Design Document

Client:
Stephen Gilbert, VRAC

Advisor:
Daji Qiao

Project Members:
Justin Derby, Kris Scott, Tyler Burnham,
Arjay Vander Velden, Will Bryan, Scott Connell

Project MAY13-20

Contents

1 Introduction

1.1 Purpose

This document describes the design decisions made for Project Eyeris. It encompasses the hardware that will be used, the open source software that will be used, and the software that will be implemented. The goal of this document is to provide the necessary design knowledge for others to be able to replicate our system.

1.2 Scope

The product being produced is a mobile eye tracking solution that streams wirelessly. The system will include stereoscopic shutter glasses with an embedded computer system and a battery. Viewers of the study will be able to view what the user is seeing with added eye position data since we are streaming the data wirelessly. The product will also track both eyes in order to tell the depth at which the user is looking for 3D environments.

1.3 Definitions, Acronyms and Abbreviations

Term	Definition
User	The user wearing the glasses who is having his/her eyes tracked.
Administrator	The user administering the use of the glasses and running the server.
Viewer	User(s) who are able to watch the stream over the internet.
Raw Video	Video that is directly coming from the cameras
Encoded Video	Video that has been analyzed
Glasses	The glasses the user is wearing the is housing the cameras, embedded systems, and battery.
Embedded Board	The board that does the processing of the cameras and outputs the (x,y) coordinates of the eye-tracking data, as well as the compressed video from the out-facing camera. The local storage will also be on this board.
Broadcasting Server	Where the Administrator and Viewers will connect to see data and streams of the Glasses from the user
Embedded Computer System	The combination of the two Gumstix embedded boards and the embedded board to transmit the video data wirelessly all worn by the user

UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ISO	International Organization for Standardization
IEEE	Institute of Electrical and Electronics Engineers (read *I-triple-E*)
RANSAC	Random Sample Concensus (an ellipse fitting algorithm)
TBB	Intel® Threading Building Blocks for C++ which provides multi-threading for multi-core processors
Boost	C++ library that provides support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing

1.4 References (MLA style)

1.4.1 Calibration Free Algorithm

Ablassmeier, Markus. *Calibration-free Eye Tracking by Reconstruction of the Pupil Ellipse in 3D Space*. *Eye Tracking Research & Applications: Proceedings ; ETRA 2008 ; [Eye Tracking Research and Applications Symposium] ; Savanna, Georgia, USA*. By Stefan Kohlbecher, Stanislavs Bardinst, Klaus Bartl, Erich Schneider, and Tony Poitschke. New York, NY: ACM, 2008. 135-38. *Eye Tracking Research & Application*. 26 Mar. 2008. Web. 20 Oct. 2012. <http://www.informatik.uni-trier.de/~ley/db/conf/etra/index.html>.

1.4.2 Starburst Algorithm

Dongheng Li; Winfield, D.; Parkhurst, D.J.; , "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on* , vol., no., pp.79, 25-25 June 2005
doi: 10.1109/CVPR.2005.531
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1565386&isnumber=33215>

1.4.3 ITU Gaze Group (IT University of Copenhagen, Denmark)

Hansen, John P., Dan W. Hansen, Javier S. Agustin, Sune A. Johansen, Henrik H. Tomra Skovsgaard, and Martin Tall. "ITU GazeGroup." *ITU GazeGroup*. ITU Copenhagen, n.d. Web. 24 Oct. 2012. <http://www.gazegroup.org/home>.

1.4.4 Japan Head-Mounted Display

Rodriguez-Silva, D.A.; Gil-Castineira, F.; Gonzalez-Castano, F.J.; Duro, R.J.; Lopez-Pena, F.; Vales-Alonso, J.; , "Human motion tracking and gait analysis: a brief review of current sensing systems and integration with intelligent environments," *Virtual Environments, Human-*

Computer Interfaces and Measurement Systems, 2008. VECIMS 2008. IEEE Conference on , vol., no., pp.166-171, 14-16 July 2008
doi: 10.1109/VECIMS.2008.4592774
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4592774&isnumber=4592737>.

1.4.5 Pupil Tracking on Off-Axis Images

Swirski, Lech, Andreas Bulling, and Neil Dodgson. Robust Real-time Pupil Tracking in Highly Off-axis Images. University of Cambridge. ACM, 2012. Web. 24 Oct. 2012. <http://www.cl.cam.ac.uk/research/rainbow/projects/pupiltracking/>.

1.4.6 Overview of H.264

Wiegand, Thomas, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. "Overview of the H.264/AVC Video Coding Standard." *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY* 13.7 (2003): 560-576.

1.4.7 H.264 Standard

Kalva, Hari. "The H.264 Video Coding Standard." *IEEE MultiMedia* 13.4 (2006): 86-90.

1.5 Overview

Section 1 is the introduction and includes a description of the project and references to documents used for research.
Section 2 provides a system overview.
Section 3 contains the system context.
Section 4 describes the system design method, standards and conventions.
Section 5 contains the component descriptions.
Section 6 includes the Requirements Traceability Matrix.

2 System Overview

2.1 System Characteristics

The system will stream the out-facing view of the user to allow real-time analysis by the administrator and viewers. There can be several viewers at any one time, which means that there is a possibility of numerous concurrent users. The client side portion of the system will be platform independent to accommodate the different operating systems of the Viewers. The embedded board will store the out-facing video of the user's view and the eye coordinates associated with those frames as a backup to curb loss in wireless connectivity or poor streaming latency. This will provide a functionality to the administrator to download high-quality data for later analysis.

2.2 System Architecture

2.2.1 Software System Architecture

Our system will be based on a client-server architecture for streaming video. Clients will connect to the server to grab data and streams. The server will cater connections through standard web browsers. The server will employ HTML5 and CSS3 compliant standards so that as time goes on, there will be little maintenance required. The system can leverage existing technology of web browsers and make it cross-platform.

The internal camera software will be written so that it takes advantage of the hardware it will be on. It will be designed with multiple processors in mind.

2.2.2 Hardware System Architecture

The glasses will have five cameras: one outward-facing and four inward-facing, two per eye. Each set of two cameras will send their video feeds to one of the two eye-tracking boards. The eye-tracking boards will analyze the video streams using the eye-tracking algorithm to determine the direction the eye is looking. Once calculated, the eye data will be sent to the main embedded board. The outward-facing video feed will also be sent to the main board. This board will synchronize the outward video with the eye data, and send it over the wireless network to the server.
See Figure 4-1 for the functional block diagram of this system.

2.3 Infrastructure Services

2.3.1 Security

Since the system will be using a client-server architecture, security will be needed on the outward facing server end. Any data that gets sent to the server needs to be sanitized and error checked to make sure malformed, or illegal data cannot compromise the system.

2.3.2 Logging

The system should log everything to provide assistance in case anything unforeseeable happens. The levels of logging are as defined:

Number	Level	Description
1	Severe	Messages indicating a serious failure
2	Warning	Messages indicating a potential problem
3	Info	Messages for informational messages
4	Config	Messages for static configuration messages

The default logging level will be 1. As the level increases, all encompassing levels will be active. For example, a level of 3 means that all Severe, Warning, and Info messages will be recorded.

3 System Context

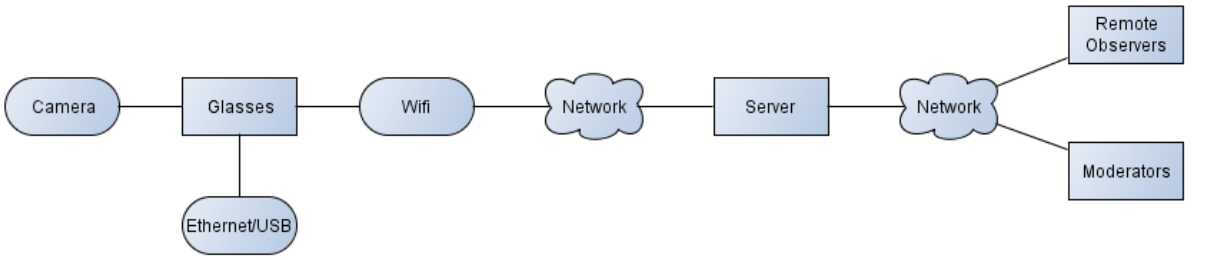

Figure 3-1: The System Block Diagram

3.1 Server

3.1.1 Data

The data that the server will hold will be the video stream of the outward facing camera as well as the x,y components of the eyes.

3.1.2 Data Format

The video format will be encoded using H.264/MPEG-4 AVC for the reason that it is a standard in the field and it is well-known and well-implemented. The x,y components of the eyes will be included in each frame of the video stream.

3.1.3 Failure

If the connection between the Server and Administrator fails, the video stream and data will be stored on the camera device itself so that not actual data loss occurs.

3.2 USB Connection

The USB connection will be provided for calibration of the system.

3.2.1 Data

The data that will be sent over the USB connection will be command payloads, video streams, and x,y components of the eyes.

3.2.2 Data Format

The data format will be H.264/MPEG-4 AVC for video feeds, while the command payloads will be of a custom data structure type. The x,y components of the eyes will be included in each frame of the video stream.

3.2.3 Failure

If the USB connection between the Admin’s Computer and Camera fails, the software on the Admin’s Computer will message the Admin of the occurrence.

4 System Design

4.1 Design Method and Standards

We are using an object oriented design approach for this product. The benefit of this is for future code reuse and modularity. Modularity in our system will allow new features in the future to be added easier.

4.2 Documentation Standards

For each software file implemented, there should be a header containing a description of what the file does and any legal information it might have. For every class defined, there should be a description of the purpose of the class. Every function needs to be commented with the description, parameters, and return information. Additional comments inside the function body will be used at the developer's discretion. Commenting conventions that will be used will be Doxygen compliant. As example is as follows:

```
/**
* Summary here; one sentence on one line (should not, but can exceed 80 chars).
*
* A more detailed description goes here.
*
* A blank line forms a paragraph. There should be no trailing white-space
* anywhere.
*
* @param string $first
*   "@param" is a Doxygen directive to describe a function parameter. Like some
*   other directives, it takes a term/summary on the same line and a
*   description (this text) indented by 2 spaces on the next line. All
*   descriptive text should wrap at 80 chars, without going over.
*   Newlines are NOT supported within directives; if a newline would be before
*   this text, it would be appended to the general description above.
* @param int $second
*   There should be no newline between multiple directives (of the same type).
* @param bool $third
*   (optional) TRUE if Third should be done. Defaults to FALSE.
*   Only optional parameters are explicitly stated as such. The description
*   should clarify the default value if omitted.
*
* @return array
*   "@return" is a different Doxygen directive to describe the return value of
*   a function, if there is any.
*/
```

4.3 Naming Conventions

Our code will be using CamelCase for coding. Camel case is the practice of writing words with some inner uppercase letters, such as compound words or phrases in which the elements are joined without spaces, while each element has a capital letter within the compound. The first letter will always be capitalized for class declarations, all others will have the first letter not capitalized.
All files will include at most one class declaration in the upper level. Its respective header file (if the file has one) shall be named the same, but with a different file extension (.h or .hpp for example). If the file holds a class, it shall be named as the class name of the upper level.

4.4 Programming Standards

4.4.1 Coding Standards

The C++ standards that will be used is the ISO/IEC 14882:2011 standard. The C Standards that will be used is the ISO/IEC 9899:2011 standard or better known as C11 standard.

4.4.2 Internet Protocol Standards

The User Datagram Protocol (UDP) will be used to provide streaming for our application. The Transmission Control Protocol (TCP) will be used to provide commands throughout the application.

4.4.3 IEEE Standards

Since our system will be using wireless transmission, one of the standards we will be using is the 802.11 Wireless LAN standard (IEEE Std 802.11-2012). Also, since our system will be using video, the system will conform to the IEEE Standard on Video Techniques: Measurement of Resolution of Camera Systems (IEEE Std 208-1995).
Our team will be following the IEEE Systems and Software Engineering Standard for Architecture Description (ISO/IEC/IEEE 42010:2011(E)) as well as the IEEE Standard for Configuration Management in Systems and Software Engineering (IEEE Std 828-2012)

4.4.4 C++ Conventions

Include statements will be at the top of the file underneath the header. Macro declarations will be under the include statements.

Class Declaration

The following is how classes will be defined in a header file. The indent size will be set to 4 spaces.

```
class MyClass
{
public:
    void myFunction( Type paramOne, Type paramTwo );
```

```
protected:
private:

};
```

Method Declaration

This is how functions will be written in the source file. There will be no space between the function name and the parentheses containing the parameters. There will be a space between the parentheses and the first and last parameters. The indent size will be 4 spaces.

```
void MyClass::myFunction( Type paramOne, Type paramTwo )
{
   // code goes here
}
```

Alternative to be used when the function declaration has too many parameters to fit within the 80 character length.

```
void MyClass::myFunction( Type paramOne,
                          Type paramTwo )
{
   // code goes here
}
```

If/Else if/Else Statements

Here is the template for If, else if, and else statements. There should be a space after the parenthesis and after the last value check before the closing parenthesis. The indent size will be 4 spaces. Assignment statements will never occur in the if or else if condition check.

```
if( value == someValue )
{
   // code goes here
}
else if( value == someOtherValue )
{
   // code goes here
}
else
{
   // code goes here
}
```

For loop

The increment value should be declared in the for declaration. A space should be between the open parenthesis and value declaration and the post loop increment and the closing parenthesis. Indent size will be 4 spaces.

```
for( int i = 0; i < someInt; i++ )
{
   // code goes here
}
```

While loop

The value being checked in the while condition should be declared outside the while loop. The value should never be assigned in the while condition. A space should be between the open parenthesis and the value as well as between the end of the condition check and the closing parenthesis. Indent size will be 4 spaces.

```
int value = 0;
while( value < 100 )
{
   // code goes here
}
```

4.5 Software Development Tools

For our project we are using eclipse/netbeans for our c++ programming. We will be using c for our embedded programming, also we will be working on a linux kernel. Tools for forming our documents: Microsoft Word and yed (for our diagrams).

4.6 Outstanding Issues

As of right now, there are no outstanding issues since this release of this document.

4.7 Decomposition Description

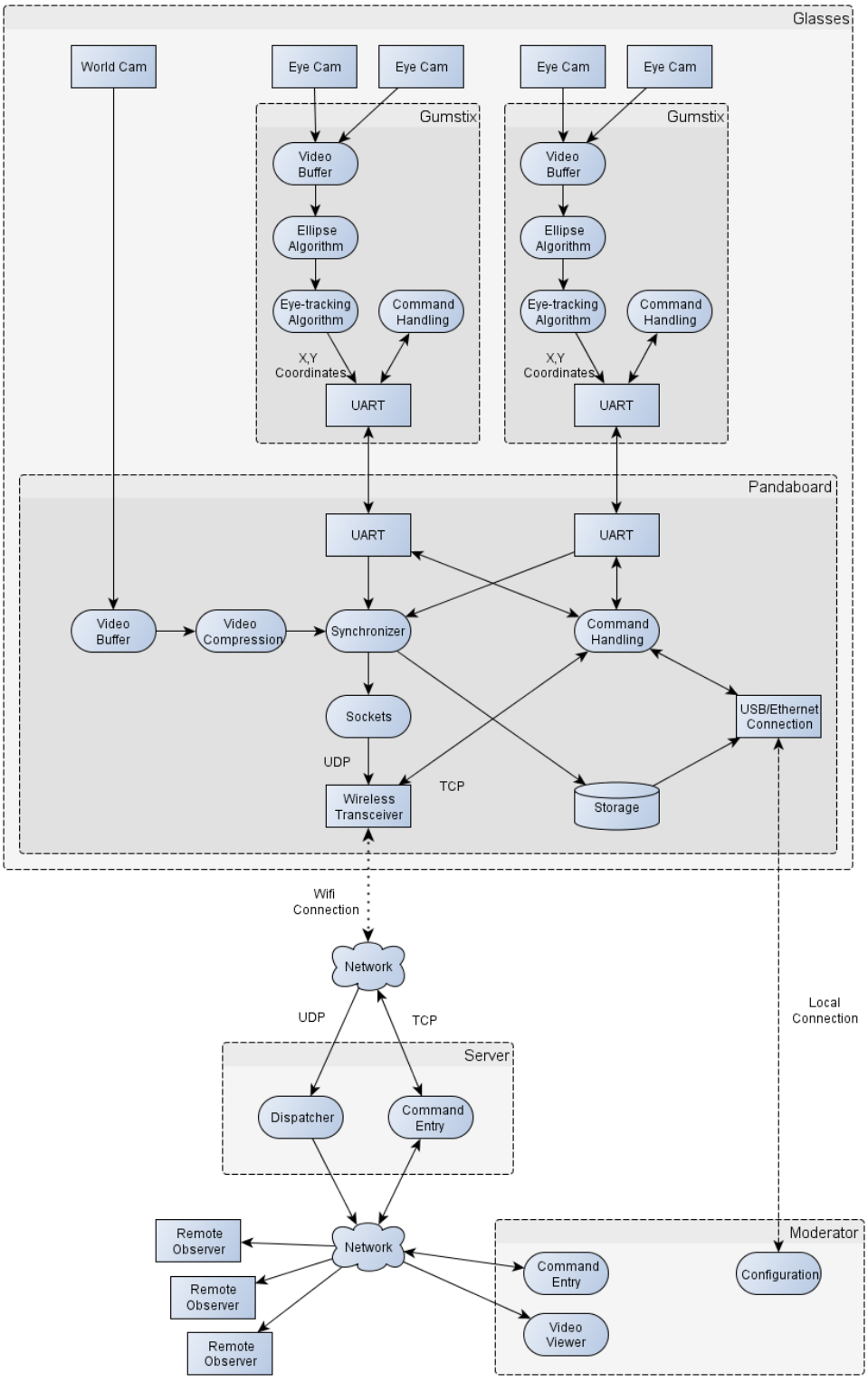

Figure 4-1: System Data Flow Block Diagram

5 Software Component Description

5.1 Video Buffer Module

5.1.1 Purpose

The reason we have this module is to handle interfacing with the camera to get the video data from a camera.

5.1.2 Function

This module will read the pins that provide the raw video data on a frame-by-frame basis. It will compile the frames together in a sequence and hold them until they are read from another module.

5.1.3 Subordinates

Ellipse Algorithm Module
Video Compression Module

5.1.4 Dependencies

A frame needs to be populated by a camera.

5.1.5 Interfaces

```
/**
 * Grabs the frame from the camera, appends it to the list of frames in the buffer
 */
public void captureFrame()
/**
 * Gets the next frame in line to be analyzed from the buffer
 * @return The latest frame in the buffer
 */
public VideoFrame getNextFrame()
```

5.1.6 Resources

Buffer

5.1.7 References

N/A

5.1.8 Processing

Polls for the video data frame from the IO interface of the camera.

5.1.9 Data

VideoFrame will hold the raw video data probably in the form of a char array
VideoBuffer will hold sequence of VideoFrames

5.2 Ellipse Algorithm Module

5.2.1 Purpose

This module will be used to find the ellipse of the eye from the video frame.

5.2.2 Function

The functionality will be to grab the next frame from the Video Buffer module and find the ellipse of the eye from the image to provide to the eye tracking algorithm.

5.2.3 Subordinates

Eye-Tracking Algorithm Module

5.2.4 Dependencies

Have to have a frame in the Video Buffer

5.2.5 Interfaces

```
/**
 * Calls to get the next frame and then analyzes it to detect the ellipse.
 * @return Ellipse object
 */
public Ellipse getElipse(VideoFrame frame)
```

5.2.6 Resources

N/A

5.2.7 References

See the reference 1.4.5 for an explanation of the algorithm being used.

5.2.8 Processing

RANSAC ellipse fitting algorithm
Haar-like features detection algorithm
Image histogram processing
OpenCV
Boost
TBB

5.2.9 Data

Ellipse which contains the data points that will construct an ellipse

5.3 Eye-Tracking Algorithm Module

5.3.1 Purpose

This module handles running the eye-tracking algorithm on the ellipse received from the Ellipse Algorithm Module.

5.3.2 Function

The functionality of this module is to grab the ellipse from the Ellipse Algorithm Module and run a series of matrix operations to find a cone intersection point to determine the position of the eyes. Then it will pass that information on to main board for synchronization and transmission.

5.3.3 Subordinates

N/A; this is the starting point, being run on its own thread.

5.3.4 Dependencies

Have to run the ellipse algorithm for each camera before running this algorithm.

5.3.5 Interfaces

```
/**
 * Runs the algorithm, starting point for the program on the GUMSTIX embedded
 * platform that will be looping until stop is called. Run on it’s own thread
 */
public void run()
/**
 * Stops the algorithm from running
 */
public void stop()
```

5.3.6 Resources

Access to UART connection for communicating coordinates to the embedded board used for transmitting.

5.3.7 References

See Section 1.4

5.3.8 Processing

Calibration free eye tracking algorithm

5.3.9 Data

Ray which is an x, y, and z 3D coordinate to determine position and depth.

5.4 Command Handling Module

5.4.1 Purpose

This module will accept commands and then process those requests, mainly for calibration or configuration purposes.

5.4.2 Function

This function will receive commands from the main embedded board via the UART connection. It will have a list of acceptable commands and carry out the specific duties associated with those commands. Will be run on it’s own thread.

5.4.3 Subordinates

UART receives something from the main embedded board.

5.4.4 Dependencies

Access to the UART

5.4.5 Interfaces

```
/**
 * Receives commands and processes them.
 */
public void receiveCommand()
```

5.4.6 Resources

Access to UART connection

5.4.7 References

N/A

5.4.8 Processing

Ran on it’s own thread, processes commands.

5.4.9 Data

Command will hold the necessary information to complete a request from the user.

5.5 Video Compression Module

5.5.1 Purpose

This module will handle compressing the video data received from the out-facing camera.

5.5.2 Function

The functionality of this module will be to receive the video data from the Video Buffer module on the main embedded board and then run the compression algorithm for the frames received. It will then pass this to the synchronizer module.

5.5.3 Subordinates

N/A this will be on it's own thread and process at it receives data.

5.5.4 Dependencies

There needs to be video data in the Video Buffer from the out-facing camera.

5.5.5 Interfaces

```
/**
 * Start will kick off the polling on the thread to look for video data to compress.
 */
public void start()

/**
 * Stop will end the compressing module and not process anymore video data.
 */
public void stop()
```

5.5.6 Resources

Access to a buffer holding the video frames to be compressed.

5.5.7 References

See section 1.4

5.5.8 Processing

We will be using the H.264 compression technique. There are open source libraries that can be used to do this.

5.5.9 Data

High Definition video data received from the out-facing camera video buffer.

5.6 Sockets Module

5.6.1 Purpose

The purpose of this module is to be able to interface with the sockets we are connecting to make videos.

5.6.2 Function

The function of this module is to be available to the synchronizer to be able to push the data through.

5.6.3 Subordinates

N/A

5.6.4 Dependencies

Must have a wireless transceiver to connect with.

5.6.5 Interfaces

```
/**
 * Connects to the Socket we are trying to connect with.
 * @return boolean returns whether or not a successful
 */
public boolean connect()

/**
 * Binds to the socket we are sending out.
 */
public void bind()

/**
 * Sends the data out UDP style.
 */
public void sendUDP()

/**
 * Sends the data TCP style.
 */
public void sendTCP()

/**
 * Recieves data back when using the TCP protocol.
 * @return boolean does is it getting the correct data back.
 */
public boolean receive()
```

5.6.6 Resources

The physical wireless transceiver.

5.6.7 References

UDP and TCP protocols.

5.6.8 Processing

Running a real time stream through the wireless transceiver.

5.6.9 Data

Simple data that does not need to be stored other than.
Sockets in which we are connecting.

5.7 Synchronizer Module

5.7.1 Purpose

This module will provide the synchronization of the three different camera sources, the two cameras tracking the eyes and the out-facing camera.

5.7.2 Function

The functionality of this module is to take the two rays from the eye tracking algorithm and sync them with each other as well as the out-facing video data to ensure the proper coordinates are linked to the correct frame.

5.7.3 Subordinates

N/A will be started on it's own thread.

5.7.4 Dependencies

The out-facing video data must already be compressed and the Eye tracking algorithm must be ran for both eyes.

5.7.5 Interfaces

```
public void receiveVideo(CompressedVideo video, DateTime stamp)
public void receiveCoord(3DRay coord, int camera, DateTime stamp)

/**
 * Starts the synchronizer thread
 */
public void start()

/**
 * Stops the synchronizer thread
 */
public void stop()
```

5.7.6 Resources

N/A

5.7.7 References

N/A

5.7.8 Processing

N/A

5.7.9 Data

N/A

5.8 Storage Module

5.8.1 Purpose

This module will handle storing the video data backup on the main embedded board.

5.8.2 Function

The functionality will be to receive the encoded video data with ray data from both eyes and store to the flash memory on the main embedded board.

5.8.3 Subordinates

N/A

5.8.4 Dependencies

Synchronization module ran and provides the data.

5.8.5 Interfaces

```
/**
 * Creates a new video file control system to provide automatic
 * video file segmentation for easy control
 * @param filename The filename of the video
 * @return Video ID to reference
 */
public int createVideoFileID(string filename)

/**
 * @param videoId ID of the video to save the video to
 * @param videoData Video data to save
 */
public void storeVideo(int videoId, Video videoData)

/**
 * Closes the video file control system
 * @param videoId Video ID to close
 * @return 0 if control system has been closed, otherwise -1
 */
public int closeVideoFile(int videoId)
```

5.8.6 Resources

Access to the flash memory on the main embedded board.

5.8.7 References

N/A

5.8.8 Processing

IO intensive writing and reading of the data to and from the flash memory.

5.8.9 Data

Encoded video data from the out-facing camera and the ray data from each eye camera.

5.9 Dispatcher Module

5.9.1 Purpose

The purpose of this module is to provide relaying for all clients that are connected to the server in an easy and manage-free way.

5.9.2 Function

When a client connects to the system, this module will open a socket and hold onto it until the client disconnects. Any data coming from the camera will be given to all sockets that are connected.

5.9.3 Subordinates

N/A

5.9.4 Dependencies

The server must be accepting connections, and reachable by it’s connection users (firewall rules).

5.9.5 Interfaces

```
/**
 * Starts the listening server
 * @return 0 if no errors, -1 otherwise
 */
public int startListening()
/**
 * Stops the listening server
 * @return 0 if no errors, -1 otherwise
 */
public int stopListening()
/**
 * Sets the port that will accept incoming connections
 * @return 0 if no errors, -1 otherwise
```

```
 */
public int setPort(int port)
```

5.9.6 Resources

N/A

5.9.7 References

N/A

5.9.8 Processing

N/A

5.9.9 Data

N/A

5.10 Command Entry Module

5.10.1 Purpose

The command entry module is on the administrator and is the component in which we are sending commands to control the administrator side.

5.10.2 Function

The purpose of the command entry module on the administrator to adjust both the streaming in and out.

5.10.3 Subordinates

N/A

5.10.4 Dependencies

Network

5.10.5 Interfaces

```
/**
 * Change Destinations
 * @param ips an array of ips with and the ports we are streaming to.
 */
public void ajustDest(ips[])
```

5.10.6 Resources

Access the network so we can get data.

5.10.7 References

See section 1.4.

5.10.8 Processing

The administrator will need to be able to process the incoming video as well as port the video to all the observers.

5.10.9 Data

N/A

5.11 Configuration Module

5.11.1 Purpose

The purpose of this module is to be able to interface with the glasses and configure them to work with everyones eyes.

5.11.2 Function

The configuration module will sync the glasses with the eyes of the user.

5.11.3 Subordinates

User Eyes

5.11.4 Dependencies

USB/Ethernet connection to the glasses.

5.11.5 Interfaces

```
/**
 * Sets the server IP and port for the system
 * to connect to when the system starts up
 * @param host IP address
 * @param port Port
 */
public void setServer(String host, int port)

/**
 * Gets the server IP and port for the system
 * @param Server IP and Port
 */
public InetAddress getServer()

/**
 * Sets the debug state of the system for
 * getting eye feeds and other information
 * @return Previous state of debugging
```

```
 */
public int setDebug(int debugging)

/**
 * Sets the parameters we need to sync the eyes.
 * @param args
 */
public int setParameters(Params args...)

/**
 * Runs the configuration test.
 * @param spots the amount of sync points
 */
public void config(int spots)
```

5.11.6 Resources

USB
Memory

5.11.7 References

See section 1.4

5.11.8 Processing

The changes will be made on the administrator and then sent over USB/Ethernet to the glasses.

5.11.9 Data

The storage will be on the glasses so we don’t need to any for the administrator in this module.

6 Hardware Component Description

6.1 TCM8230MD Camera

The TCM8230MD(A) is a camera module which includes area color image sensor embedded with camera signal processor that meets with VGA format. In the sensor area 492 vertical and 660 horizontal signal pixels, and the image size meets with 1/6 inch optical Format. Use of the CMOS process enables low power consumption operations. It also provides excellent color reproduction through its primary color filter, and embedded camera signal processor enables small and simple camera system.

6.1.1 Purpose

The camera captures an image of the eyes to stream to the embedded computer systems.

6.1.2 Function

The TCM8230MD will be configured to provide VGA (640x480) video output, YUV422 format, at 30 fps. The module can be configured via its I2C interface. The images stream on a 8 bit parallel output.

6.1.3 Interfaces

External Clock
EXTCLK: pin 2
The camera relies on an external clock signal. All cameras in the system will be using the same clock signal from the main board.

I2C
SDA: pin 5
SCL: pin 4
The I2C interface (address 0x3C), can be used to configure the device:

- Image resolution (VGA, QVGA, QQVGA, CIF, QCIF, subQCIF)
- Framerate (15fps, 30fps)
- Image format (YUV422, RGB565)

Data
DOUT0:7: pins 11-14, 17-20
The 8 bit parallel data out pins stream the image data according to the configuration.

DCLK: pin 10
The clock signal for the data interface.

VD: pin 8
HD: pin 9
Vertical and horizontal synchronization signals for the pixel data.

6.1.6 Power

PVDD: pin 1, 2.8V
Power for camera sensor.

IOVDD: pin 16, 2.8V
Power for I/O.

DVDD: pin 6, 1.5V
DVSS: pin 7, GND
The logic high and low voltages for digital circuits and analog to digital converters.

6.1.7 References

Datasheet: https://www.sparkfun.com/datasheets/Sensors/Imaging/TCM8230MD.pdf
Product Page: https://www.sparkfun.com/products/8667

6.2 TCM8240MD Camera

The TCM8240MD is a high quality, very small 1.3 mega-pixel color camera from Toshiba with the standard data+I2C interface. The nice thing is that we have a good supplier of these cameras, so they should be around for some time. This camera is also unique in that it offers on-board JPEG compression.

6.2.1 Purpose

The camera captures an image of the world to stream to the embedded computer systems.

6.2.2 Function

The TCM8240MD will be configured to provide 1300x1040 video output, YUV422 format, at 15 fps. The module can be configured via its I2C interface. The images stream on a 8 bit parallel output.

6.2.3 Interfaces

External Clock
EXTCLK: pin 13
The camera relies on an external clock signal. All cameras in the system will be using the same clock signal from the main board.

I2C
SDA: pin 10
SCL: pin 8
The I2C interface (address 0x3D), can be used to configure the device:

- Image resolution (VGA, QVGA, QQVGA, CIF, QCIF, subQCIF)
- Image format (YUV422, RGB565)

Data
DOUT0:7: pins 4-1, 24-21
The 8 bit parallel data out pins stream the image data according to the configuration.

DCLK: pin 19
The clock signal for the data interface.

VDLK: pin 16
HDLK: pin 15
Vertical and horizontal synchronization signals for the data blanking period.

6.2.6 Power

PVDD: pin 6, 2.8V
Power for camera sensor.

IOVDD: pin 18, 2.8V
Power for I/O.

DVDD: pin 12, 1.6V
The logic voltage for digital circuits and analog to digital converters.

6.2.7 References

Datasheet:
https://www.sparkfun.com/datasheets/Sensors/Imaging/TCM8240MD_E150405_REV13.pdf
Product Page: https://www.sparkfun.com/products/8668

6.3 Gumstix DuoVero

The DuoVero Crystal COM features the dual-core OMAP 4430 Digital Video Processor running at 1GHz with 1GB RAM on board.

6.3.1 Purpose

The two Gumstix boards will interpret the images from the eye cameras and send data about the eye direction to the main board. Dedicating one Gumstix per eye will divide the work needed to process the video streams from the cameras and ensure that the images are processed as fast as possible.

6.3.2 Function

Each of the two Gumstix boards will read and interpret the images from the cameras. The boards will be running the images through the Ellipse and Eye-Tracking algorithms, and then send the X,Y coordinates to the main board over UART.

6.3.3 Interfaces

The Gumstix uses an expansion board to provide the I/O pins used by the Linux Kernel. These pins will provide the data in from the cameras, and the UART connection to the main board.

6.3.6 Power

Powered via expansion board (DuoVero series or custom) connected to dual 70-pin connector.

6.3.7 References

Product page: https://www.gumstix.com/store/product_info.php?products_id=285

6.4 PandaBoard

PandaBoard is a low-cost, open OMAP 4 mobile software development platform. It's built around the Texas Instruments OMAP44xx processor.

6.4.1 Purpose

This serves as the central board which controls all of the other modules and components connected to the glasses.

6.4.2 Function

The PandaBoard will organize and communicate all of the necessary data. It will grab the outward-facing video feed and compress it. It will then sync the video feed with the eye-tracking algorithm output data from the Gumstix. The PandaBoard will also packetize the information into UDP and stream it over the network using Wifi.

6.4.3 Interfaces

The PandaBoard will connect to the Gumstix using two UART connections using its general I/O pins. The video feed will also be streamed through general I/O pins or camera extension header. It will use its embedded wireless 802.11n to stream out the video and coordinates, as receive commands from the administrator.

6.4.6 Power

The PandaBoard can be powered at 3.7 - 5 V from a battery source.

6.4.7 References

Product page: http://www.pandaboard.org/content/platform

Document Control

Title:	Design Document
Issue:	Issue 1, Draft 2
Date:	2 December 2012
Author:	Project Eyeris
Distribution:	Online
Filename:	DesignDocument.pdf

Document Change Record

Date	Version	Author	Change Details
25 October 2012	Issue 1 Draft 1	Project Eyeris	First draft
2 December 2012	Issue 1 Draft 2	Project Eyeris	Content and visual changes

Appendix A

A.1 Eye Tracking Algorithm Research

A.1.1 Calibration Free Algorithm

The Calibration Free algorithm reconstructs the pupil ellipse for 3D space. It can do this by using two calibrated cameras per eye. It finds the intersection of two cones through a series of matrix operations.

Pros

- Developed for a mobile system
- Calibration free - you only need to calibrate it one time and then it will work on anyone.
- Accuracy within ~1.5 degrees based on how well one-time calibration is done and image noise
- 2D tracking to 3D projection (Can find convergence based on ray casts)
- Algorithm is specified (but not implemented) in the document

Cons

- Requires two cameras for each eye, which means more image analyzing
- Many variables need to be known before the one-time calibration
- Use of matrix and trig functions - could slow the algorithm and not provide real-time
- Needs another algorithm to provide the ellipses to do calculations - can be provided by numerous libraries.

A.1.2 Starburst Algorithm

The Starburst Algorithm is an open sourced algorithm that does dark-pupil infrared tracking. It is a hybrid between feature-based and model-based tracking which is a mixture for runtime performance and accuracy.

Pros

- Developed for a mobile system.
- Source code already available. Developed in *C*++
- Accuracy to within one degree of visual angle based on how well the calibration was done.
- Only requires one camera per eye.
- The algorithm doesn’t seem to have a lot of processing intensive steps so performance is better than pure model based algorithm.

Cons

- Will require calibration every time based on a number of points in the scene.
- Not as accurate if the glasses move at all, calibration may need to be done again.

A.1.3 Japan Head-Mounted Display

This solution was developed for severely disabled people so that they can operate things with only eye movement. The first (and documented) solution has a built in display, while the second iteration has no display, with the user looking at a computer screen.

Pros

- Uses a compensation algorithm to help improve accuracy of the algorithm
- Calibration is a two step process

Cons

- No documented implementation to look at for help
- Needs at least two mirrors and a very complicated setup
- No error of accuracy given.
- Has display on system which we don't need

A.1.4 ITU Gaze Group Algorithm

The ITU GazeGroup out of ITU in Copenhagen created an algorithm to do eye tracking. They used a system consisting of cameras and a computer. They developed a lot more for gaze based interactions.

Pros

- Already has been used in a system before.
- Open sourced and already developed.

Cons

- Developed in *C#* and requires *.Net* framework, essentially very difficult to do on an embedded platform.
- Publications are unavailable to see specifics about the algorithm.
- Calibration requires the subject to hold head very still.

For our project, we will be using the *Calibration Free Algorithm*. We are choosing this because of the accuracy and limited calibration. We are concerned that the matrix operations will require a lot of cpu time but we are working with hardware that should defer this problem. The limited calibration gives us more flexibility and accuracy in case the glasses move even a little bit. Hopefully, if we have time, we will implement the *Starburst Algorithm* in place of our ellipse-finding algorithm if the current ellipse algorithm is slower. Since the *Starburst Algorithm* is 2D to 2D mapping, we can rewrite it to give us the 2D ellipse it finds and then feed that into the *Calibration Free Algorithm* to provide 2D to 3D tracking with better accuracy.