Service and the second is a microcontroller. To meet the **IEEE** Std 1621 -2004 standards [7] we decided to implement three individual states each represented by a unique LED color. To achieve our goals, we are utilizing an Arduino Uno because it meets our current needs but provides a strong foundation to prototype future developments and stretch goals. It contains a good assortment of connectivity as shown in Table 2, including a built in serial interface, analog inputs, and digital I/O pins and offers a wide array of shield add-ons. Most importantly, 6 of the digital **I/O** pins have **PWM** capabilities that are fundamental for proper control of internal fans and also required to achieve automatic rising and falling for the LumenX^3 outlined as a P3 stretch goal.

Table 2 – Arduino Uno specifications [3]

Device	Arduino Uno R3
Microcontroller	ATmega328
Digital I/O Pins	14 (6 PWM)
Analog Input Pins	6
Flash Memory	32 KB
SRAM	2KB
EEPROM	1KB
Clock Speed	16 MHz
Operating Voltage	5V

On the Core Subsystem, we decided to implement a Windows System Service to communicate with the Arduino Uno microcontroller. The service is loaded during the boot phase of the MeegoPad T01 and on startup will perform a function to send an instruction over **COM** port. On shutdown, another function is performed which sends a different instruction over COM port. This service is virtually invisible to the end user and sufficient protection is set to ensure that it cannot be altered or changed in such a way to hurt user experience.

Meanwhile, the Arduino Uno microcontroller will be connected and powered by a USB hub provided by the MeegoPad T01. This direct connection allows the interfacing to be carried out with minimal design complexity. When the user turns on the MeegoPad T01, it will start up the Arduino Uno. The Arduino Uno waits until it receives a command over COM port. As the Windows 8.1 OS boots up on the MeegoPad T01 it will load the Windows System Service. The Windows

System Service opens the same COM port and sends instructions as needed. Two LED's, one green and one red, are attached to the Digital I/O pins of the Arduino Uno. The green LED is triggered by the startup function of the Windows Service whereas the red LED is triggered by the shutdown function. The Windows System Service is designed to run over the lifetime of the Windows session so the LED status indicated should be accurate and reliable.

4.1.1 Projection Subsystem Design

The main hardware device required for the projection is a projector. Since we were severely restricted on size, we opted for a small pico projector that would provide enough brightness and a high resolution. The AAXA P3 Pico Projector was our projector of choice, boasting more than High-Definition (HD) 720p resolution and 50 Lumens of brightness, which are rarely found in pico projectors given their tiny size. This AAXA P3 Pico Projector is still small enough to fit in the palm of one's hand and also has an HDMI port, which works perfectly with the MeegoPad T01. With its set of specifications, as shown in Table 3, the AAXA P3 Pico Projector is a great fit for our proof-of-concept prototype of the LumenX^3.

Table 3 – AAXA P3 Pico Projector Specifications [4]

Max Brightness	50 Lumens
Max Resolution	1024x768
Contrast Ratio	1000:1
Projection Lens	Manual Focus
Projection Image	Available Size 7 ~ 80 inch
Lamp	Triple RGB LEDs with Vibrant Color Technology Life 15,000hrs
Aspect Ratio Control	16:9
Battery Life	65+ minute Li-Ion battery life
Dimensions	4.6" * 2.6" * 1.4"
Weight	0.8 lbs
Power Consumption	11.5w
Power Supply	5V 3.0A
Video In	HDMI

Perspective Correction Driver Design

The position of the Pico projector will be at a height of 29cm and projecting downward at an angle of 60 degrees to produce a screen that is mostly in focus (over 80%) and has a screen large enough for a user to comfortably use it – about 10 inches diagonally before perspective correction and approximately 8.5 inches diagonally after perspective correction. The higher the projector is moved, the larger the screen however the larger the case must be. The lower the projector is moved, the smaller the projection angle is in order to keep a larger screen, however the portion of the screen that stays in focus becomes smaller very quickly. As well, the Leap Motion Controller has a finite range in which the screen must stay within to detect gestures. With screen size and

clarity and the Leap Motion Controller range as our most important factors, we decided to place the projector in this position, which is also illustrated in Figure 4.

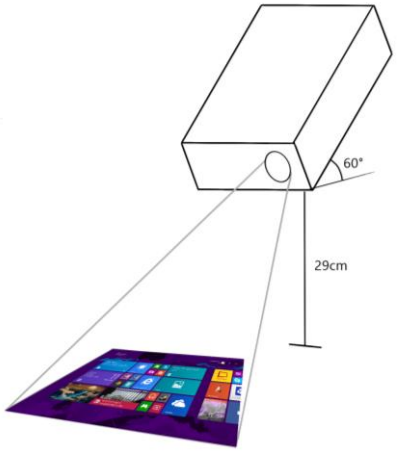
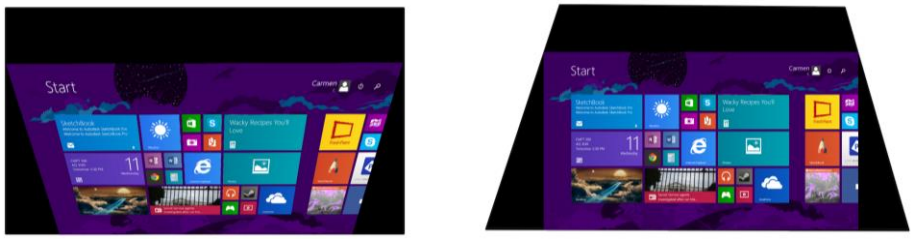


Figure 4 – Sketch of the projector placement and the resulting skewed projection

Since the projector is projecting at an angle, the resulting screen appears as a trapezoid. Consequently, the resulting operating system screen will be skewed, shown in Figure 5 a). In order to transform this trapezoidal screen back into its original rectangular form of the same proportions but of smaller size, the screen must be corrected by pinching in the screen at the top, as shown in Figure 5 b). When testing the MeegoPad T01 computer with the Pico Projector, the only available resolution was 1280 x 720, a high definition resolution. Thus the driver will be written to project at this resolution.

Operating System Output Screen **Resulting Projection**

a) Original computer screen, without perspective correction



b) Perspective corrected computer screen

Figure 5 – Operating System screens before and after ideal perspective correction and their resulting trapezoidal projections

A Windows display driver works by copying over a **bitstream** of data from the source (the Operating System) to a destination, the primary output monitor (Pico Projector). This bitstream describes each pixel colour of the entire screen, beginning with the pixel at the top left corner. This pixel will be referenced as being in pixel location (0, 0), in which the x-axis describes the width of the screen from left to right and the y-axis describes the height of the screen from top to bottom. The next pixel described in the bitstream will be the pixel one to the right: pixel (1, 0). After a whole row of pixels, the bitstream moves on to describe the next row in the same way, starting from the leftmost pixel, and repeats this for the rest of the screen. This is illustrated in Figure 6.

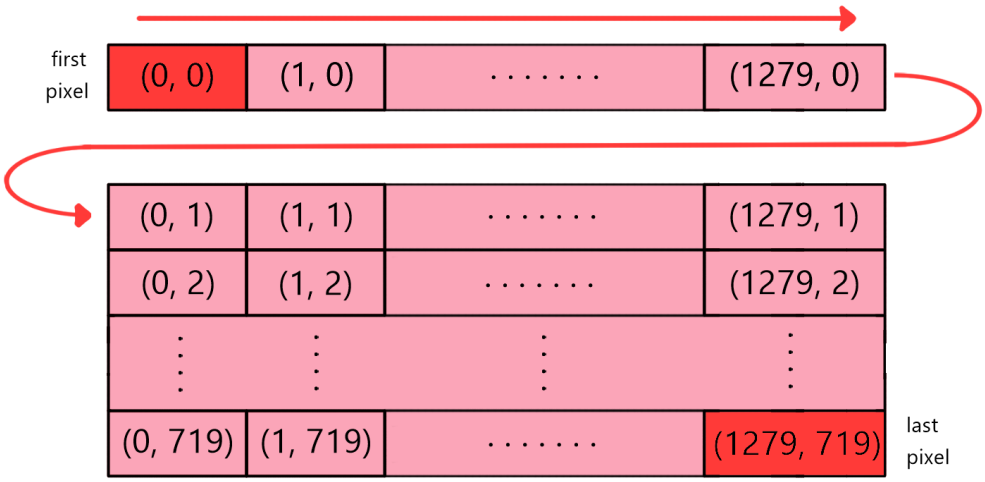


Figure 6 – Visualization of the pixel sequence in the bitstream of a 1280x720 display

Windows driver do not allow the use of floating point values or math computations that result in floating point values. Since the computations performed on the bitstream of pixels contribute directly to visual lag on the display, the perspective correction process must not be too complex. In order to meet these constraints, we will be writing an algorithm outside of the driver that will provide a mapping of source (x, y) coordinates to their perspective corrected (x, y) coordinates, which will be copied into the destination bitstream. All other pixels in the destination will be set to black. This method requires two integer arrays for the (x, y) mapping and one Boolean array to determine which pixels to set to black in the destination, thus it will add a few Megabytes of size to the driver but will allow us to perform a minimal amount of calculations.

The algorithm for generating the perspective correction mapping arrays consists of two parts: determining the outer coordinates of the perspective corrected screen to maximize rectangular size in the projection's trapezoidal output, then using linear algebra to generate the mapping matrices.

Determining Outer Coordinates

We want to maximize the size of the corrected screen within the trapezoid to give the user a better experience when interacting with the device. Since the trapezoid is wider at the bottom, the rectangle should have the same bottom edge, with side edges travelling upward until they

intercept the sides of the trapezoid. This is illustrated in Figure 7. This problem of finding the largest rectangle in a trapezoid can be solved by finding the intercept of two lines: one line representing all possible top corners of the rectangle calculated using the aspect ratio, and the other line representing one slanted side of the trapezoid. Where these two lines intersect will be the location of one of the rectangle's bottom corners. The problem set up is shown in Figure 7.

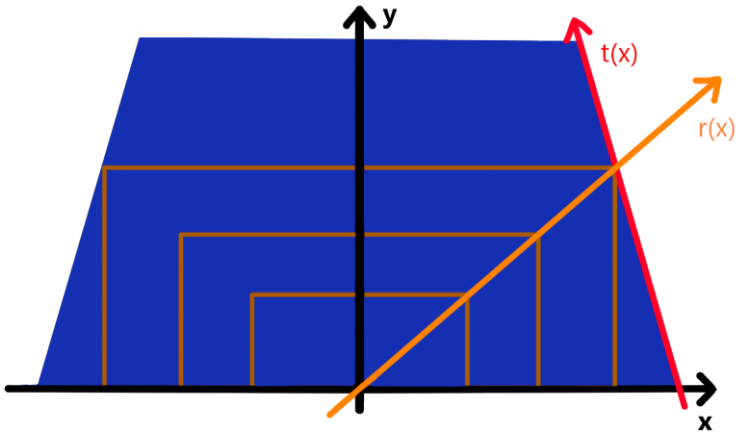


Figure 7 – Largest rectangle in a trapezoid problem represented by two lines

Since the trapezoid and rectangle are symmetric, we will focus on the left hand side of the problem to find the bottom left rectangle corner, and then mirror this to find the bottom right corner. Since the screen resolution is 1280 x 720 pixels, line 1 has the equation:

$$r(x) = y_1 = \frac{720}{1280/2}x = 1.125x \tag{1}$$

The trapezoid line has the following equation, where a is the longer side of the trapezoid, b is the shorter side of the trapezoid, and h is the height of the trapezoid:

$$t(x) = y_2 = \frac{h\left(x-\frac{a}{2}\right)}{\frac{b}{2}-\frac{a}{2}} \tag{2}$$

Since vertical scaling distortion is very small, we will assume it is negligible when mapping this coordinate onto the original screen. After solving for the intercept (x', y') of the above two lines, the x-coordinate of the screen's indented bottom left corner is simply equal to x'. The x-coordinate of the bottom right coordinate is the width minus x. Finally, the y-coordinate, y', of the corrected screen's top corners on the source screen is calculated by the following equation:

$$y' = 720 \times \frac{(h-y)}{h} \tag{3}$$

We now have the four outer coordinate mappings from the source to perspective corrected destination, as shown in Figure 8.

Figure 8 – The four corners of the windows screen and their locations after perspective correction to maximize post-correction screen size

Perspective Correction Using Homography Estimation

Homography estimation is a technique many modern computer vision projects use to perform perspective correction. One of the basic methods of homography estimation is the Discrete Linear Transform (DLT) algorithm [8]. Letting (x, y) represent a coordinate in the source screen and (u, v) represent a coordinate in the destination screen, the mapping of (x, y) to (u, v) can be described by the equation:

$$c\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{H}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{4}$$

where c is any non-zero constant and $\boldsymbol{H} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$, a 3-by-3 homography matrix describing the DLT. For one point correspondence, the above equations can be simplified into the following two equations:

$$-h_1x - h_2y - h_3 + (h_7x + h_8y + h_9)u = 0 \tag{5}$$

$$-h_4x - h_5y - h_6 + (h_7x + h_8y + h_9)v = 0 \tag{6}$$

In matrix form, these equations become:

$$A_i\boldsymbol{h} = \boldsymbol{0}, \tag{7}$$

where $A_i = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & v \end{pmatrix}$ and $\boldsymbol{h} = (\ h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6 \quad h_7 \quad h_8 \quad h_9\ )^T$. Taking our four sets of corresponding corner points, we get 4 A_i matrices which can be stacked on top of each other to form an 8x9 matrix, which can replace A_i in the above equation and can be used to solve for all the values of the homography matrix.

Once the homography matrix for our specific DLT has been determined, we can use Equation (4) to determine the (u, v) perspective corrected coordinates for every single (x, y) pixel in the uncorrected screen. The resulting u-coordinates will be placed in a 720x1280 matrix, where the row index is the corresponding x-coordinate and the column index is the corresponding y-coordinate. Another matrix for the v-coordinates can be created in the same way. Looping through every (u, v), we can also create a matrix of Boolean values that are true if a (u, v) exists for that row and column index, else they are false, denoting this pixel location on the perspective corrected screen must be set to black. Finally, in the driver, each time a call is made to copy display data from the Windows 8.1 OS to the hardware, we simply copy each pixel in the source bitstream to their perspective-corrected locations in the destination bitstream. Lastly, we set all remaining destination pixels to be black by setting all the pixel's data values to 0. The size of each pixel is given by the OS, and is stored in a variable accessible within the driver.

4.1.2 Touch Gesture Recognition Subsystem Design

In compliance with the requirements of the Touch Gesture Recognition Subsystem, we considered many different methods and technologies to track finger movements accurately and reliably. Since we are not using a physical screen for user interaction, we need a detection method capable of tracking at a distance. We initially proposed three methods to meet the above requirement: visible light detection, thermal detection, and infrared detection. Through further investigation of all three methods, we determined that the projected screen would interfere with detection by means of visible light. In addition, we discovered that accurate thermal detection requires expensive instruments beyond our budget. Therefore we adopted infrared detection for the Touch Gesture Recognition Subsystem. Infrared detection works well for our purpose since it cannot be interrupted by the projection and is relatively inexpensive in comparison to thermal detection.

Gesture Recognition Sensor Hardware

We decided to use an infrared (**IR**) detection device called the Leap Motion Controller to accomplish our goals. Instead of building custom IR detection hardware from the ground up, we purchased an off-the-shelf product to speed up our development since the Leap Motion Controller has a mature and advanced SDK. The Leap Motion Controller has a maximum detection range of 30cm. From our testing, the detection area on the projection surface is maximized when the Leap Motion Controller is at a height of 10cm and is angled to be 45 degrees below the horizontal.

Multi-touch Software Design

The Multi-touch Software operates by means of three essential algorithms: 2D Location Tracking, Touch Determination, and Windows Touch Injection. In order to develop the necessary algorithms using the Leap Motion Controller, we first need to understand how the Leap Motion Controller detects hand movements and what type of data it produces. The Leap Motion Controller has an array of IR diodes that emit IR light towards objects in front of it and detects those objects using two IR cameras. By having a fixed distance between the two cameras, the Leap Motion Controller can then calculate a 3D location for the object derived from the small discrepancy in object location seen by each IR camera. Figure 9 illustrates how the Leap Motion Controller's 3D coordinate system looks like.

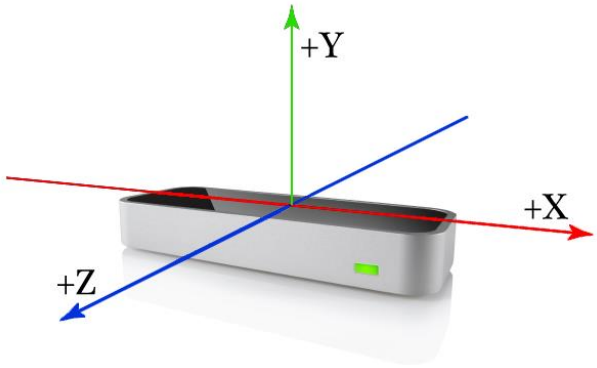


Figure 9 – Leap Motion Controller 3D coordinate system

The Leap Motion Controller is able to actively track up to ten fingertips and output a set of coordinates for each finger. Using the coordinate data, our algorithms will determine whether or not a touch has occurred, as well as the location of each touch in the Windows 8.1 OS screen coordinates.

In order to translate the positional information of the fingertips received by the Leap Motion Controller into Windows touch inputs, we will write a Windows 8.1 background application for this task. In following **Object-Oriented design** principles, we will be ensuring maintainability, reliability and quality of the background application. This application will consist of three classes: the Leap Motion Controller class, the Main class and the Listener class.

The Leap Motion Controller class is used to establish the connection between the Windows 8.1 OS and the Leap Motion Controller and will initialize the state of the Leap Motion Controller. It will also give the user a warning if the Leap Motion Controller is not connected to the device.

The Main class first creates an instance of the controller class and passes in all the necessary flags to initialize the states of the controller class. Afterwards it creates an instance of the listener class and attaches it to the controller. Once this is done, the application now is able to constantly receive data from the Leap Motion Controller, and process and performs actions through the execution of instructions inside the listener class.

The Listener class handles and processes all the data coming from the Leap Motion Controller. All functions implemented in this class will be called constantly upon receiving new positional data from the Leap Motion Controller. There are three major algorithms implemented in this class: the 2D Location Tracking Algorithm, the Touch Determination Algorithm, and Windows Touch Injection Algorithm.

2D Location Tracking Algorithm

The goal of this algorithm is to allow the system to be constantly aware of the real time 2D location of multiple fingers on or above the projection surface. This is done through the use of the Skeleton Model and the Bounding Box Detection Model provided in the Leap Motion SDK. As

shown in Figure 10, when the Leap Motion Controller is placed on a flat surface with detection side facing upward, the most sensitive detection range forms a virtual 3D box; we will refer to this as the bounding box. Upon receiving the 3D positional values of the fingers, the application creates this bounding box and normalizes all the positional values into the range of 0 to 1. It then goes through a recalibration process so that the application can adjust the bounding box size and renormalize the values according to the required projected screen size. Thus, by multiplying these normalized values with our screen dimensions, the application is able to map the position of the hand to its corresponding screen pixels in real time. In combining the Bounding Box Detection Model and Skeleton Model shown in Figure 11, the application is able to track up to 10 fingers simultaneously with very low latency.

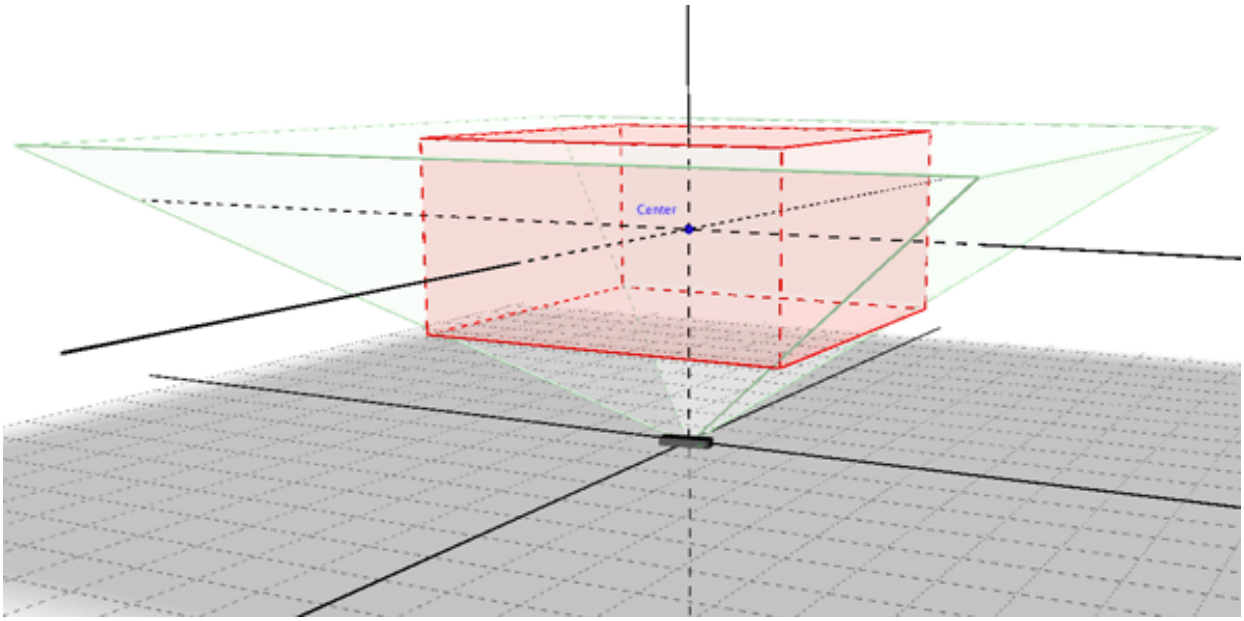


Figure 10 – Visualization of the Bounding Box Model in the Leap Motion SDK

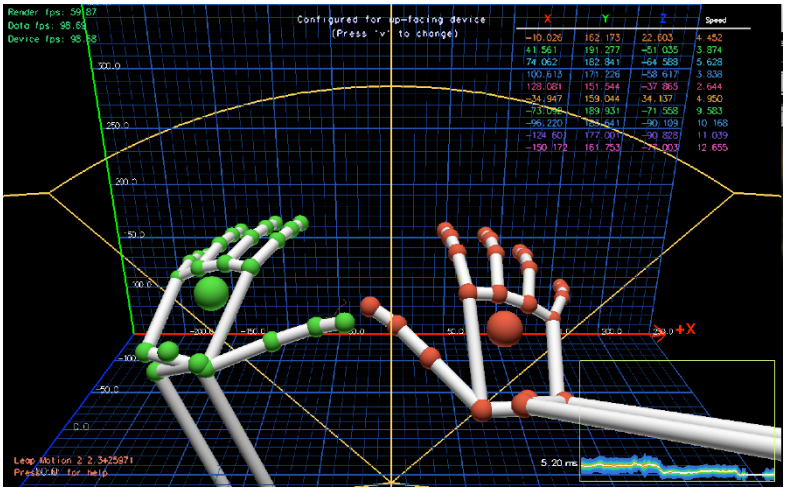


Figure 11 – Skeleton Model for Real-Time Multi-Finger Tracking

Given that the Bounding Box Detection Model provided in the Leap Motion SDK was designed to function while the Leap Motion Controller is facing vertically upward, the Bounding Box Detection Model must be changed in order to work with the Leap Motion Controller facing 45 degrees below the horizontal. With the Leap Motion Controller mounted at the 45-degree configuration with axes as shown in Figure 12, when a finger moves along the projection surface from left to right, it moves along the x-axis. When a finger moves up and down along the projection surface, it is essentially moving only along the y-axis. This is because there is no need to calculate the height of the finger when determining its corresponding 2D coordinates in the Windows 8.1 OS. Theoretically, either the y-axis or the z-axis can be used, however after experimenting with the Leap Motion Controller it was apparent that the Leap Motion Controller's z-axis was not as sensitive as the y-axis. Thus the y-axis was chosen for our algorithm.

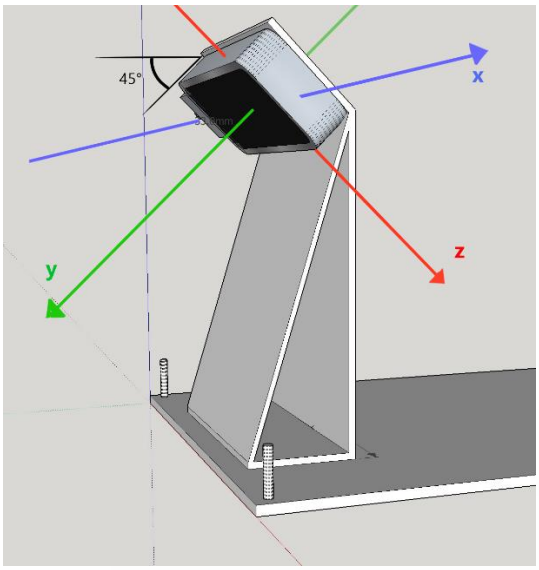


Figure 12 – 3D coordinate system of the Leap Motion Controller

In order to determine the exact coordinates of the projected screen, an initial calibration process of the Bounding Box Model is necessary. This is done by touching the four corners of the projected screen and mapping these to the four bounding corners of the Operating System's screen. The Bounding Box Model will then go through the recalibration process as previously described. Once the Bounding Box is calibrated, the application will be able to extract the hand and finger positional values from the Leap Motion Detection Frames by calling upon methods provided in the Leap Motion SDK.

For each touch event, an (x, y) coordinate value is retrieved from the Leap Motion Controller. Let the Leap Motion Controller be h cm above ground, these values must then be corrected to account for the 45-degree tilt, resulting in a new positional $(x_{corrected}, y_{corrected})$ value:

$$x_{corrected} = x \tag{8}$$

$$y_{corrected} = y * \cos\left(\frac{\pi}{4}\right) + \frac{h}{\tan\left(\frac{\pi}{4}\right)} \tag{9}$$

The four coordinates of the projected screen will be represented as follows: Let (TLx, TLy) represent the top left corner coordinate, let (TRx, TRy) be the top right corner coordinate, let (BLx, BLy) be the bottom left corner coordinate, and let (BRx, BRy) be the bottom right corner coordinate. The mapped x and y coordinates (Mx, My) for the new orientation can then be calculated by taking the previously corrected x and y values, which have values between 0 and 1, and scaling them based on the four corner coordinates:

$$Mx = \frac{(x_{corrected} - TLx)}{TRx - TLx} \tag{10}$$

$$My = \frac{(y_{corrected} - TLy)}{BLy - TLy} \tag{11}$$

Finally the algorithm will feed the constant stream of (Mx, My) into the operating system during a touch event.

Touch Determination Algorithm

The primary function of the Touch Determination algorithm is to differentiate whether a finger is touching the surface. Since the Leap Motion Controller is mounted at an angle of 45 degrees below horizontal, the controller's 3D coordinate system is also offset by 45 degrees in respect to the surface. As a result we cannot determine a touch solely on whether the fingertip crosses a specific coordinate along the z-axis. Nevertheless we figured the projection surface can be represented by a 3D plane, such that the touch action can be recognized by checking if the finger location satisfies the constructed plane equation. An illustration of how we can construct a plane is shown in Figure 13.

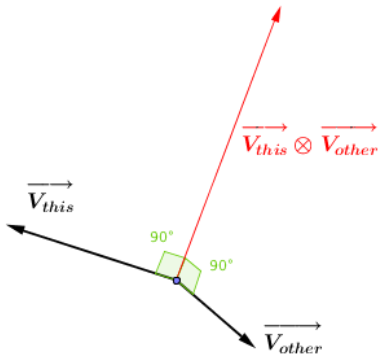


Figure 13 – Cross product illustration

This algorithm operates on the same set of positional values used by the 2D location tracking algorithm, but instead makes use of the coordinates in all three dimensions. Using three points along the projection surface, a plane equation is generated by means of the following equations:

Let $\dot{\boldsymbol{p}}_1$, $\dot{\boldsymbol{p}}_2$ and $\dot{\boldsymbol{p}}_3$ be the 3D positional values of the three points along the projection surface and $\dot{\boldsymbol{v}}_1$, $\dot{\boldsymbol{v}}_2$ be the vectors formed by the three points, such that:

$$\dot{\boldsymbol{v}}_1 = \dot{\boldsymbol{p}}_1 - \dot{\boldsymbol{p}}_2 \tag{12}$$

$$\dot{\boldsymbol{v}}_2 = \dot{\boldsymbol{p}}_1 - \dot{\boldsymbol{p}}_3 \tag{13}$$

The normal vector, $\dot{\boldsymbol{n}}$, of the corresponding plane can be calculated by taking the cross product of $\dot{\boldsymbol{v}}_1$ and $\dot{\boldsymbol{v}}_2$:

$$\dot{\boldsymbol{n}} = \dot{\boldsymbol{v}}_1 \times \dot{\boldsymbol{v}}_2, \tag{14}$$

Using the normal vector, the plane equation can be constructed by taking the dot product:

$$\dot{\boldsymbol{n}} \cdot (\dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}_1) = 0, \tag{15}$$

where $p = (x, y, z)$.

Once the plane equation is generated, continuous input of the fingertip's x, y, and z coordinates will produce a resulting product. Depending on the value of the product, the algorithm will determine whether the fingertip is below the plane, on the plane, or above the plane. As a result, we can accurately and reliably determine if a touch has occurred. Once a finger touch is determined, the function will call the Windows Touch Injection Function with the mapped x and y coordinates (Mx, My) and trigger a Windows touch event.

Windows Touch Injection Algorithm

This algorithm takes in the real-time mapped coordinates (Mx, My) as arguments once a finger touch is determined and multiplies them by the screen width, screen height, and pixel density to get the corresponding pixel area in the Windows OS. It then calls the touch injection API from the

Windows library to trigger a touch event. The touch injection API can set the touch strength, affected area of the event, number of touch, as well as the event type. Based on the functions provided with the API, we will implement the following touch gestures in our proof-of-concept model: single tap, double tap, tap and hold, and drag.

The single tap and double tap gestures can be implemented simply by calling the InjectTouchInput() method. The number of taps is equal to number of touches the application detects in a very short time interval, the default value being one second for our prototype.

The tap and hold gesture can be implemented through the usage of a timer object. When the position of the finger satisfies the plane equation for more than two seconds, the application will issue a tap and hold event flag. This flag tells the touch injection API to change into hold mode which blocks out any tap gestures until the finger is lifted. The application can then keep calling the InjectTouchInput() method in a loop until the finger is lifted up from the projection surface. Based on the occurrence of a tap and hold event, the drag gesture can be implemented by constantly grabbing the positional values from the Leap Motion Controller and updating the real-time mapped coordinates (Mx, My) inside the loop.

4.2 Physical Design

4.2.0 General Device Design

The physical design comprises of the mechanical case and the electrical components of the LumenX^3. In order to demonstrate our innovative screen projection and touch gesture recognition product features as part of a mobile computing device, we aim to make our physical design both functional and aesthetically pleasing. The physical design will have to integrate the various subsystems and components, provide a robust structural enclosure and most importantly, realize the height and placement requirements of our Pico Projector and Leap Motion Controller.

In our original vision for the LumenX^3, the entire device was meant to resemble a compact and seamless cube. The user would be able to use the device anywhere a flat surface can be found and conveniently transport it much more securely than a similar computing device with a glass screen. However, as we investigated the physical and technological constraints behind projection and touch gesture detection, the proof-of-concept design evolved to accommodate the height, positioning, logistic and opening requirements of our system. In our proof-of-concept model, the foremost priority is to demonstrate functionality; the miniaturization of the device will follow in subsequent prototypes and market ready versions.

In order to preserve functionality, the physical placement and spacing of our hardware components is critical. As explained previously, the Pico Projector must be at a height of 29 cm and projecting downward at an angle of 60 degrees to produce a useable screen of 8.5 inches diagonally and 80% viewable area in focus. In addition, the Leap Motion Controller has a maximum detection range of 30cm in which the screen must stay within to detect gestures. Through our testing, we figured the optimal detection angle to be 45 degrees below horizontal in combination with a height of 10cm. This configuration maximizes the detection area of the projected screen, guarantees detection accuracy and prevents interference between either component.

In accordance with the General Device Requirements 4.2.0, Mechanical Case Design 4.2.1 and Electrical Design 4.2.2 from our functional specifications, our physical design incorporates the following considerations. First and foremost, the mechanical case serves as the exterior point of contact with users. It has to enclose and secure all components in each of our subsystems as we are developing with off-the-shelf parts. The case will take the shape of a rectangular prism, based off on our original design, for strong structural integrity and ease of component placement.

Furthermore, to accommodate the angle and height at which the Pico Projector needs to project a suitable screen, the case must support or be compatible with a rising mechanism that can achieve the required height. The internal spacing of the device must be large enough to successfully position all components for proper functioning. Physical connections between the various hardware components as well as electrical cables for power will need to fit. In addition, the case must have openings through which the projection and hand tracking can physically occur through the case of the device.

Finally, the portability and aesthetics concerns of the physical design greatly influences design aspects like the dimensions of the device, the case material, and the operational interface presented to the user. We will be using the rapid prototyping technique of 3D printing to realize a durable and functional case with the aesthetics of a professionally produced consumer product.

The various design decisions taken into consideration and subsequent implementations for the proof-of-concept device are detailed in the following sections. All of our design and modelling are done in CAD SketchUp [9] in preparation for 3D printing, where the units of measurements are in millimeters (mm).

4.2.1 Mechanical Case Design

Overview

From our developmental testing and experiments with the Pico Projector and Leap Motion Controller, we found operational requirements that optimize the projected screen size, detection accuracy and sensor range. From the component placement and case design point of view, the requirements simplify to the following: the bottom-most of the Pico Projector must be at a height of 29 cm above the surface, and the Leap Motion Controller must be separated from the Pico Projector by 5 cm. In order to reconcile these requirements with the dimensions of the components themselves and the desire for minimization of dimensions, we implement a two-shell design that uses the case itself as the rising support for the Pico Projector.

The case is separated into two shells, where the inner shell can be raised up to our desired height interpedently from the outer shell. Therefore two states are possible with our device: a portable, Compact Mode with half the height of the higher Projection Mode. The user will be able to switch to the Compact Mode when the device is not being used for maximum portability. These two functional modes are illustrated in the following images.

Figure 14 – Mechanical Case in Projection Mode

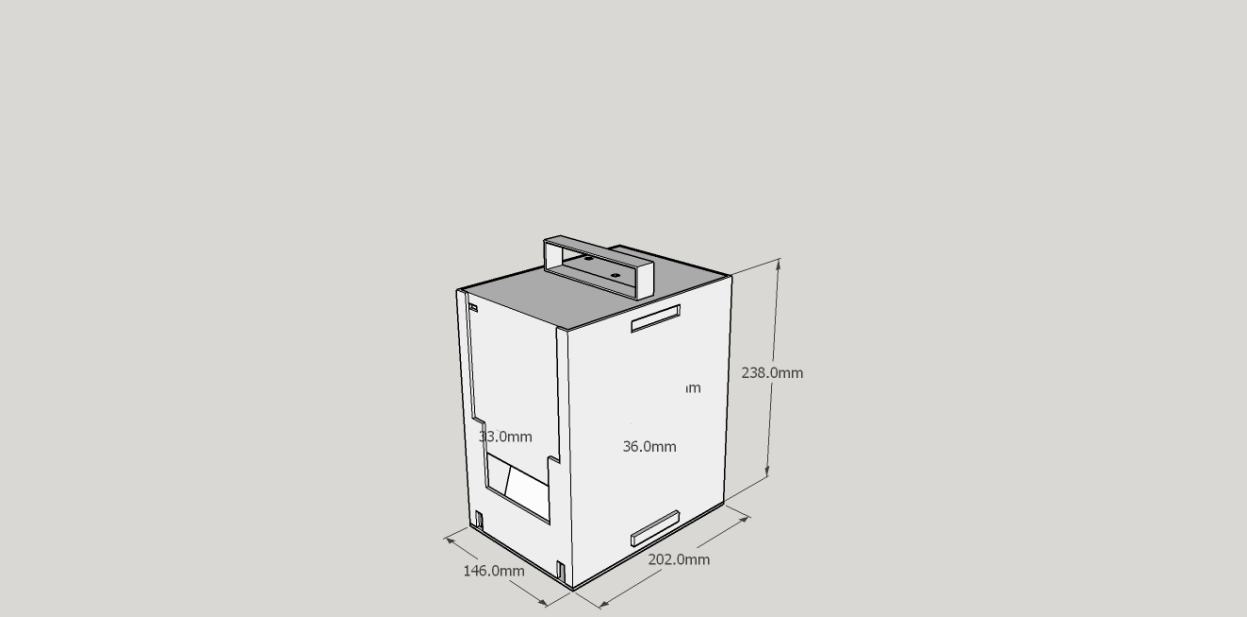


Figure 15 – Mechanical Case in Compact Mode

As seen from the models, our device is anticipated to have dimensions: width of 14.6 cm, length of 20.2 cm, and a height of 23.8 cm in its portable state and 44.6 cm in its projection mode. The entire case will be printed as three separate pieces for easier construction: the outer shell, the inner shell and the base. In addition, stands for the Pico Projector and the Leap Motion Controller, the handle on top and the locking mechanism will be printed separately and attached at the end.

A render of the case with the out shell moved to the side and compared with the inner shell is shown in Figure 16. The shells each have a wall thickness of 3 mm for optimized 3D printing stability and strength.

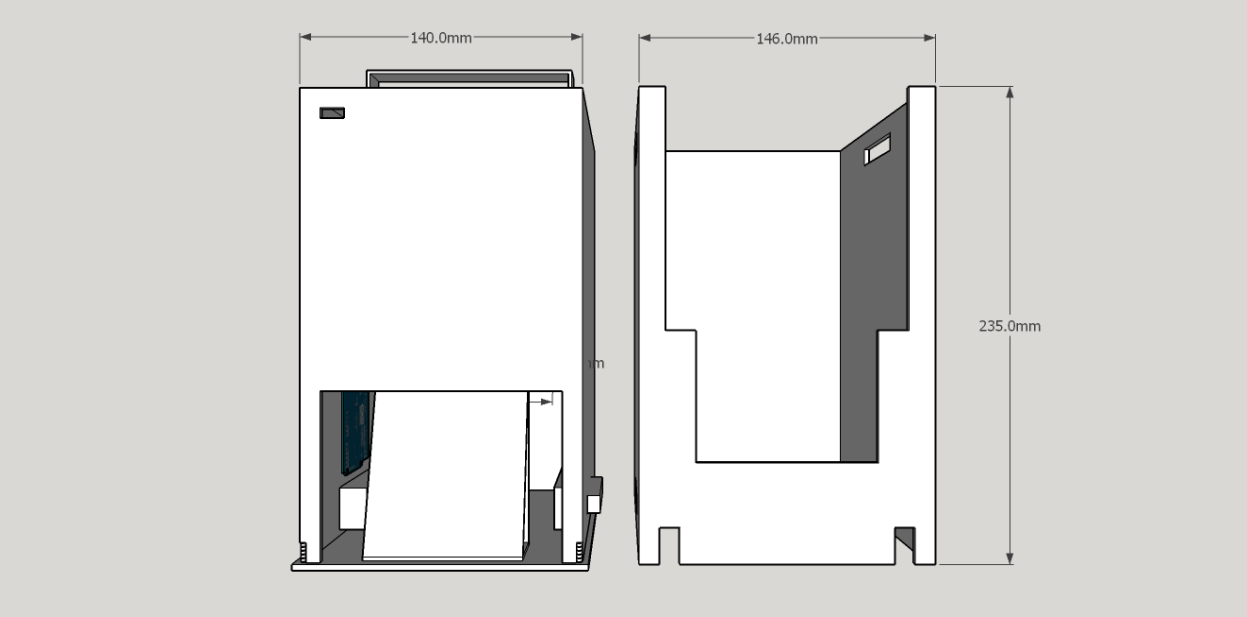


Figure 16 – Comparison of two shells

Component Placement

With the capability of the CAD software we are able to model our components to their exact specifications and have incorporated their placements within the device with our models. We have also considered the addition of cables and USB connections to and from the hardware components. This is represented as rectangular prisms connecting to the main bodies of the components. The isometric view, as shown in Figure 17, gives a broad view of all the major components including the Pico Projector, the Leap Motion Controller, the MeegoPad T01 and the Arduino Uno. All components are shown in the portable state of the device, allowing room for the interconnections of USB's and cables, as well as electrical adapters and the USB hub.

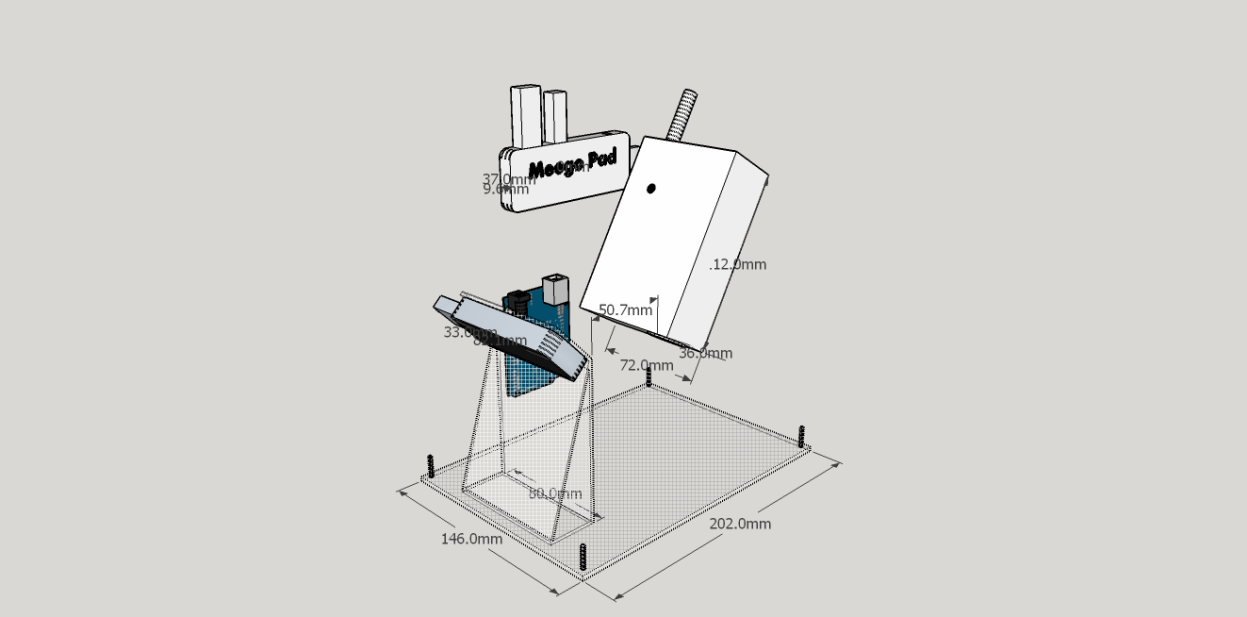


Figure 17 – Isometric view of internal component placement

The interior component placement also satisfies the requirement of the 5 cm spacing between the Pico Projector and the Leap Motion Controller, as seen in the parallel projection right rendering following, as well as the 29 cm height requirement of the projector, seen in the following Figures 18 and 19.

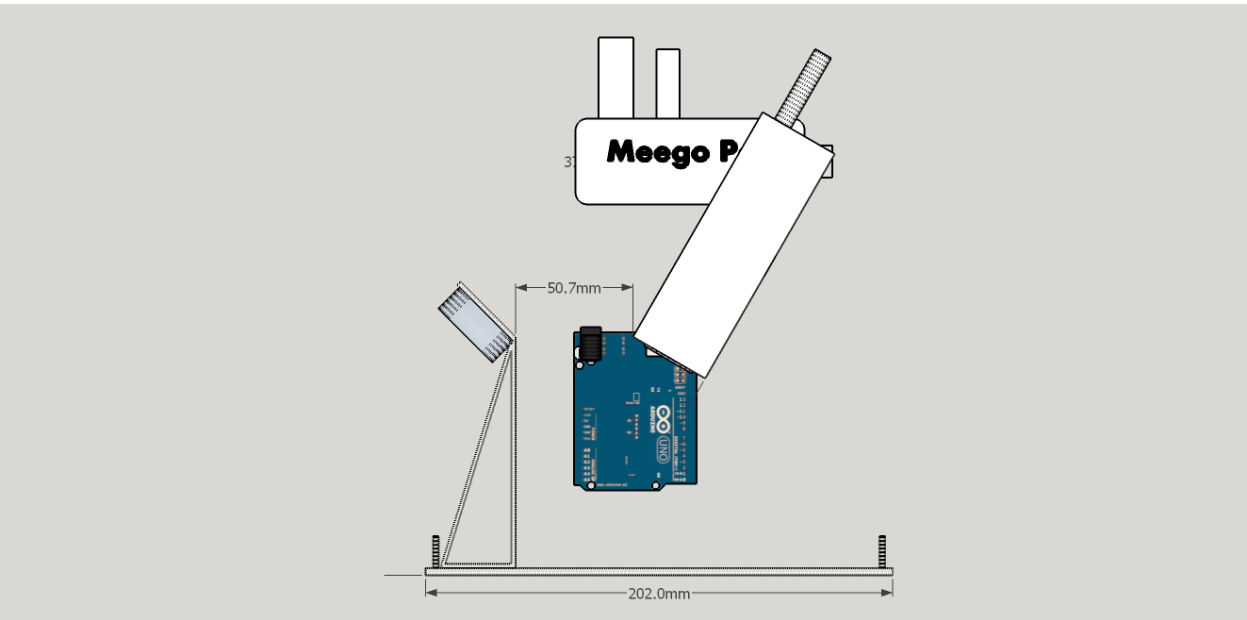


Figure 18 – Realization of 5 cm separation between Leap Motion Controller and Pico Projector

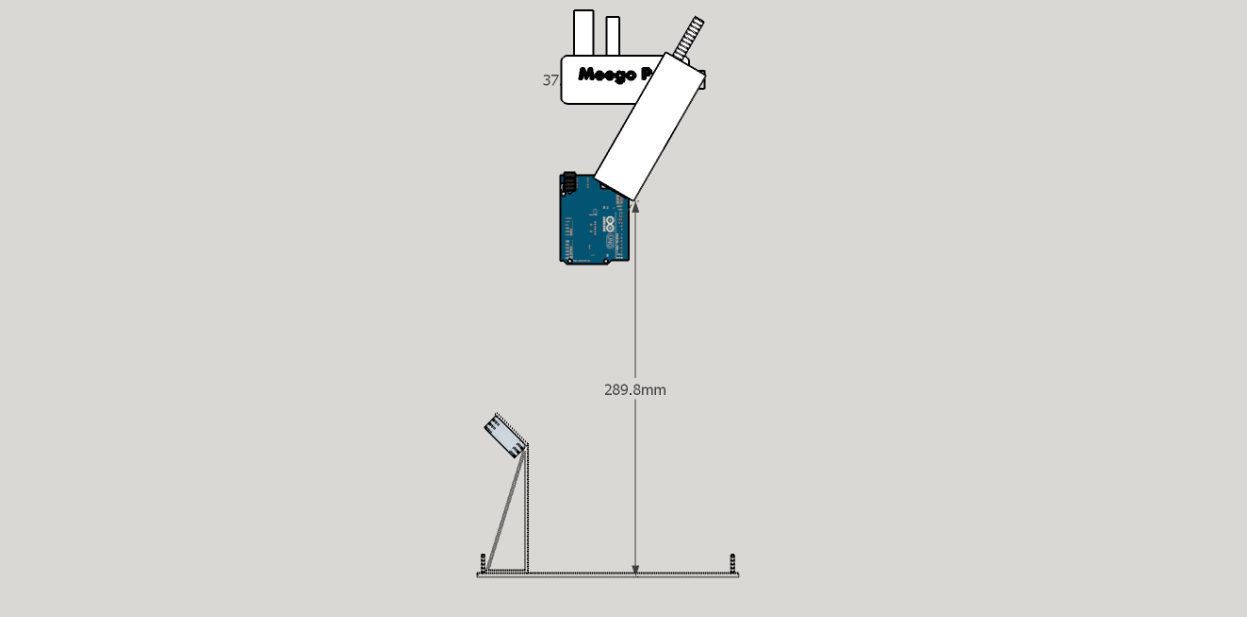


Figure 19 – Realization of 29 cm height requirement of Pico Projector

The Leap Motion Controller will be attached to a stand to the base via a triangular cylindrical stand, offering the 45 degrees and 10 cm height requirements for the Controller's positioning. Similarly, the Pico Projector is anchored to another triangular attachment from the top of the inner shell, providing a 60 degrees tilt. This attachment will also contain openings for access to the power button of the projector.

The Pico Projector, the Leap Motion Controller, and the Arduino Uno will all be attached to the inner wall of the inner shell at the top, and move with the height changes initiated by the user. The power adapters, extension power bard, USB hub and additional wiring and cables will remain at the bottom of the case, securely attached to the base. An opening for the electrical outlet cord and plug is at the back of the device. These placements are seen in Figure 20 below.

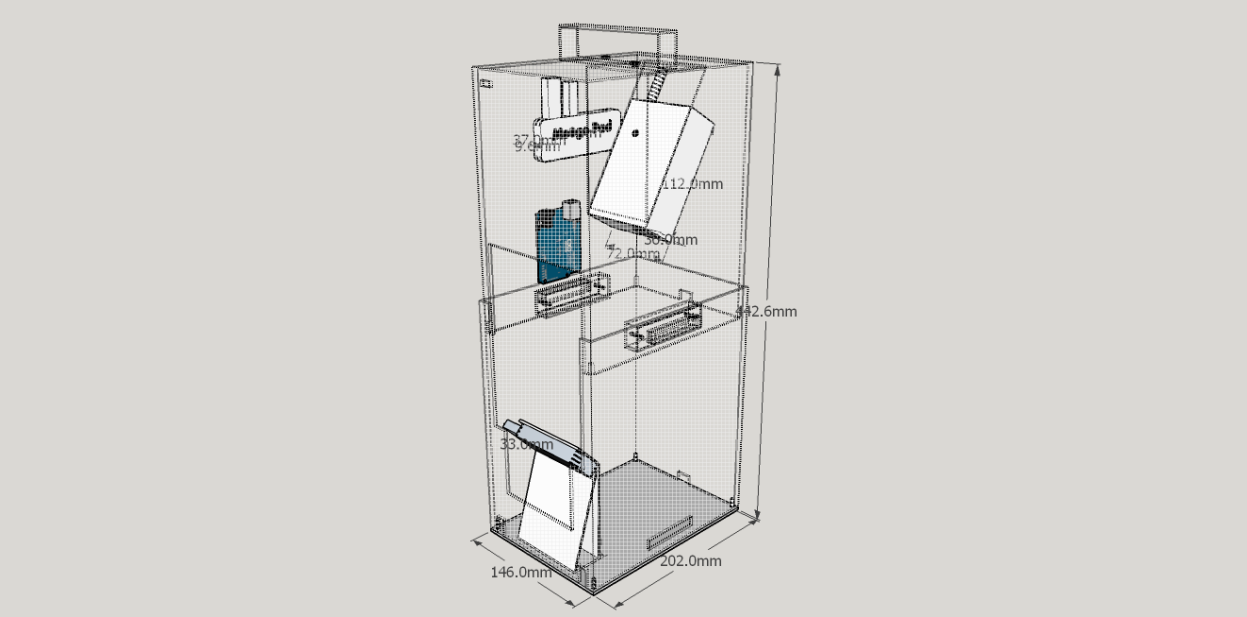


Figure 20 – Internal placement of components in Projection Mode

Case Function Realization

To facilitate the capability to change height as required by the case design, the two shells will be separated by a minimum distance allowing the inner shell to slide up and down. The user can use the handle on top of the inner shell to pull it up. In order for the case to remain stable in either of its two states, a locking mechanism comprised of a spring-loaded button is available for the user to activate at the bottom and top of the device. The buttons are the rectangular protrusions seen on the bottom right and left sides of the case, seen in Figure 21.

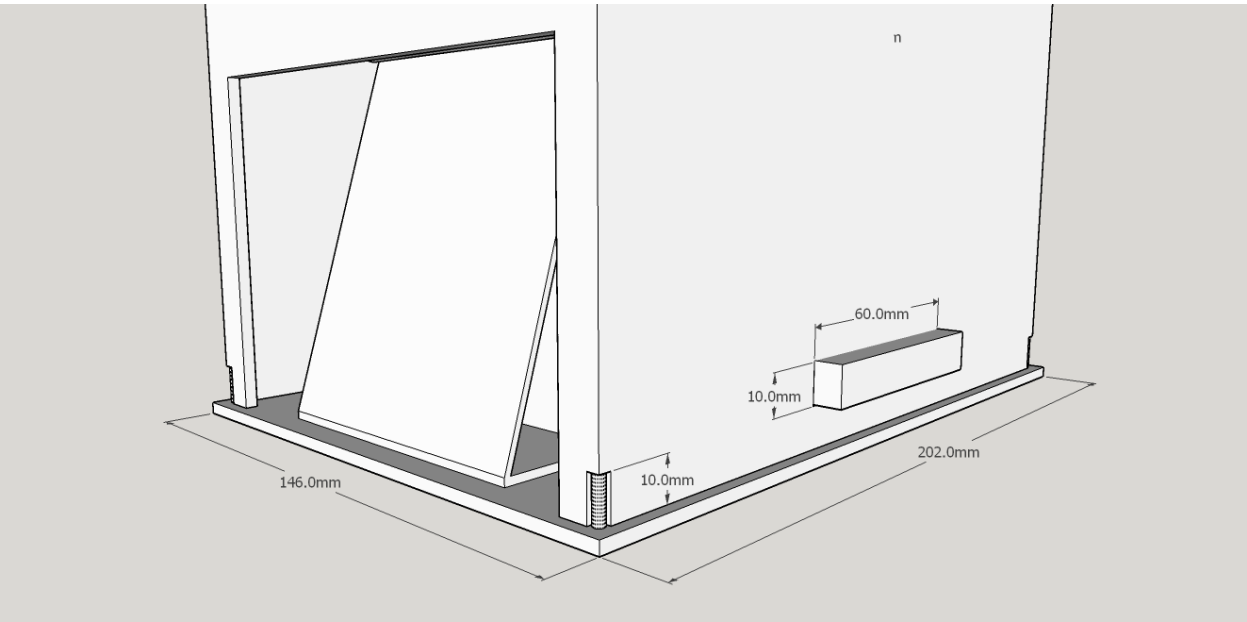


Figure 21 – Bottom view of inner shell with corner springs and button

From the Compact Mode, the user pushes in the two buttons into the device, past the outer shell, causing the inner shell to pop up due to additionally loaded springs on the bottom four corners. The springs initiate the rising mechanism without user intervention. The buttons will no longer protrude from the openings in the outer wall, and instead move along the inner shell as the inner shell becomes free to move. A handle attachment on the top of the inner shell provides and physical extension by which the user can lift the inner shell. The handle is designed to accommodate most user hand sizes and attaches to middle for stable rising actuation. Once the user reaches the height of the Projection Mode, the aligned button will pop out again from the openings at the top of the out shell, locking the two shells together once more.

The locking mechanism was designed with stability and ease of use in mind. With a width of 6 cm and height of 1 cm, the button is meant to act as a physical bar preventing rotation and pivoting of the two shells, especially in the projection mode. The spring-loaded push operation improves user experience by presenting an intuitive and familiar operation found in everyday life. Compared with other methods, the push button operation reduces the number of user steps as well as securely stabilizes the mechanical locking. Close up views of the button and the well that it sits in, with resting and pushed-in state, are shown below, respectively.

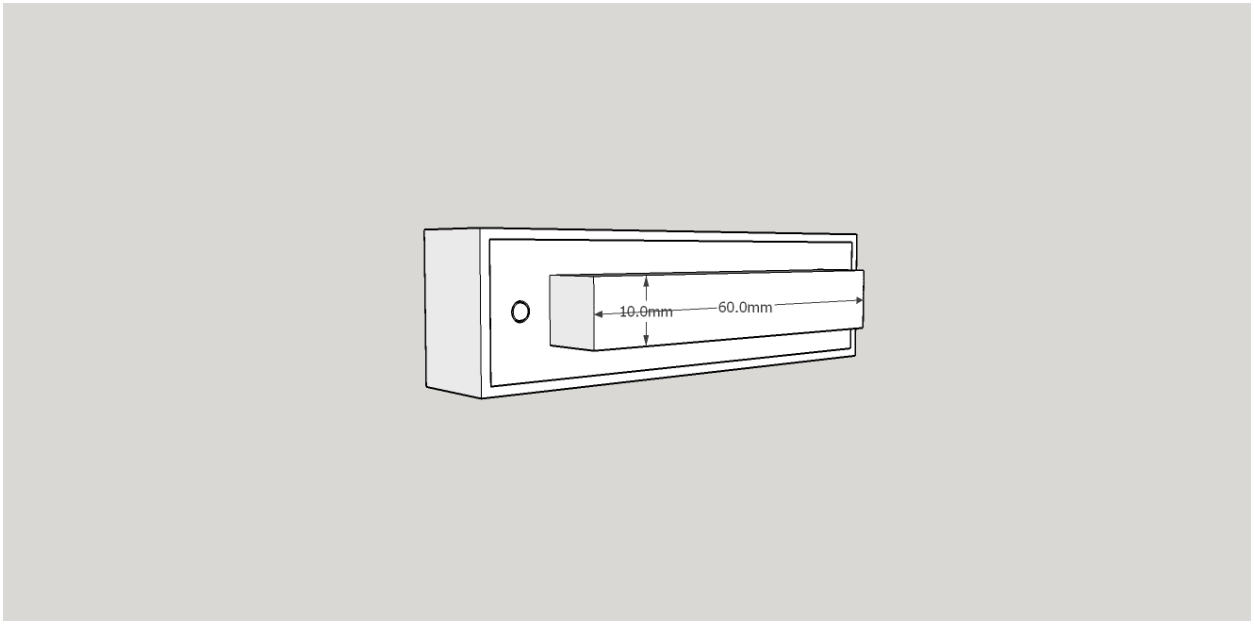


Figure 22 – Button in extended configuration

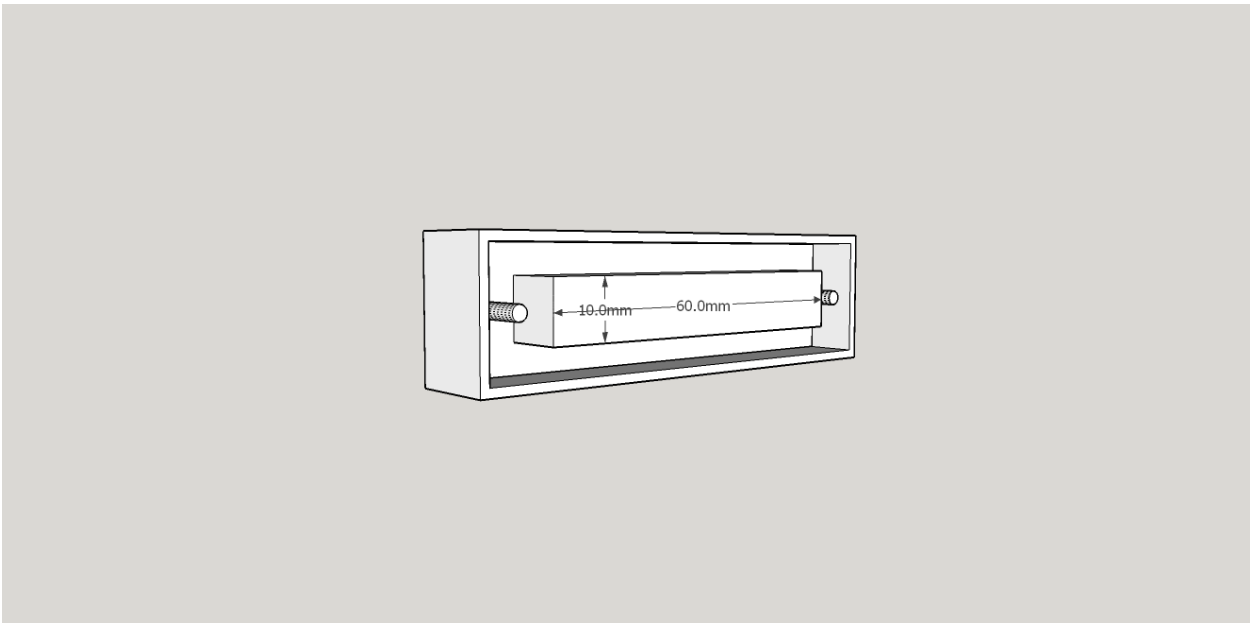


Figure 23 – Button in compressed configuration

The user can interact with the casing and rising mechanism by pushing in the two buttons as depicted in Figure 24, then pulling up the inner shell as in Figure 25.

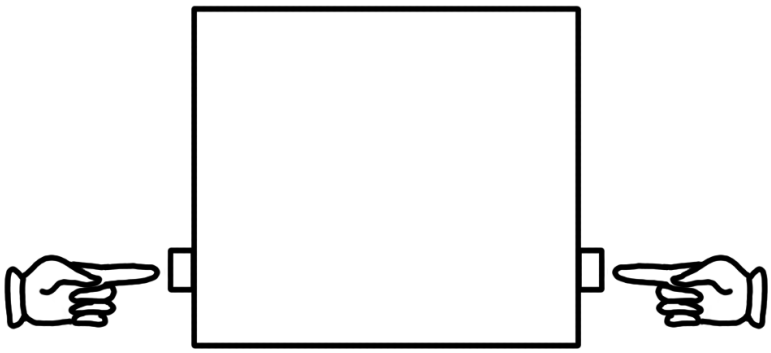


Figure 24 – Visualization of button press by user

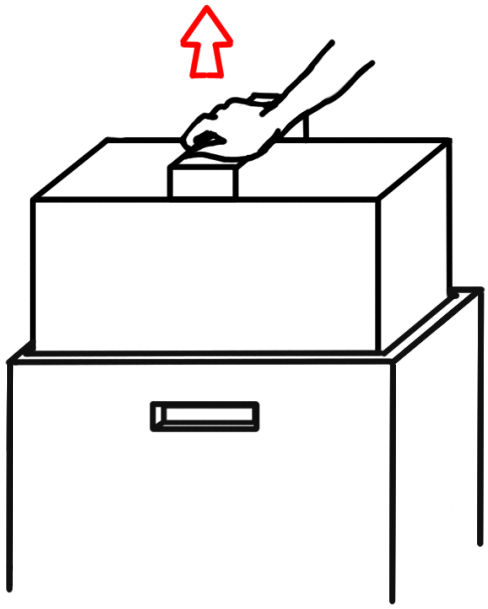


Figure 25 – Visualization of user pulling up the inner shell

Due to the nature of optical projection and infrared sensing, openings have to be made on the case for the projection and detection to occur without blockage. The two-shell design maximizes the area of exposure in the projection mode, while minimizing the openings in the portable state. In additional, in order to provide the user with USB ports, two openings are present near the bottom of the device The frontal view of the device giving the dimensions of the openings in the projected state are shown next.

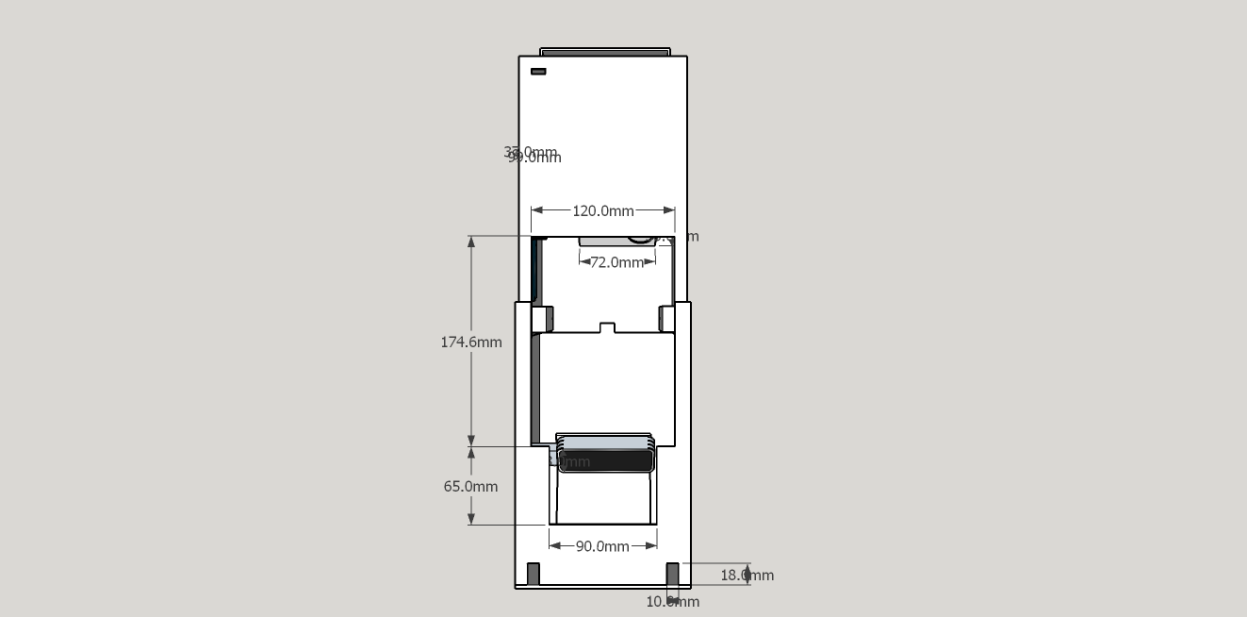


Figure 26 – Front view of openings in Projection Mode

Additionally, openings for access to internal components and openings for System Status Indicator LED's are present near the top left of the inner shell. The user requires access to the power buttons of the Pico Projector and the MeegoPad T01, which will be facilitated through a physical extension to the outside of the top of the case. The two circular openings seen behind the handle serves as the access points. The LED openings are located on the top left corner on the front side of the inner shell and provide through-holes for the light indication of system status. A close-up of the top inner shell is shown below.

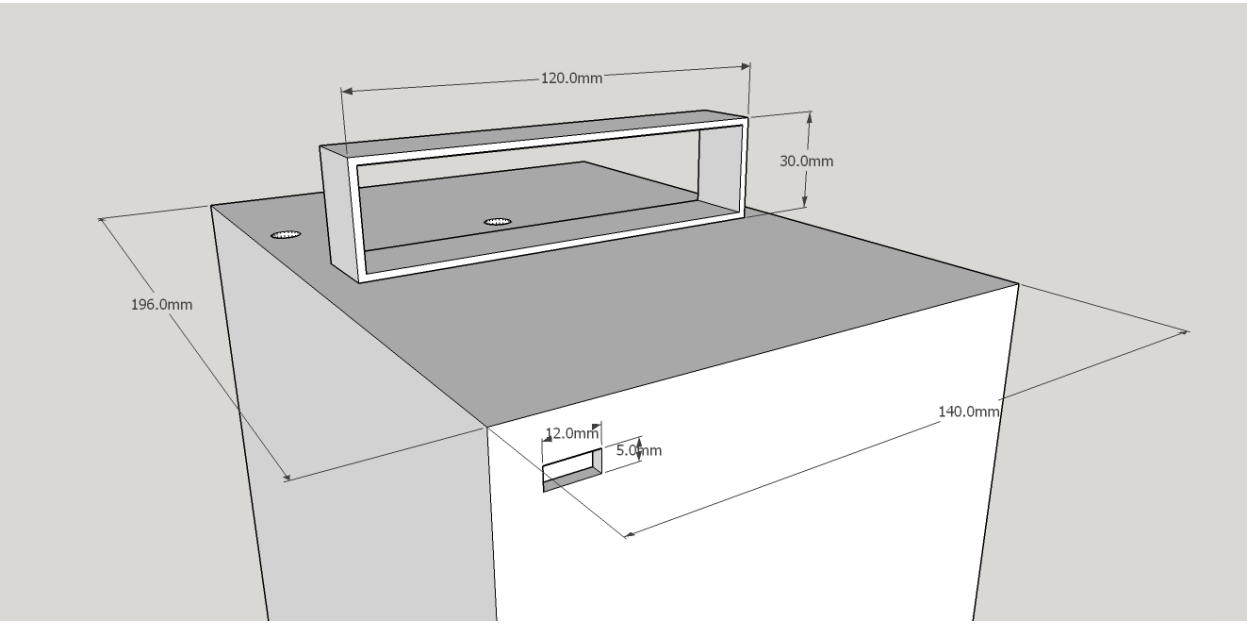


Figure 27 – Top view of inner shell showing openings to power controls and LED's

Other Designs

Various other designs that satisfy the height and placement requirements of our device also exist in terms of the rising support, the locking mechanism and the case material. To facilitate a height increase, support poles connecting the Pico Projector can be attached to a base. This approach may offer a more stable height increase without the need for locking, but does not completely enclose the internal components and exposes wiring or cables. Locking mechanisms using a pull-out hook can be easier to implement. However, as we found out in our cardboard mock-up, a single pivot point does not guarantee the stability required for projection. A scissor lifting mechanism offers the best accuracy and stability, but relies on separate moving parts and takes up critical internal spacing needed by other components. In terms of case material, we have also considered using cardboard, paper mache and modelling plywood should 3D printing prove to be too difficult, impractical or expansive to use. However, as 3D printing most commonly use plastic as its printing material, the strength and rigidity achieved with plastic combined with the accuracy afforded by 3D printing makes it an ideal rapid prototyping process.

4.2.2 Electrical Design

The electrical design for our proof-of-concept device mainly concerns the powering of each of our subsystems and hardware components. The main priority is to interconnect the Pico Projector, the Leap Motion Controller, the MeegoPad T01 and the Arduino Uno within the confines of our case and power the entire system using a single unified power bar.

Both the Pico Projector and the MeegoPad T01 requires its own specific AC-DC adapter for power. These will be plugged into a mini extension power bar with a single cable outlet. The user can then use the single extension cord to plug-in the device, fulfilling the requirements for using the North American standard off wall 100-240V at 50-60 Hz AC. The USB hub used to provide additional USB ports will provide power in addition to facilitating communication for the Arduino Uno. The LED's required for the System Status Indicator will directly obtain its power from the Arduino Uno 5V and GND ports.

5.0 Test Plan

General Testing

The following are a list of tests we will be running on the LumenX^3 device to ensure proper functionality and compliance with all IEEE standards we are following ([7], [10], [11], and [12]).

Test Case 01 – Lifting the Mechanical Case from Compact Mode to Projection Mode		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 on a flat and stable surface, push the buttons on the left and right sides of the box simultaneously	- Inner box rises slightly - Side buttons do not pop back out	
2. Pull the handle at the top of the LumenX^3 upwards	- Inner box slides upward with little effort and does not tilt in any direction	
3. Continue raising the handle until the device locks into its taller Projection Mode	- Inner box slides upward until at the Projection Mode height (projector should be 29cm above the surface) - Once at Projection Mode height, the side buttons automatically pop out from the upper holes	
4. Push or pull the handle upwards or downwards	- Device should not move at all and should be stable - Inner box should not slide - Side buttons should not pop inward	

Test Case 02 – Compressing the Mechanical Case from Projection to Compact Mode		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 on a flat and stable surface, push the buttons on the left and right sides of the box simultaneously	- Inner box slides down gracefully, but does not reach Compact Mode height - Side buttons do not pop back out of the case	
2. Push the handle at the top of the box down until the device locks into Compact Mode	- Inner box slides downward until at Compact Mode height, some resistance felt but still quite easy to lock into place - Once at Compact Mode height, the side buttons automatically pop out from the lower holes	

3. Push or pull the handle upwards or downwards	- Device should not move at all and should be stable - Inner box should not slide - Side buttons should not pop inward	

Test Case 03 – Powering on the LumenX^3		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. Plug the power plug of the LumenX^3 into a power source	- LED light on LumenX^3 turns on, and shines a red colour, indicating the computer is off	
2. Raise the device to Projection Mode	- Device locks into projection mode (similar to Test Case 01 results)	
3. Press the projector's power button on the top of the LumenX^3	- Projector turns on and a blank projection is seen on the tabletop surface	
4. Press the computer's power button on the top of the LumenX^3	- LED light on LumenX^3 turns from red to green light - MeegoPad T01 computer turns on, and the Windows Operating System screen is seen in the projection	

Test Case 04 – Powering off the LumenX^3 (Method 1 – Software Shut Down)		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, turn off the computer from within the Operating System software	- LED light on LumenX^3 turns from green to red light, indicating the computer is off - Projection shows Windows 8.1 OS shutting down, and eventually becomes a blank projection screen	
2. Press the projector's power button on the top of the LumenX^3	- Projection turns off completely	
3. Transform the LumenX^3 from Projection Mode to Compact Mode	- Device locks into Compact Mode (similar to Test Case 02 results)	
4. Unplug the power cord from the power source	- LED light turns off, no light emitted from the LumenX^3	

Test Case 05 – Powering off the LumenX^3 (Method 2 – Force Shut Down]		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, press and hold the computer's power button on the top of the LumenX^3 for 8 seconds	- LED light on LumenX^3 turns from green to red light after 8 seconds, indicating the computer is off - Projection immediately changes from projecting Windows to a blank projection after 8 seconds	
2. Press the projector's power button on the top of the LumenX^3	- Projection turns off completely	
3. Transform the LumenX^3 from Projection Mode to Compact Mode	- Device locks into Compact Mode (similar to Test Case 02 results)	
4. Unplug the power cord from the power source	- LED light turns off, no light emitted from the LumenX^3	

Test Case 06 – Case Stability		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. Plug the power plug of the LumenX^3 into a power source	- LED light on LumenX^3 turns on, and shines a red colour, indicating the computer is off	
2. Tilt the LumenX^3 from side to side	- All inner components should remain in their place and all connections should remain intact - Inner case and buttons should remain in their same positions	
3. Raise the device to Projection Mode	- Device locks into projection mode (results as in Test Case 01) - Inside the case, all inner components are still in their original locations	
4. Press the projector's power button on the top of the LumenX^3	- Projector turns on and a blank projection is seen on the tabletop surface	
5. Press the computer's power button on the top of the LumenX^3	- LED light on LumenX^3 turns from red to green light - MeegoPad T01 computer turns on, and the Windows Operating System screen is seen in the projection	

6. Tilt the LumenX^3 from side to side	- All components should remain in their place - Inner and outer case should not move respective to each other - Buttons remain locked in the upper holes	
7. Use the LumenX^3 like a typical content consumer	- Verify that every component has remained the same - Projection has not have moved or tilted and is still projecting the computer screen in correct proportions - Touch remains as accurate as it was before tilting the device - Windows 8.1 OS continues to be fully functional	
8. Turn off the computer from within the Operating System software	- LED light on LumenX^3 turns from green to red light, indicating the computer is off - Projection shows Windows 8.1 OS shutting down, and eventually becomes a blank projection screen	
9. Press the projector's power button on the top of the LumenX^3	- Projection turns off completely	
10. Transform the LumenX^3 from Projection Mode to Compact Mode	- Device locks into Compact Mode with the same amount of ease as in Test Case 02 results - No cables or components have displaced, interfering with the compression of the case	
11. Unplug the power cord from the power source	- LED light turns off, no light emitted from the LumenX^3	

In the following Projection and Touch Test Plans, each set of tests will assume that the other subsystems have passed their own tests and are working as intended.

Projection Testing:

Test Case 07 – General Projection Functionality Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, use the LumenX^3 as if the projection was a capacitative touchscreen of a Windows 8.1 computer	- The projected computer screen is always proportionate, with the rest of the projection always a black colour - No part of the computer screen is cut off in the projection - Corrected computer screen within the projection is a constant rectangle of size 18.53cm by 10.10cm - Projected Windows 8.1 OS screen has extremely minimal delay never rising beyond 0.1 seconds	

Test Case 08 – Projection Resolution Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, open the Screen Resolution page in the Control Panel of the Windows 8.1 OS and view available resolutions	- Only 1280 x 720 is shown as the sole available resolution, in Landscape Orientation, on Display: 1. MStar Projector	

Test Case 09 – Projection Clarity Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, open a text editor and display words of size 9, 10, and 11 font	- Size 9 font words and larger are easily readable without straining the eyes	

Test Case 10 – Projection Accuracy Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, open a text editor, bring up the on-screen keyboard and type words at 20wpm	- All letters tapped by the user are accurately shown to be typed onto the screen in the correct order	

Test Case 11 – Projection Stress Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, open a text editor, bring up the on-screen keyboard and type words at 20wpm	- All letters tapped by the user are accurately shown to be typed onto the screen in the correct order	

Touch Test Plan

Test Case 12 – Touch Performance Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, tap on a large Windows desktop icon	- Large Windows desktop icon is easily pressed without much precise effort by the user	
2. Tap on a large Windows taskbar program	- Large Windows taskbar program is easily pressed without much precise effort by the user	
3. Open a text editor, bring up the on-screen keyboard and type words at 20wpm	- No characters are dropped, all letters correctly tapped by the user are accurately shown on the screen in their correct order	

Test Case 13 – Tap Gesture Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, tap the centre of the screen	- Tap gesture visibly seen in the centre of the Windows 8.1 OS screen in the projection, and the OS reacts to the tap event	
2. Tap each of the four corners of the projection	- All four tap gestures visibly seen at their respective corners of the Windows 8.1 OS screen in the projection and the OS reacts to each tap event	

Test Case 14 – Double Tap Gesture Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, double tap the centre of the screen	- Double tap gesture visibly seen in the centre of the Windows 8.1 OS screen in the projection, and the OS reacts to the tap event	
2. Double tap each of the four corners of the projection	- All four double tap gestures visibly seen at their respective corners of the Windows 8.1 OS screen in the projection and the OS reacts to each double tap event	

Test Case 15 – Tap and Hold Gesture Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, tap and hold the centre of the screen	- Tap and hold gesture visibly seen in the centre of the Windows 8.1 OS screen in the projection - OS reacts to the tap and hold event by registering it as a right click	
2. Double tap each of the four corners of the projection	- All four tap and hold gestures visibly seen at their respective corners of the Windows 8.1 OS screen in the projection - OS reacts to each double tap event as right click events	

Test Case 16 – Drag Gesture Test		
Actions/Steps	**Expected Result**	**Test Result (Pass/Fail)**
1. With the LumenX^3 powered on and in its Projection Mode, drag a finger across the screen	- Drag gesture visibly seen on the Windows 8.1 OS screen in the projection - OS reacts to the drag event	
2. Drag a Windows desktop icon to the right side of the screen and continue dragging finger into the black projection border	- Drag gesture registered by OS and Windows desktop icon moves to the right, letting go once the finger has reached the border of the screen	
3. Drag a finger from outside the right side of the screen in toward the centre of the screen	- OS begins registering the drag gesture once the finger reaches the border of the screen - Windows 8.1 OS reacts to this gesture by displaying the Charms Bar on the right side of the screen	

6.0 Conclusion

The LumenX^3 is a new addition to a consumer's smart device portfolio. It is designed with collaboration and portability in mind by utilizing projection instead of having a physical screen. Consequently, we must also employ a fingertip tracking method by means of infrared cameras. As a result of all these innovative approaches to redefine how a user can interact with a smart device, more hardware space is now available for greater computing power meanwhile keeping the LumenX^3 as a portable device one can bring anywhere.

The design specification clearly defines solutions to meet the functional specification of the LumenX^3. Development of the device has taken place in two distinct phases, modular development and system integration. To ensure the product quality, a testing procedure will be employed in multiple stages in compliance with the IEEE testing standard. The development of the proof-of-concept model is in the integration stages while complying with all proof-of-concept requirements outlined in the functional specification document (marked with P1). The anticipated delivery date of the proof of concept model will be the end of April 2015.

Works Cited

[1] C. Tang, H. Mak, H. W. Ng, D. Mok and G. Yu, "Design Specification – LumenX3: Projected Mobile Computer," 2015.

[2] "MeeGoPad T01 Microsoft Windows 8.1 OS TV Stick - Quad-Core CPU, 2GB RAM, 32GB Internal Memory, Bluetooth, HDMI Interface (White)," Q. C. Factories, 2015. [Online]. Available: http://www.amazon.com/MeeGoPad-T01-Microsoft-Windows-Stick/dp/B00RVCGNEC. [Accessed 20 January 2015].

[3] Arduino, "Arduino Uno," 2015. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardUno. [Accessed 22 January 2015].

[4] AAXA TECHNOLOGIES INC., "P3 Pico Projector," 2015. [Online]. Available: http://www.aaxatech.com/products/p3_pico_projector.htm. [Accessed 23 January 2015].

[5] Leap Motion, Inc, "Leap Motion Controller," 2014. [Online]. Available: https://www.leapmotion.com/. [Accessed 10 January 2015].

[6] Microsoft Corporation, "Windows 8.1 tutorial," [Online]. Available: http://windows.microsoft.com/en-ca/windows/tutorial. [Accessed 20 January 2015].

[7] IEEE Standards Association, "1621-2004 - IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments," 2004. [Online]. Available: http://standards.ieee.org/findstds/standard/1621-2004.html. [Accessed 28 January 2015].

[8] E. Dubrofsky, "Homography Estimation," THE UNIVERSITY OF BRITISH COLUMBIA, Vancouver, 2009.

[9] SketchUp, Trimble Navigation Limited, 2013. [Online]. Available: http://www.sketchup.com/. [Accessed 2 March 2015].

[10] IEEE Standards Association, "29148-2011 - Systems and software engineering -- Life cycle processes -- Requirements engineering," 2011. [Online]. Available: http://standards.ieee.org/findstds/standard/29148-2011.html. [Accessed 28 January 2015].

[11] IEEE Standards Association, "730-2014 - IEEE Standard for Software Quality Assurance Processes," 2014. [Online]. Available: http://standards.ieee.org/findstds/standard/730-2014.html. [Accessed 28 January 2015].

[12] IEEE Standards Association, "829-2008 - IEEE Standard for Software and System Test Documentation," 2008. [Online]. Available: http://standards.ieee.org/findstds/standard/829-2008.html. [Accessed 28 January 2015].