NavEYEgation Technologies

November 12, 2015

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

RE: ENSC 440W Capstone Project Design Specification for the Gaze-controlled Mouse

Dear Dr. Rawicz,

The attached document is a Design Specification for NavEYEgation Technologies' Gaze-controlled Mouse System. This system will allow the user to operate a mouse using only their eyes without the need for costly eye-tracking hardware.

This document details design specifications only for the proof-of-concept and prototype models. It will also describe features to be included in the production model, however these features will not be implemented.

NavEYEgation Technologies is comprised of five dedicated 4th and 5th year engineering students: Ramin Nazarali, Nathan Samchek, Sunny Chowdhury, Mani Ahuja, and myself. If you have any questions or comments please feel free to contact me at msantiag@sfu.ca.

Sincerely,

Monica Santiago
CEO, NavEYEgation Technologies

Enclosure: *Design Specification for the Gaze-controlled Mouse*

NavEYEgation Technologies

Design Specification for the *Gaze-controlled Mouse*

Project Team: Ramin Nazarali
Monica Santiago
Nathan Samchek
Mani Ahuja
Sunny Chowdhury

Contact Person: Monica Santiago
msantiag@sfu.ca

Submitted To: Dr. Andrew Rawicz – ENSC 440W
Steve Whitmore – ENSC 305W
School of Engineering Science
Simon Fraser University

Issued date: November 12, 2015

Abstract

The Design Specification for the Gaze-controlled Mouse describes the design of our prototype model. As such, only design specifications which meet functional requirements of priority I and II will be described as specified in our Functional Specification document: *Functional Specification for the Gaze-controlled Mouse.* [1]

This document will describe reasoning for our design choices for the prototype model. Features to be implemented in the production model will also be described but not detailed at length. The hardware component of our system consists of a small infrared camera, which will accompany a Single Board Computer (SBC). These will be mounted on the host computer and will read the motion of the pupil/iris. The software component consists of instructions and image processing which will be done using a SBC and an application on the user's personal computer. The SBC will determine the user`s point of gaze in screen coordinates and detect indicators corresponding to mouse events and send these to the user`s PC. An application running on the user`s PC would then project the mouse cursor on the screen accordingly and allow them to click and drag, as they would with a regular mouse.

The software program process for both the gaze tracking algorithm and that required to implement cursor movements and mouse events will be described in detail in this document. This document will also include a test plan for our system and its submodules.

Table of Contents

List of Figures

List of Tables

Glossary

Corner Vector (CV) – The vector between the corners of the eye

Degree of Freedom (DOF) - a direction in which independent motion can occur

Effective Resolution – The number of pixels from the camera's video image that contain the user's face and eyes

Eye Vector (EV) – The vector from the inner eye corner to the centre of the iris

Haar Cascade – A type of classifier, typically used for finding facial features, that uses many Haar-like features combined in a classifier cascade

HSV Channel – Hue Saturation Value Channel

IEEE – Institute of Electrical and Electronics Engineers

IEC – International Electrotechnical Commission

ISO – International Organization for Standardization

LBP Cascade – A type of classifier similar to a Haar Cascade that uses local binary patterns combined in a classifier cascade

Mouse Event – Actions used interact with the computer via a pointing device (e.g. left click, right click, double left click, drag, scroll)

Odroid (Model: C1+) – Single Board Computer designed by HardKernel

OpenCV – An image processing library

Point-of-Gaze – Point on the screen where the user is looking

1. Introduction

A gaze-controlled mouse is a system that uses the movement and state of the user's eyes to interact with a personal computer. The system is made up of software as well as hardware components to ease the integration and enhance the user experience. It will use the position of the pupils relative to the rest of the eye to correctly map the cursor on a computer. Additionally, it will use a combination of blinks, dwell time, or other methods to interact with the user's computer.

The system is primarily designed for individuals with musculoskeletal, inflammation, or motor degeneration issues such as Parkinson's disease, arthritis, and carpal tunnel syndrome where the use of hands is painful or impossible. The design specifications for this gaze-controlled mouse will be detailed in this document, as proposed by NavEYEgation Technologies.

1.1 Scope

This document describes how the design of the Gaze-controlled Mouse meets the functional specifications listed in *Functional Specification for the Gaze-controlled Mouse* [1]. It includes design specifications required to meet the prototype I (proof of concept) and prototype II requirements. As such, only design choices required to meet functional requirements of priority I and II will be described in detail.

1.2 Intended Audience

This document will be used by all team members of NavEYEgation Technologies during the design and testing phases of development. During the design phase, it will be used as a guide throughout the implementation process. During the testing phases, it will be used along with the function specification to verify that these goals have been met and that the restrictions have been followed.

2. System Overview

A single infrared camera is mounted next to the output screen which is intended to be controlled through gaze-tracking as shown in Figure 1.

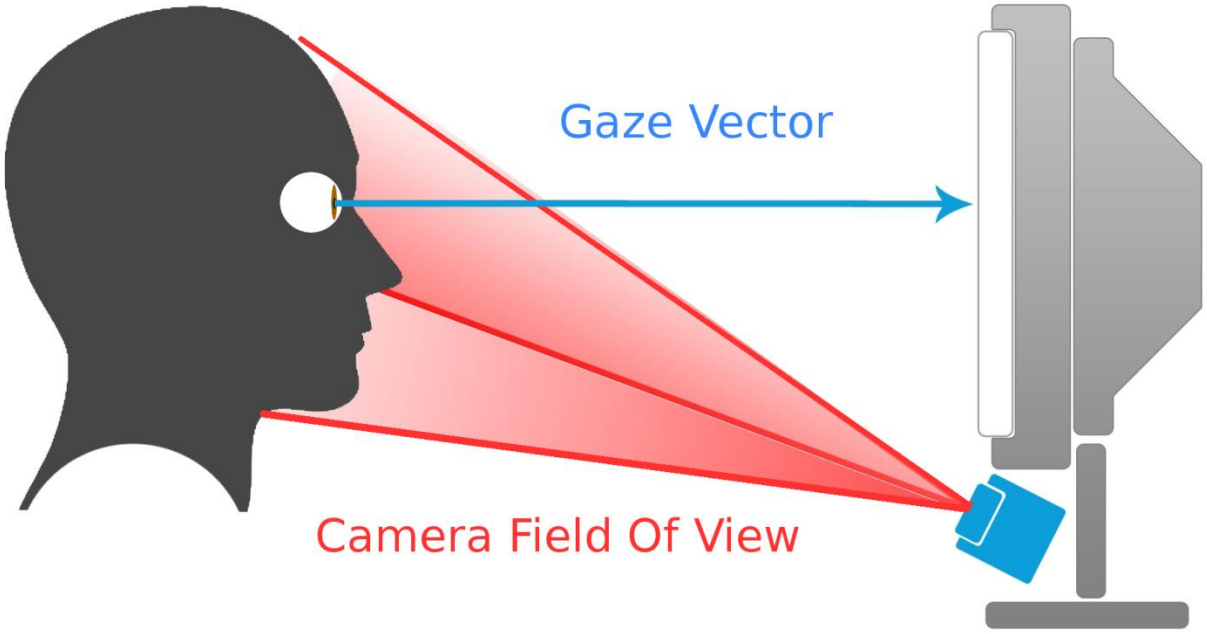

Figure 1: Gaze-Controlled Mouse System

The user's full face will be visible within the mounted camera's field of view and the input from this camera will be fed through a software image processing algorithm on a Single Board Computer (SBC) shown in Figure 2 to determine the user's point of gaze.

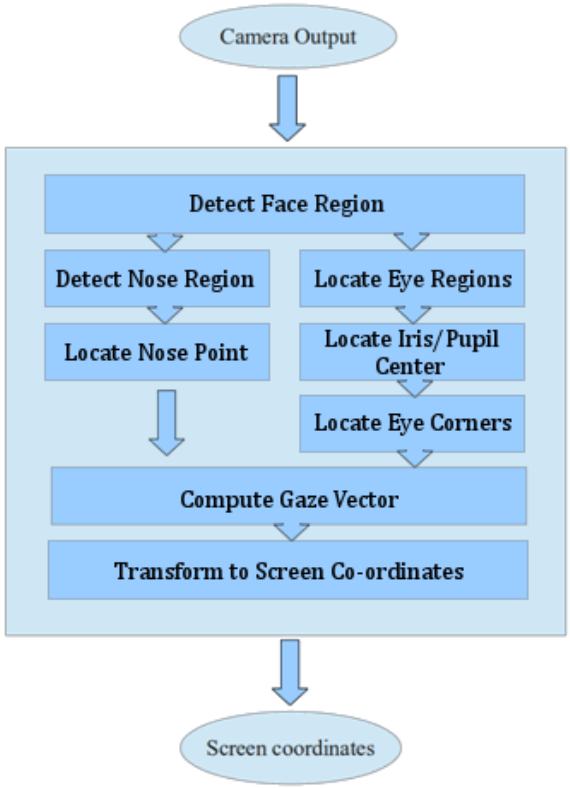

Figure 2: Algorithm for determining point of gaze on screen

Upon using the gaze-tracking solution for the first time, a one-time calibration step must be performed. This calibration step would consist of looking at several sequentially displayed points on the screen the user is intending to use. The gaze of the user during this step would be recorded and used to establish the screen's position relative to the camera.

The screen coordinates provided as output by the algorithm in Figure 2 would be sent to an application running on the user`s PC would then project the mouse cursor on the screen accordingly and allow them to click and drag, as they would with a regular mouse. For example blinking both eyes twice can translate to a single click of the cursor, and closing one eye completely can correspond to a drag action. The whole system is shown in Figure 3.

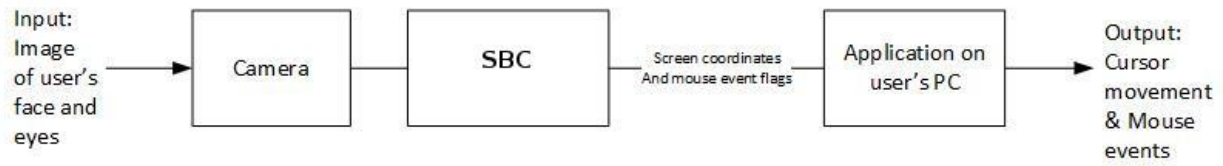

Figure 3: Gaze-controlled Mouse High Level Diagram

3. Gaze-Tracking Algorithm

3.1 Finding Key Locations on Face

3.1.1 Detecting the Face Region

To detect the face region, a Haar Cascade classifier for finding frontal faces was used, and, since it is unlikely that multiple users will be using the same device simultaneously, the first face region found by the classifier is used as the face region for the rest of the algorithm. A Haar Cascade classifier was chosen because it is a very accurate method for locating face regions. Furthermore, a version for frontal faces is already trained and available in OpenCV. An LBP Cascade classifier could be used as a faster, but less accurate alternative.

3.1.2 Locating the Eye Regions

For a face region detectable by the Haar Cascade classifier, the eye regions will fall into approximately the same area for every face with width = (width of the face region)/ 3 and height = (height of the face region)/4 [2]. Using geometry to determine the eye regions is ideal compared to using another classifier since it will be faster than the classifier, and the regions only need to be accurate enough to contain the iris/pupil center and eye corners.

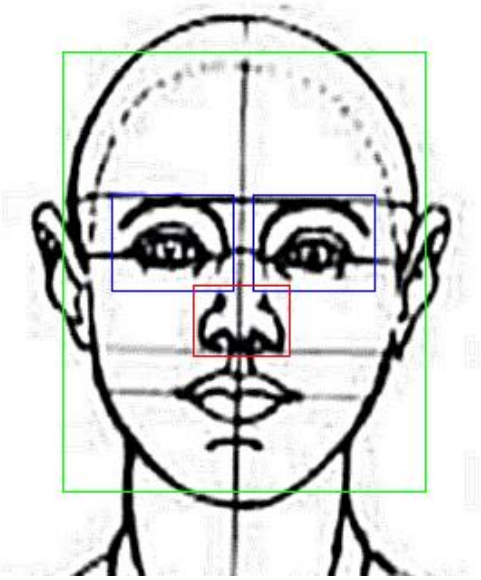

Figure 4: Face, Eye and Nose Regions

3.1.3 Finding the Iris or Pupil Center

To find the iris center, the method described in [2] is used on an up-scaled image of an eye region. This method uses color information and radial symmetry to determine the iris center. This method was chosen because of its high accuracy, which is very important since the location of the iris center is the primary factor in determining where a user is looking.

3.1.3.1 Removing Glare

Users wearing spectacles cause some issues in the above algorithm due to glare from the computer screen and surrounding area on the spectacles. The effect of glare can be minimized by first splitting the eye region into its HSV channels. Then an appropriate thresholding operation is performed on the H and S channels to determine the area where the glare is located in the eye region. The pixels in this area can then be mean shifted to the mean of the rest of the eye region. The resulting new eye region image can then be combined with the normal eye region image used in the earlier method for iris center detection to reduce the effect of glare, but without the noise incurred by using the new eye region image alone.

Alternatively, if using an IR camera, and thus no color information is available, Hough circle detection can be used to find the pupil center since the pupil area has a strong difference from the iris in IR and the pupil will not be occluded by the eyelids like the iris. The main benefit of this method is its higher speed.

3.1.4 Finding the Eye Corners

To find the eye corners, the expected geometry of an eye corner is used. First, the eye region is split into two eye corner regions using the center of the iris detected earlier. Then each eye corner region is scaled to a fixed size and a fixed kernel of roughly the shape of an eye corner is applied. The location of the resulting maximum is then used as the eye corner location.

3.1.5 Finding the Nose Region and Point(s)

To find the nose region, a Haar Cascade classifier for finding noses is used on the face region. A Haar Cascade classifier was chosen because of its accuracy at finding the nose region. Furthermore, it gives us the location of a point fixed on the face that can be used for calculating the users face orientation.

3.2 Gaze Vector Estimation

3.2.1 Derivation Technique to Find the Gaze Point

In order to determine the gaze point of the user on a screen device, the positions of several key landmarks on the users face and eyes first need to be determined, followed by an analysis of these landmarks relative to each other and the spatial environment in which they are found. The following sections describe in detail the stages of analysis that are required assuming that these points have already been found; a visual summary of all the necessary points is provided in Figure 5. Included are landmarks referencing the eye centers, eye corners, and tip of the nose.

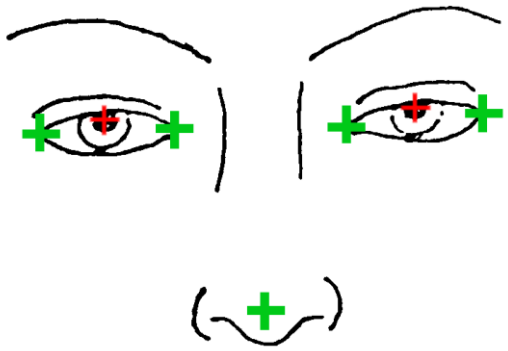

Figure 5: Summary of All Landmarks Necessary for Gaze Point Determination

3.2.2 Eye Orientation

The algorithm determines the gaze point by deriving a mapping of the user's eye pupil/iris combination to areas on the user devices screen. This is accomplished by finding the center of the user's iris as well as the two opposing eye corners for each eye. Once these points are found, one vector is drawn from the inner eye corner to the center of the iris (the eye vector EV), and another vector joins the two eye corners (the corner vector CV), as depicted in Figure 6.

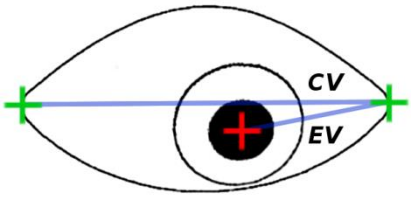

Figure 6: Close up of eye with landmarks and vectors visible

As the user looks at different areas, their eye will move, and so EV will vary in length and orientation. The changes of EV were measured with respect to the corner vector CV because CV is comprised of points attached to the users face and give.

3.2.3 Compensating for Head Movement

The previous procedure for finding the users gaze point works under the assumption that there are only two reference frames: one for the eye and one for the screen, and a mapping between these two frames can be accomplished. However there are three reference frames the system needs to account for given any user: the eye, head, and screen. The extra reference frame for the head represents the possibility of rotational and translational head movement (6 DOF). The following sections detail the method used to accommodate for this movement such that the gaze point can still be reliably determined.

3.2.3.1 Rotation

Head motion in the X plane is detected by the use of a projected vector from the outer edge of each eye corner as shown in Figure 7. The change in length of this horizontal face vector (HFV) allows for the measuring of the amount of rotation committed in the Y-axis. This is due to the perspective change during a rotation, which will shorten the length of the HFV. Similarly, projecting a vertical vector bisecting the HFV and connecting with the nose landmark (the VFV) will provide a way to measure any amount of rotation along the X-axis. Finally rotations in the Z-axis can be accounted for by measuring the orientation of HFV.

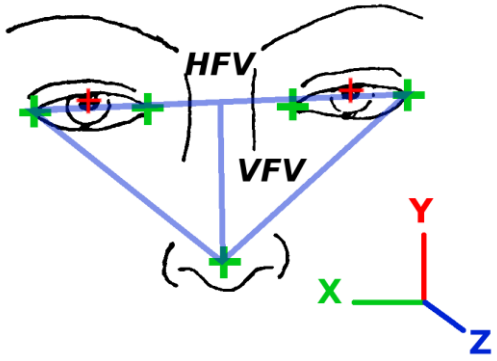

Figure 7: Diagram of Horizontal and Vertical face vectors

Measuring the lengths of these face vectors will only provide the degree of rotations, not the direction; to solve for the direction, it becomes necessary to use the Z-offset properties of the nose landmark. While the face rotates the nose will move in a different manner compared to the rest of the landmarks used due to the nose not being in the same Z-plane, as illustrated in Figure 8.

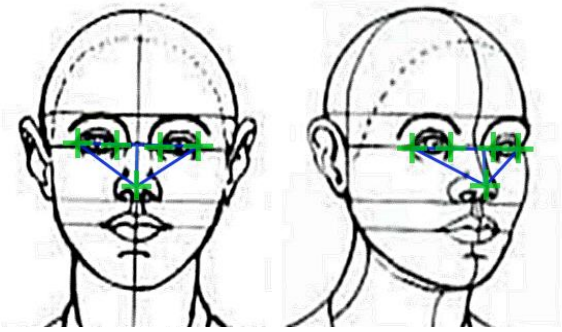

Figure 8: Illustration of Offset Characteristics of Nose Landmark

Once the rotation in the XY axes have been determined, it becomes possible to take the projection of the viewed *EV* from the eyes onto a plane rotated by the given amount to find the true undistorted *EV*, as shown in Figure 9.

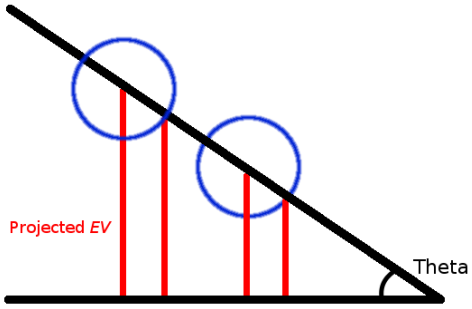

Figure 9: Determination of Undistorted EV

3.2.3.2 Translation

Similarly to rotation, translation can also accounted for by finding the necessary offset to apply to EV to bring the measured vector properties to its true dimensions. By taking the measured XY translational displacement from where the user initialized calibration to the current location, the amount of necessary offset for the EV can be found. Using experimental results, it is possible to find a scaling factor to apply against the offset found to provide the correct adjustments to the EV.

4. Hardware

4.1 Odroid C1+

The ODROID C1+ (Odroid) shown in Figure 4, is an extremely versatile and powerful board computer which can be used for a variety of applications. It consists of 4 cores (quad-core) which is made of the world-renowned Acord RISC Machine (ARM) architecture. It is primarily made to run many of the free operating systems widely available and used such as Ubuntu, ARCHLinux and Debian.

Figure 10: Odroid C1+

The specifications of this powerful board computer are as follows:

Table 1: Odroid C1+ Specifications

ODROID C1+
Amlogic ARM® Cortex®-A5(ARMv7) 1.5Ghz quad core CPUs
Mali™-450 MP2 GPU (OpenGL ES 2.0/1.1 enabled for Linux and Android)
1Gbyte DDR3 SDRAM
Gigabit Ethernet
40pin GPIOs + 7pin I2S
eMMC4.5 HS200 Flash Storage slot / UHS-1 SDR50 MicroSD Card slot
USB 2.0 Host x 4, USB OTG x 1 (power + data capable)
Infrared(IR) Receiver
Standard Type-A HDMI connector
An I2S bus to support HiFi audio add-on boards
CEC function that doesn't require the RTC backup battery
A power path from USB OTG port as well as DC barrel connector
Improved SD-card compatibility
Ubuntu or Android OS

4.2 Camera

The camera is a simple HD webcam by Logitech shown in Figure 5. The model is C615. It gives an extremely clear, smooth and sharp picture.

Figure 11: Logitech C615

The camera is used to capture the pupils and making sure the point-of-gaze is captured correctly. It will detect the face and the pupils to track the point of gaze of the user.

The specifications are as follows:

Table 2: Logitech C615 Specifications

Logitech C615
Full HD 1080p video capture (up to 1920 x 1080 pixels) with recommended system
HD video calling (1280 x 720 pixels) with recommended system
Logitech Fluid Crystal™ Technology*
Autofocus
Photos: Up to 8 megapixels (software enhanced)
Built-in mics with automatic noise reduction
Hi-Speed USB 2.0 certified (recommended)
Universal clip fits laptops, LCD or CRT monitors

5. Connectivity

5.1 Design

The screen-coordinates and indicators for mouse events that are generated by the gaze-tracking algorithm are sent to an application on the user`s computer via Ethernet. This application will move the cursor and enable the user to interact with the computer via calls to functions from the windows.h library. The application will also display points on the screen which are required to calibrate the Gaze-controlled mouse upon first use. Figure n describes the software process exchange between the Odroid and the application on the user`s PC.

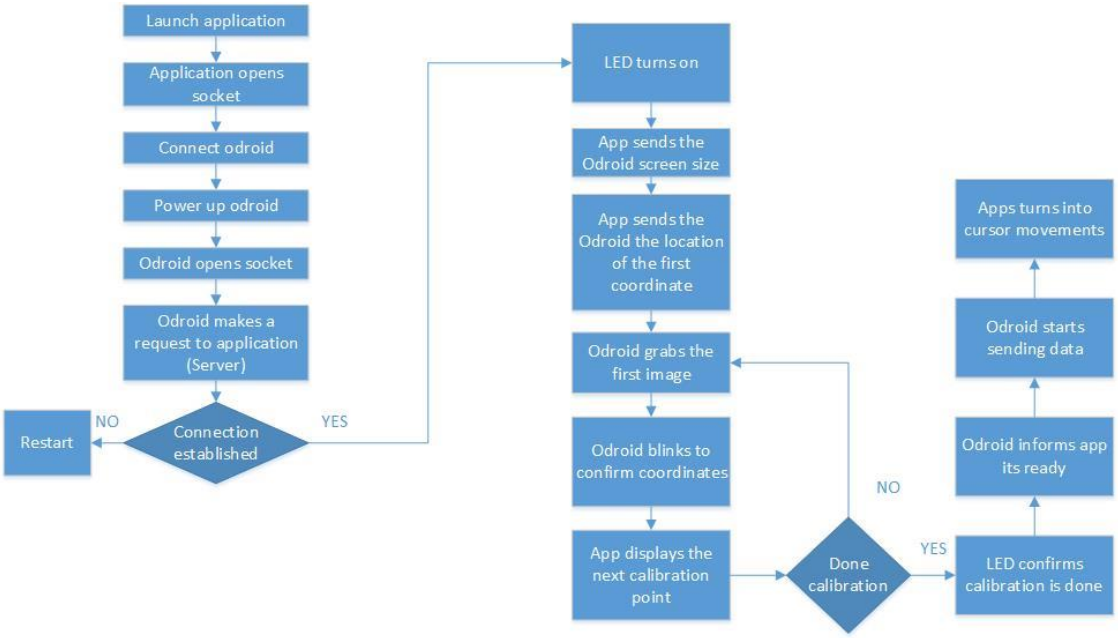

Figure 12: Gaze-Controlled Mouse System Operation

Once calibrated, the application will call functions to convert screen-coordinates into cursor movements and mouse events indicators/flags into clicking and dragging.

5.2 Hardware choices

Since the application will be running on the user's computer, this aspect of the system, by virtue, meets all the environmental system requirements listed in section 2.3 of the functional specification [1].

Ethernet and socket programming were selected to establish communication between the Odroid and the user`s PC because of its speed and ease of implementation in both hardware and software. Serial communication was also considered. While serial communication appeared to be simpler to implement in software, special hardware was needed to physically establish the connection. In addition, connection via Ethernet was found to be faster than that via serial cable. In the future a proper USB device driver will be developed for the production model.

6. User Interface

The User Interface (UI) is the control of the system. It will be the primary method to control the system and ensure everything is functioning as it is supposed to. The UI will consist of two parts, software and hardware. The software implementation will be fully designed in C++ and the Qt framework, with external libraries to enhance the software's flexibility, and to ensure that all the aspects of the software operate with efficiency.

6.1 Software

The primary goal of the software is to the calibrate the gaze-tracking code correctly for the camera to pick up on all the calibration points as well as alert the user of any errors that may occur during the calibration. Likewise, if the system is running free of all errors, the software displays this.

As stated above, the software UI is fully implemented in C++. With the addition of Qt, which is an application framework, building a Graphical User Interface (GUI) becomes much easier. The great thing about Qt lies in its cross-platform abilities. Developing it on Windows ensures that it will also work on the Odroid board which runs Linux. Compatibility will not be an issue. The board, the Host-PC Application have to work smoothly for calibration to work correctly and relay the data back to the board.

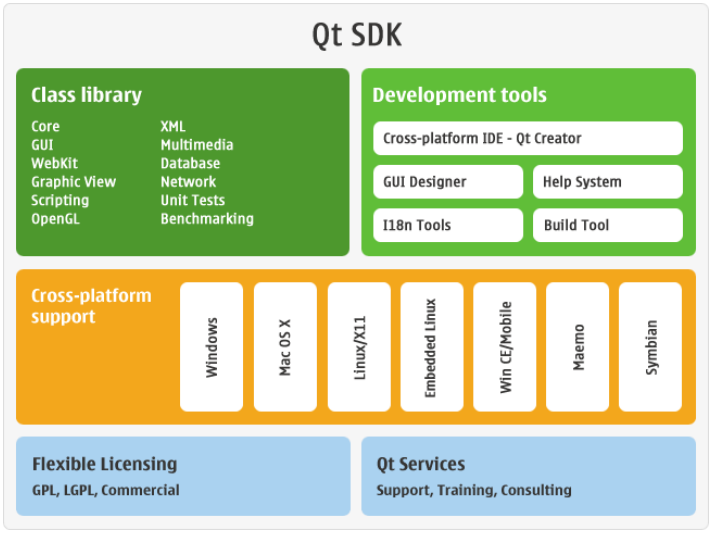

Figure 13: Qt SDK Features

The UI displays all the information required to setup the Odroid to be used with the gaze-controlled mouse; in particular:

- Calibration button to calibrate the monitor so that the camera can pick up all the pixels of the screen
- Reset button to recalibrate the software
- Status field to let the user know that Odroid is currently in calibration mode

The Odroid will collect the data from images that the user will look at during the calibration mode. This data helps generate reference screen coordinates . Once data is collected, the host-pc is contacted to let it know that calibration has been completed. Host-pc turns to active mode where it gets data from the Odroid.

6.2 Hardware

The hardware is the more important component of the two aspects of the user interface. The hardware consists of the Odroid board and a camera. The board is the heart of the system. It will be responsible for collecting data regarding the dimensions of the user's screen as well as image processing during calibration mode. This will collect information which will trigger the onboard software to produce screen coordinates. Once the screen coordinates have been collected, it will be collected by the Host-PC Application and the mouse will appear at the collected screen coordinates. This process will occur in real-time so as the point of gaze from the pupil travels, the cursor will coordinate with it.

The camera is responsible to capture the user's pupils and see where the point of gaze from the pupils is moving on the screen. The system will also make use of Light Emitting Diodes (LEDs) to communicate to the user what the mode the board is in, and the current status of the board. In particular:

- 1 button to start shutdown script for Odroid (turn off)
- 1 button to restart eye tracking program (also functions as calibration initialize i.e. software reset)
- 1 LED to display Odroid on/off status (Odroid already has led on board for this)
- 1 LED to indicated calibration mode on/off
- 1 LED to indicate eye tracking status (success/failure)
- 1 LED to indicate Odroid-pc connection status

7. Test Plan

First, the gaze-tracking algorithm and the application on the user's PC (Host PC Application) will be tested separately (Unit Testing). When each part has been confirmed to be operating correctly, the two will be combined and the normal use case will be tested (Integration Testing). Testing of the gaze-tracking software during both unit testing and integration testing will be done under ideal lighting conditions.

7.1 Unit Testing

During unit testing, the Odroid will be preconFigured to only run the code relevant to the unit that is being tested. Peripherals such as a monitor, mouse and keyboard will be connected to the Odroid for troubleshooting purposes.

7.1.1 Gaze-Tracking Software

1. Run gaze-tracking code
2. Verify that it finds the face, points on the face, eyes, eye centres…
3. <<Verify more stuff here as needed>>
4. Verify that screen-coordinates and mouse-event flags are produced as indicated in Table 3

Table 3: Gaze-Tracking Software Unit Test User Inputs and Expected Outputs

Mouse Function	Test User input	Expected output
Cursor movement	User looks at an icon on their desktop	Gaze-tracking code produces screen coordinates
Left clicking	User winks left eye	Gaze-tracking code produces a left-click flag
Right clicking	User winks righteye	Gaze-tracking code produces a right-click flag
Double-left-click	User blink both eyes	Gaze-tracking code produces a Double-left-click flag
Drag	User looks at a point on their screen User closes one eye User looks at another area of their screen. User opens that eye	Gaze-tracking code produces screen coordinates and left button down flag Gaze-tracking code produces left button up flag

7.1.2 Host-PC Application code dedicated to moving the cursor and causing mouse-events

1. Run Host-PC Application
2. Connect and turn on Odroid
3. Run Connectivity code only
4. Verify that a connection has been established between the user`s PC and the Odroid
5. Verify that the mouse behaves as expected when the test inputs given in Table 4 are applied manually

Table 4: Host-PC Application Code for Mouse Function Unit Test Inputs and Expected Outputs

Mouse Function	Test input	Expected output
Cursor movement	Screen-coordinates corresponding with an icon on the desktop	Cursor moves to that icon
Left clicking	Left-click flag	The icon is highlighted
Right clicking	Right-click flag	The context menu appears
Double-left-click	Double-left-click flag	A window associated with that icon opens
Drag	Left-button-down flag and screen coordinates Left-button-up flag	That icon is moved to another area on the desktop

7.2 Integration Testing

Integration testing consists of three stages: Connectivity, Calibration and Active. The Connectivity stage confirms that the Odroid is able to communicate with the Host-PC Application. During the calibration stage, the required calibration points are gathered. This stage also serves to confirm that the system is functioning properly in calibration mode. Finally, the Active stage verifies the capability of the Gaze-controlled mouse to operate as a regular mouse.

7.2.1 Connectivity

User Input: User runs the Host-PC Application, connects the Odroid and turns it on

Expected Output:

- The Odroid enters calibration mode and requests for the size of the user's screen.
- An LED turns on, indicating that the connection has been established
- Another LED turns on, indicating that the system is in calibration mode
- The Host-PC Application enters calibration mode and sends the Odroid the size of the screen
- The Odroid requests for the first calibration point
- The Host-PC Application displays the first calibration point

7.2.2 Calibration

User Input: User looks at calibration point

Expected output during calibration:

- The Odroid collects data about the image of the user looking at the calibration point
- The Odroid requests for the an image of the next calibration point until calibration is complete

Expected output after calibration

- The Odroid tells the Host-PC Application that calibration is complete
- The calibration LED turns off and an LED indicating that at the system is in active mode turns on
- The Host-PC Application goes into active mode and requests data from the Odroid

7.2.3 Active

When the user performs actions given in the User Input column of Table 5 the corresponding result given in the Expected Output column should occur.

Table 5: Gaze-Controlled Mouse Integration Test User Inputs and Expected Outputs

Mouse Function	User input	Expected output
Cursor movement	User looks at an icon on their desktop	Odroid sends screen coordinates to the Host-PC Application Cursor moves to that icon on their desktop
Left clicking	User winks left eye	Odroid sends a left-click indicator to the Host-PC Application That icon is highlighted.
Right clicking	User winks righteye	Odroid sends a right-click indicator to the Host-PC Application The context menu appears.
Double-left-click	User blink both eyes	Odroid sends a Double-left-click indicator to the Host-PC Application A window associated with that icon opens
Drag	User looks at an icon on their desktop User closes one eye User looks at another area of their screen. User opens that eye	Odroid sends screen coordinates and left button down indicators to the Host-PC App That icon is dragged to the area of the screen that the user is looking at Odroid sends a left button up indicator to the Host-PC App

8. Conclusion

This document describes design solutions chosen to meet the function specifications of the Gaze-controlled mouse. A camera was chosen, among other reasons, to determine the gaze-point via non-invasive means. A separate SBC was chosen to ease integration for users with older systems and enhance user experience. During implementation, this document will be used as a guide to ensure that all functional and design specifications are met.

References

[1] NaEYEgation Technologies, "Functional Specification for the Gaze-controlled Mouse," Burnaby, BC, 2015.

[2] E. Skodras and N. Fakotakis, "Precise localiszation of eye centers in low resolution color images," *Image and Vision Computing,* 2015.