Projected Mobile Computer

LumenX³

Version 1.1.1

May 5, 2015

Team Members:

Carmen Tang
Davin Mok
Gary (Guo) Yu
Herman Mak
Him Wai (Michael) Ng

OBELXTECH

Abstract

This paper 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.

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; and finally, all hardware components enclosed within a rigid shell while keeping overall device size to a minimum to maximize portability. Hardware choices and motivations behind our design choices will be justified by examining a set of specific system performance requirements.

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.

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.

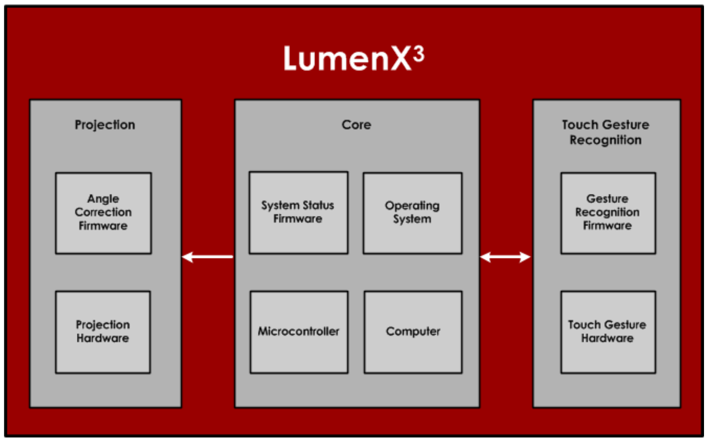

Figure 1 – Subsystem layout for the LumenX^3 system

The 3 subsystems will operate using the following hardware components:

- Core subsystem: MeegoPad T01 [1], Arduino Uno [2], and LED's
- Projection subsystem: AAXA P3 Pico Projector [3]
- Touch Gesture subsystem: Leap Motion Controller [4]

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 [5] 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.

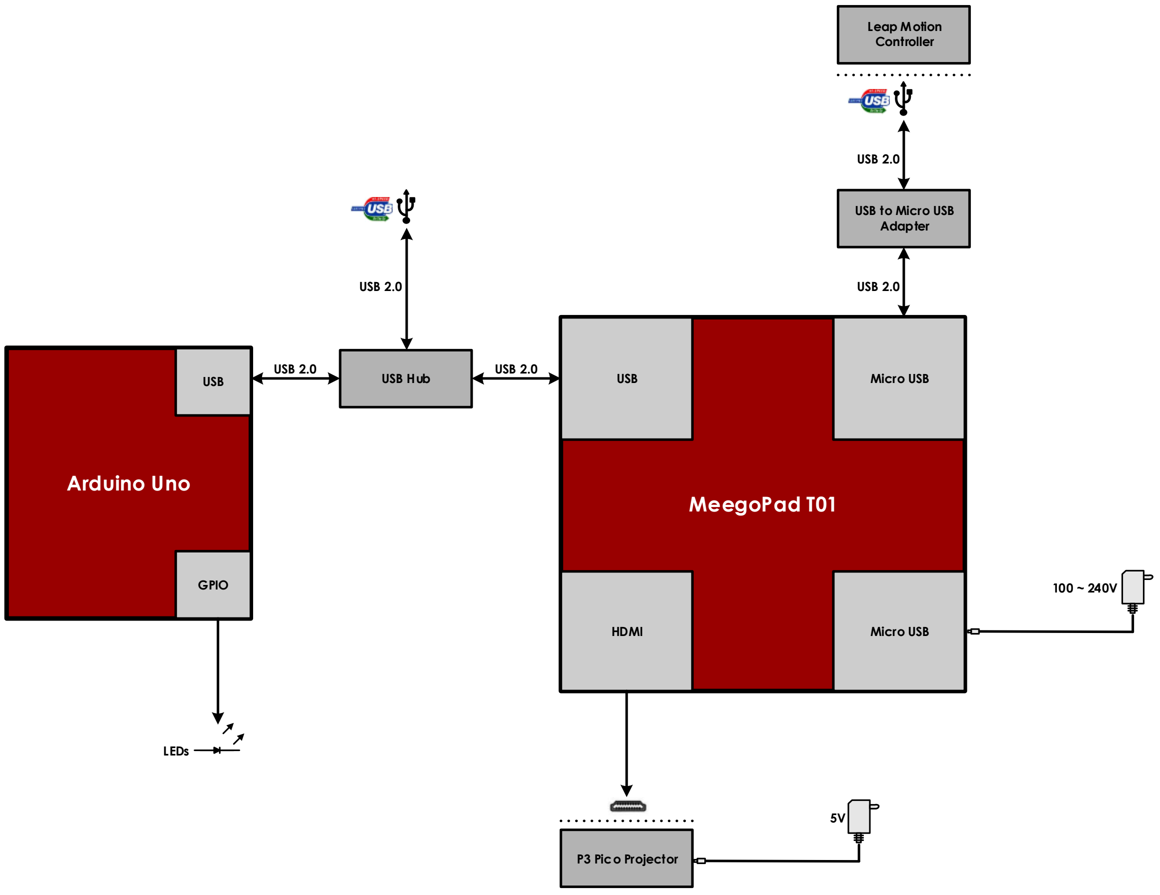

Figure 2 – Block diagram for the LumenX^3 system

3.0 Product Design

3.1 System Design

3.1.0 Core Subsystem Design

The purpose of the Core Subsystem is to facilitate the interaction between itself and the several other key subsystems of the LumanX^3. It is comprised of a Meegopad T01 PC-On-A-Stick that utilizes the latest Intel Atom "Bay Trail" chip. The Quad-Core processor runs at 1.33GHz with 2GB of DDR3L RAM. It is capable of running a full Windows 8.1 OS and directly communicates to the Status Indicator over COM3 serial port. The USB port is connected to a USB Hub which interacts with the Leap Motion controller and a HDMI output to the projector.

Figure 3 – Meegopad T01 PC-On-A-Stick [1]

Figure 4 – Arduino UNO R3 Microcontroller [2]

The Status Indicator consists of an Arduino UNO R3 microcontroller and several LEDs. The Arduino UNO uses a ATmega328 chip that runs at 16MHz. The board offers 14 digital I/O pins 6 of which have a PWM capability and 32kb of memory for storing code. 3 LEDs, red yellow and green, are connected to their own respective digital I/O pin and to the common GND pin provided by the Arduino. Depending on the command sent over serial by the Meegopad T01 it will set one LED to be On (HIGH) while the others to be Off (LOW).

3.1.1 Projection Subsystem Design

Since we are severely restricted on size, we opt for a small pico projector that provides enough brightness and a high resolution. The AAXA P3 Pico Projector is our projector of choice, boasting more than High-Definition (HD) 720p resolution and 50 Lumens of brightness yet is still small enough to fit in the palm of one's hand. The HDMI port works perfectly with the MeegoPad T01.

Placing the pico projector at a height of 29cm and projecting downward at an angle of 60 degrees produces a screen that is mostly in focus (over 80%) and has a screen size large enough for comfortable portable use, about 10" diagonally. The higher the projector is moved, the larger the screen and the larger the case must be. Lowering the height and decreasing the angle can make large screens, however it loses a lot of focus. The Leap Motion Controller also has a finite range in which the screen must stay within to detect gestures.

Since the projector is at an angle, the resulting screen appears as a trapezoid. 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.

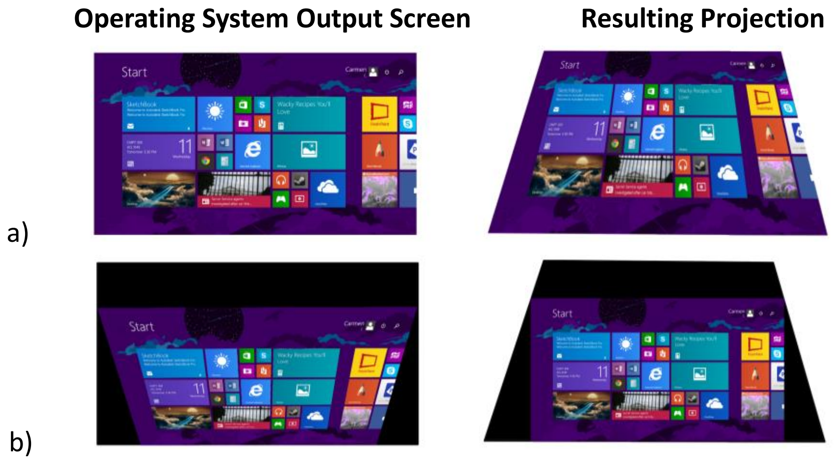

Figure 5 – Operating System screens before and after ideal perspective correction and their resulting trapezoidal projections

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. 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 write an algorithm to generate the perspective correction mapping. It consists of two parts: determining the outer coordinates of the perspective corrected screen to maximize rectangular size in the projection's trapezoidal output, and 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. This problem of finding the largest rectangle in a trapezoid can be solved by finding the intercept of two lines, as shown in Figure 6.

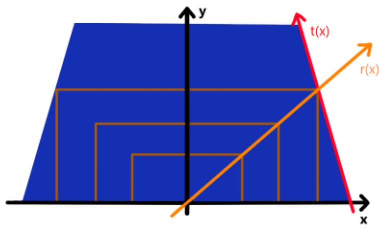

Figure 6 – Largest rectangle in a trapezoid problem represented by two lines

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 [6]. 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:

$$c\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{H}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \text{ where } \boldsymbol{H} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \tag{1}$$

where c is any non-zero constant. This can be simplified into:

$$-h_1x - h_2y - h_3 + (h_7x + h_8y + h_9)u = 0 \tag{2}$$

$$-h_4x - h_5y - h_6 + (h_7x + h_8y + h_9)v = 0 \tag{3}$$

In matrix form, these equations become:

$$A_i\boldsymbol{h} = \boldsymbol{0}, \text{ where } A_i = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & v \end{pmatrix} \tag{4}$$

And $\boldsymbol{h} = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{pmatrix}^T$. 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 replaces A_i in the Equation (4) 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 used Equation (4) to determine the (u, v) perspective corrected coordinates for every single (x, y) pixel in the uncorrected screen. 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.

▶ 3.1.2 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. As a result, we decided on using an infrared detection device called the Leap Motion Controller (Figure 7) to accomplish our goals. 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. Software is then developed according to the Object-Oriented design principles to utilize its capabilities.

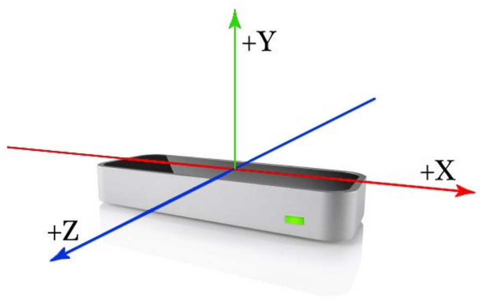

Figure 7 – Leap Motion Controller 3D coordinate system

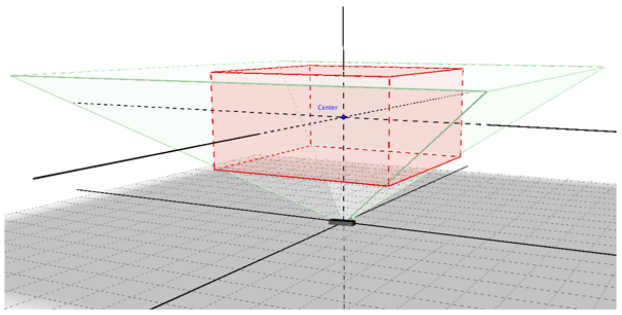

Figure 8 – Visualization of the Bounding Box Model

Multi-touch Software Design

The Multi-touch Software operates by means of three essential algorithms: 2D Location Tracking, Touch Determination, and Windows Touch Injection.

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. As shown in Figure 8, upon receiving the 3D positional values of the fingers, the application creates a 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.

With the Leap Motion Controller mounted at the 45-degree configuration with axes as shown in Figure 9, 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 moving only along the y-axis and z-axis. It is apparent that the Leap Motion Controller's z-axis is not as sensitive as the y-axis, thus the y-axis is chosen for our algorithm.

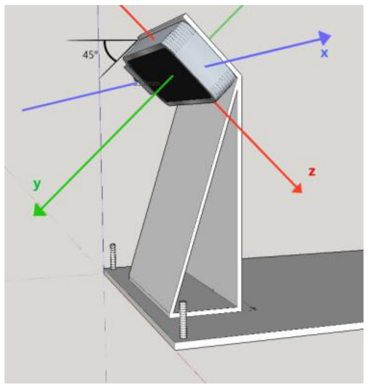

Figure 9 – 3D coordinate system of the Leap Motion Controller

In order to determine the exact coordinates of the projected screen, an initial calibration process of the Bounding Box Model is necessary. For each touch event, a (x, y) coordinate value is retrieved from the Leap Motion Controller. Let the Leap Motion Controller be h cm above ground, these values are then = corrected to account for the 45-degree tilt, resulting in a new positional $(x_{corrected}, y_{corrected})$ value:

$$x_{corrected} = x \tag{5}$$

$$y_{corrected} = y * \cos\left(\frac{\pi}{4}\right) + \frac{h}{\tan\left(\frac{\pi}{4}\right)} \tag{6}$$

These corrected values will be normalized to the range of 0 and 1 in relative to the coordinates of the 4 corners of the projected screen. Finally the algorithm will feed the constant stream of coordinates 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. 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. We can construct a plane according to equations in Figure 10.

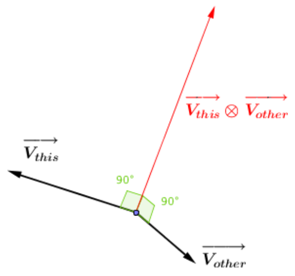
$\overrightarrow{V_{this}} \otimes \overrightarrow{V_{other}}$
$\overrightarrow{V_{this}}$
90°
90°
$\overrightarrow{V_{other}}$

Figure 10 – 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. The continuous input of the fingertip's x, y, and z coordinates will then 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 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 and trigger a Windows touch event.

Windows Touch Injection Algorithm

This algorithm takes in the real-time mapped coordinates 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, the following touch gestures are implemented in our proof-of-concept model: single tap and drag.

▶ 3.2 Physical Design

▶ 3.2.0 General Device Design

The physical design comprises of the mechanical case of the LumenX[3], which integrates the various subsystems and components, provides a robust structural enclosure and most importantly, realizes the height and placement requirements of our Pico Projector and Leap Motion Controller. All of our design and modelling are done in CAD SketchUp [7] in preparation for 3D printing, where the units of measurements are in millimeters (mm).

▶ 3.2.1 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. The requirements are: 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 realize these requirements, 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. Two states are possible with our device: a portable, Compact Mode with half the height of the higher Projection Mode. The user can switch to the Compact Mode when the device is not being used.

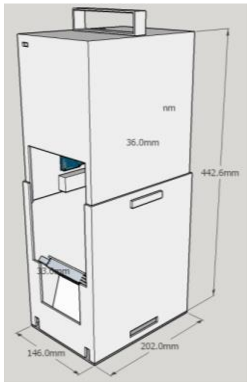

Figure 11 – Mechanical Case in Projection Mode

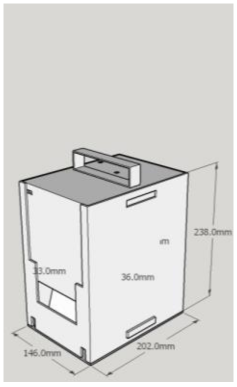

Figure 12 – Mechanical Case in Compact Mode

Case Function Realization

To facilitate the capability to change height as required by the case design, the two shells are 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 13.

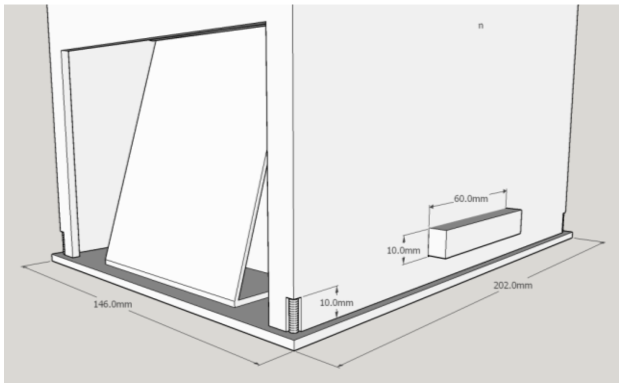

Figure 13 – Bottom view of inner shell with corner springs and button

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, and 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 in Figures 14 and 15, respectively.

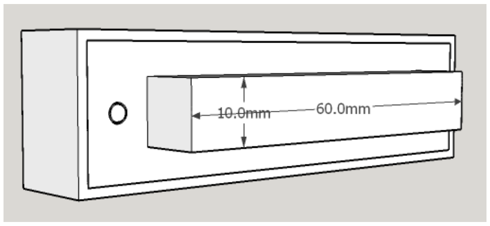

Figure 14 – Button in extended configuration

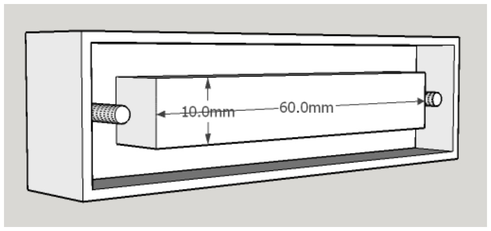

Figure 15 – Button in compressed configuration

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.

Additionally, openings for access to internal components and openings for System Status Indicator LED's are present on 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.

4.0 IEEE Standards Applied

ISO/IEC/IEEE 29148-2011: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering [8]

The LumenX^3 is developed through the use of a Software Requirements Specification document that we have produced, titled "Functional Specification – LumenX^3: Projected Mobile Computer". Within the functional specification document we have outlined both hardware and software requirements for our device. These specifications have been used as guidelines for features and developmental goals throughout the project.

▶ IEEE 829-2008: IEEE Standard for Software and System Test Documentation [9]

The LumenX^3 is tested through the use of a System Test Document that we have produced. The document is used to effectively test the required functionalities outlined in our functional specifications document.

▶ IEEE 730-2014: IEEE Standard for Software Quality Assurance Processes [10]

The LumenX^3's software system is developed in accordance to the IEEE standard for software quality assurance processes. Unit tests are written for different modules and the teams that are responsible for the implementation of different software modules are in constant communication on the progress and dependencies between each module.

▶ IEEE 1621-2004: IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments [11]

The LumenX^3 is designed in accordance to the IEEE standard in our Status Indicator as part of our Core Subsystem. Three colored LEDs are always visible on the exterior of the LumenX^3 to reduce system status ambiguity, increase power savings, and improve overall interaction of the device.

5.0 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 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.

This paper clearly defines design specification and 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, and all completed within our original schedule of the Capstone project semester. To ensure the product quality, all design considerations are compliant with various IEEE standards.

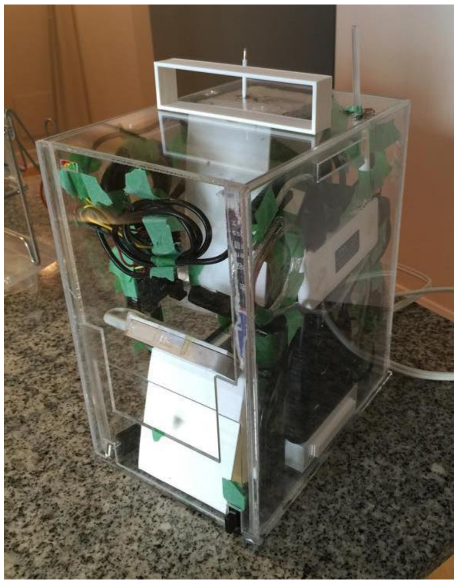

Figure 16 – Final prototype model of LumenX^3

Works Cited

[1] "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].

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

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

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

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

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

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

[8] 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].

[9] 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].

[10] 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].

[11] 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].