Design Report

For

LED Video Player

Version 1.2

Prepared by Tell O'Neal & Alexander Eklund

University of Idaho Capstone Design

Last modified May 14th 2014

Table of Contents

1. Executive Summary

The system being developed is an ***LED*** Matrix capable of displaying video. Potential applications of such a device are signs and large video displays used at concerts and sporting events. Our solution is a proof of concept for an open source design that business owners and hobbyists could use to build and maintain their own sign or video display. Current vendor solutions are very expensive to rent or buy and maintain, so such a design could be very helpful to entities that are interested in LED displays, but have a limited budget.

The LED Matrix can be assembled from one or more 32x32 LED panels. Upon completion of this project, functionality will be demonstrated with video played on a 2x2 and also a 4x1 configuration of panels.

The LED matrix and hardware required to drive it will be accompanied by a web based software interface that will allow the output of the display and the source of content for the display to be managed and configured from any internet connected device such as a PC, tablet, or smartphone.

2. Background

The LED panels that our project uses are the building blocks for vibrant street signs and enormous LED walls and displays. LED signs are an effective way of advertising, and can be found standing next to streets, on buses, and buildings. LED displays are used extensively in the entertainment industry. They can be found hanging in the middle of most professional and high level college sporting venues, used to playback clips of recent plays in slow motion, and give the audience a live, close up view of the action. Where LED displays really stand out is at live music events and dance clubs. An LED wall towering behind a live band or DJ shines very bright in an evening setting, and can help create the right atmosphere and visual content to go along with music. Video clips, evolving geometric shapes, graphs of complex functions, and strobing effects can look impressive when edited and played in time to music on LEDs, and for many types of rock and dance music complement a live show very well.

Major sporting venues are generally going to have access to a budget that will allow them to meet their LED display needs through traditional vendors. However, small businesses and gigging musicians alike could benefit from an open source design that provides code and plans to create an LED display of a size to fit their needs.

To truly make the design layman friendly, providing some parts to make wiring, giving structure to, and storing the system a breeze would be necessary. This is where a business model could potentially fit into this “open source” design. Provide code and plans freely, and work as a vendor of

the parts necessary to build the LED displays. Users would help each other work through issues on an internet forum, limiting the need for customer service.

3. Problem Definition

3.1 Goals and Deliverables

LED Matrix Functions:

- Display Video on an LED matrix consisting of 4 panels (2 x 2 array)
- Display Video with 24 bit color
- Display Video at 30 Frames per Second

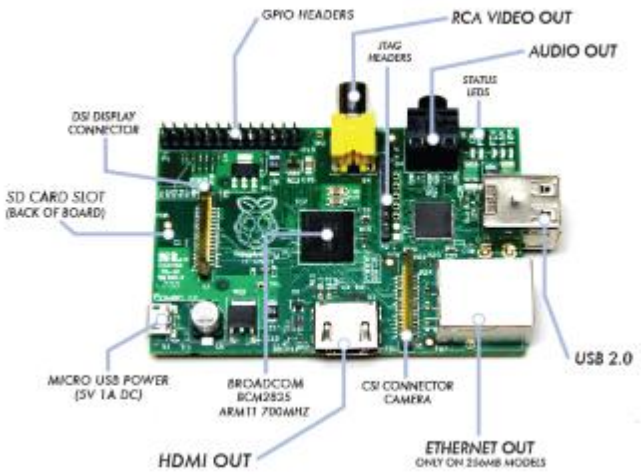
User Interface Functions:

- Allow user to specify Video Source located on YouTube or locally
- Allow user specify Number of LED Panels & Arrangement
- Allow user to select Portion of Video Source to be Used
- Pause and Play LED Display

3.2 Specifications & Constraints

All software for this system will be developed to run on a Raspberry PI Model B developed by the Raspberry Pi Foundation. The software will be targeted for a port of Debial Linux distribution for the Raspberry Pi called Raspbian.

Table 1- Critical Specifications for Raspberry Pi Model B

	**SOC (System on a Chip)**	Broadcom BCM2835
	CPU	700 MHz Low Power ARM1176JZ-F Processor
	GPU	512MB SDRAM
	Onboard storage	8GB SD card
	Network Connection	Onboard 10/100 Ethernet RJ45 Jack
	Alternate Network Connection	USB 802.11n WIFI adapter
	Operating System:	Linux (Raspbian“wheezy”) Kernel v.3.10

3.2.1 Operating Environment for RTL/VHDL

The RTL developed to drive the LED panels will be developed for an Altera cyclone family of FPGAs. For this project the RTL will be executed on the DE0-Nano development board from Altera.

Table 2- Critical Specifications of DE0-Nano Dev Board

	**FPGA**	Altera Cyclone IV EP4CE22F17C6N
	Memory	32 MB SDRAM. 2Kb EEPROM
	Clock	50 MHz oscillator

3.2.2 Assumptions and Dependencies

3.2.2.1 Software Dependencies

Software will be developed for computers running a Debian based Linux Operating System.

The following 3rd party software libraries will be required to be installed on the computer running the software.

- NodeJS
  - Npm (node package manager)
- OpenCV
- FFmepeg

3.2.2.2 Hardware Dependencies

- The following type of LED panels:
  - Adafruit 32x32 RGB LED matrix Panel, ID:607
    http://www.adafruit.com/products/607
- Raspberry PI model B
- Altera Cyclone Family of FPGAs

4. Project Plan

4.1.1 Team responsibilities

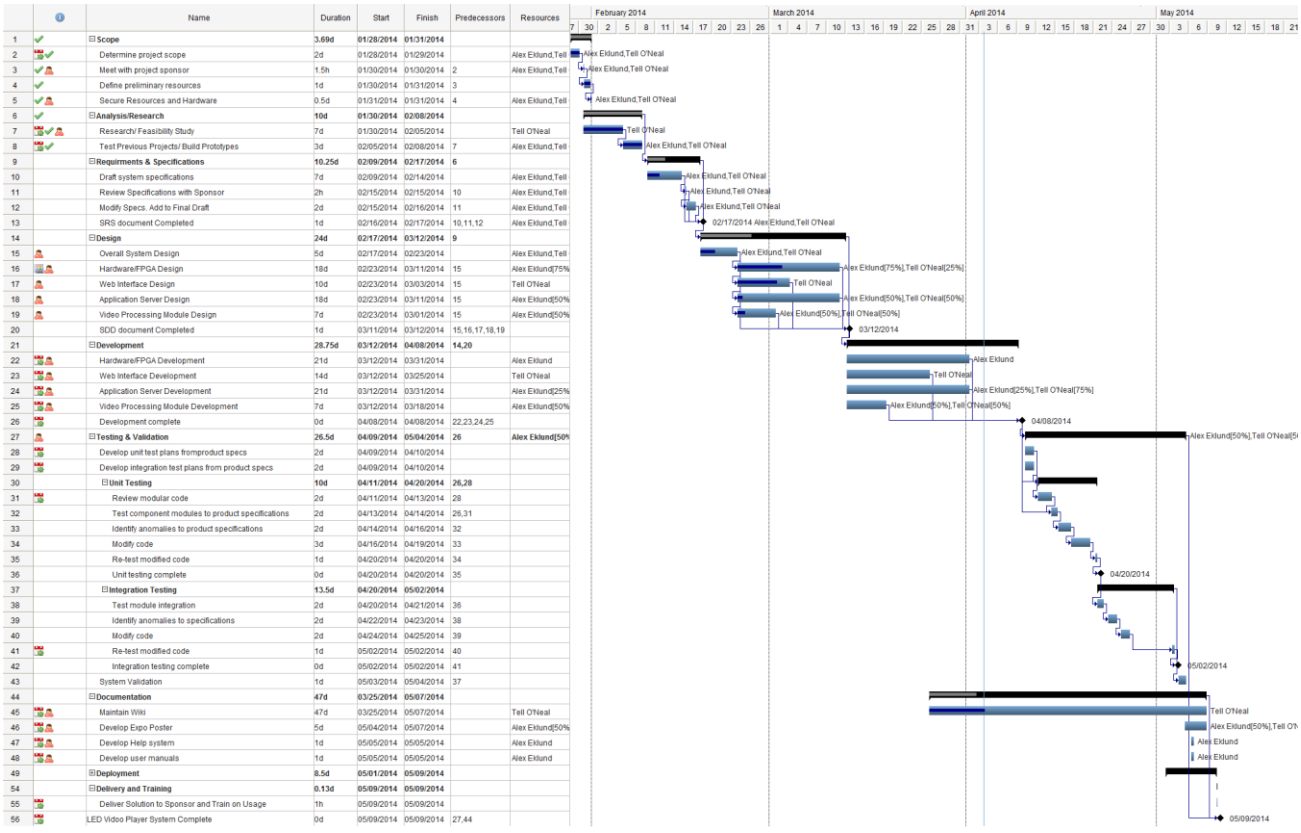
4.1.1.1 Tell

- Research Possible Concepts
- Write SRS Document
- Develop Project Schedule
- Develop User Interface
- Develop Conversion Module
- Prepare Snapshot Day Poster
- Prepare Design Review Presentation
- Prepare Expo Poster
- Setup Expo Booth
- Contribute Details about User Interface to Final Design Report

4.1.1.2 Alex

- Develop FPGA VHDL Code
- Develop FPGA Driver
- Integrate FPGA Driver into Conversion Module
- Manage Wiki Page
- Prepare Final Design Report

4.1.2 Gantt chart



	Name	Duration	Start	Finish	Predecessors	Resources
1	**Scope**	**3.69d**	**01/28/2014**	**01/31/2014**		
2	Determine project scope	2d	01/28/2014	01/29/2014		Alex Eklund,Tell
3	Meet with project sponsor	1.5h	01/30/2014	01/30/2014	2	Alex Eklund,Tell
4	Define preliminary resources	1d	01/30/2014	01/31/2014	3	
5	Secure Resources and Hardware	0.5d	01/31/2014	01/31/2014	4	Alex Eklund,Tell
6	**Analysis/Research**	**10d**	**01/30/2014**	**02/08/2014**		
7	Research/ Feasibility Study	7d	01/30/2014	02/05/2014		Tell O'Neal
8	Test Previous Projects/ Build Prototypes	3d	02/05/2014	02/08/2014	7	Alex Eklund,Tell
9	**Requirments & Specifications**	**10.25d**	**02/09/2014**	**02/17/2014**	**6**	
10	Draft system specifications	7d	02/09/2014	02/14/2014		Alex Eklund,Tell
11	Review Specifications with Sponsor	2h	02/15/2014	02/15/2014	10	Alex Eklund,Tell
12	Modify Specs. Add to Final Draft	2d	02/15/2014	02/16/2014	11	Alex Eklund,Tell
13	SRS document Completed	1d	02/16/2014	02/17/2014	10,11,12	Alex Eklund,Tell
14	**Design**	**24d**	**02/17/2014**	**03/12/2014**	**9**	
15	Overall System Design	5d	02/17/2014	02/23/2014		Alex Eklund,Tell
16	Hardware/FPGA Design	18d	02/23/2014	03/11/2014	15	Alex Eklund[75%
17	Web Interface Design	10d	02/23/2014	03/03/2014	15	Tell O'Neal
18	Application Server Design	18d	02/23/2014	03/11/2014	15	Alex Eklund[50%
19	Video Processing Module Design	7d	02/23/2014	03/01/2014	15	Alex Eklund[50%
20	SDD document Completed	1d	03/11/2014	03/12/2014	15,16,17,18,19	
21	**Development**	**28.75d**	**03/12/2014**	**04/08/2014**	**14,20**	
22	Hardware/FPGA Development	21d	03/12/2014	03/31/2014		Alex Eklund
23	Web Interface Development	14d	03/12/2014	03/25/2014		Tell O'Neal
24	Application Server Development	21d	03/12/2014	03/31/2014		Alex Eklund[25%
25	Video Processing Module Development	7d	03/12/2014	03/18/2014		Alex Eklund[50%
26	Development complete	0d	04/08/2014	04/08/2014	22,23,24,25	
27	**Testing & Validation**	**26.5d**	**04/09/2014**	**05/04/2014**	**26**	**Alex Eklund[50%**
28	Develop unit test plans fromproduct specs	2d	04/09/2014	04/10/2014		
29	Develop integration test plans from product specs	2d	04/09/2014	04/10/2014		
30	**Unit Testing**	**10d**	**04/11/2014**	**04/20/2014**	**26,28**	
31	Review modular code	2d	04/11/2014	04/13/2014	28	
32	Test component modules to product specifications	2d	04/13/2014	04/14/2014	26,31	
33	Identify anomalies to product specifications	2d	04/14/2014	04/16/2014	32	
34	Modify code	3d	04/16/2014	04/19/2014	33	
35	Re-test modified code	1d	04/20/2014	04/20/2014	34	
36	Unit testing complete	0d	04/20/2014	04/20/2014	35	
37	**Integration Testing**	**13.5d**	**04/20/2014**	**05/02/2014**		
38	Test module integration	2d	04/20/2014	04/21/2014	36	
39	Identify anomalies to specifications	2d	04/22/2014	04/23/2014	38	
40	Modify code	2d	04/24/2014	04/25/2014	39	
41	Re-test modified code	1d	05/02/2014	05/02/2014	40	
42	Integration testing complete	0d	05/02/2014	05/02/2014	41	
43	System Validation	1d	05/03/2014	05/04/2014	37	
44	**Documentation**	**47d**	**03/25/2014**	**05/07/2014**		
45	Maintain Wiki	47d	03/25/2014	05/07/2014		Tell O'Neal
46	Develop Expo Poster	5d	05/04/2014	05/07/2014		Alex Eklund[50%
47	Develop Help system	1d	05/05/2014	05/05/2014		Alex Eklund
48	Develop user manuals	1d	05/05/2014	05/05/2014		Alex Eklund
49	**Deployment**	**8.5d**	**05/01/2014**	**05/09/2014**		
54	**Delivery and Training**	**0.13d**	**05/09/2014**	**05/09/2014**		
55	Deliver Solution to Sponsor and Train on Usage	1h	05/09/2014	05/09/2014		
56	LED Video Player System Complete	0d	05/09/2014	05/09/2014	27,44	

5. Concepts Considered and Concept Selection

This project builds on work completed by two previous Senior Design groups, team RPLD in the spring of 2012, and team Automaton in fall of 2013.

Team RPLD developed an LED panel driver running on a ***Raspberry Pi*** that was capable of driving multiple LED displays aligned horizontally. In addition they developed PC client software that could be used to send a text message that scrolled across the displays from a remote PC. Their software allowed for basic configuration such as scrolling speed, message, and number of displays.

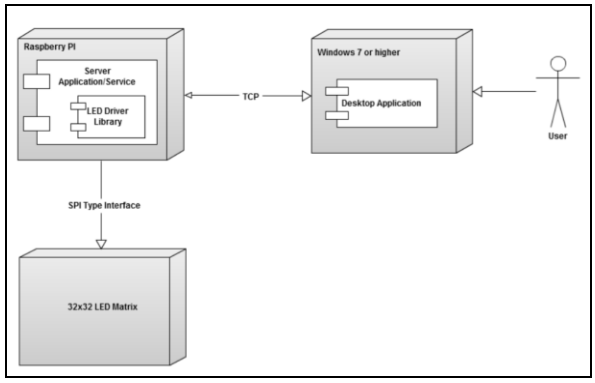


Figure 1- Team RPLD System Overview

Team Automaten used the LED driver developed by team RPLD, and developed additional software modules that allowed video to be displayed on the display. Team Automaten's software allowed for video to be downloaded or streamed from YouTube, converted to a format suitable for display on the LED panels, and displayed on a single LED panel. Team Automaten provided only a command line based user interface for controlling output to the LED and abandoned work on the GUI application built by team RPLD.

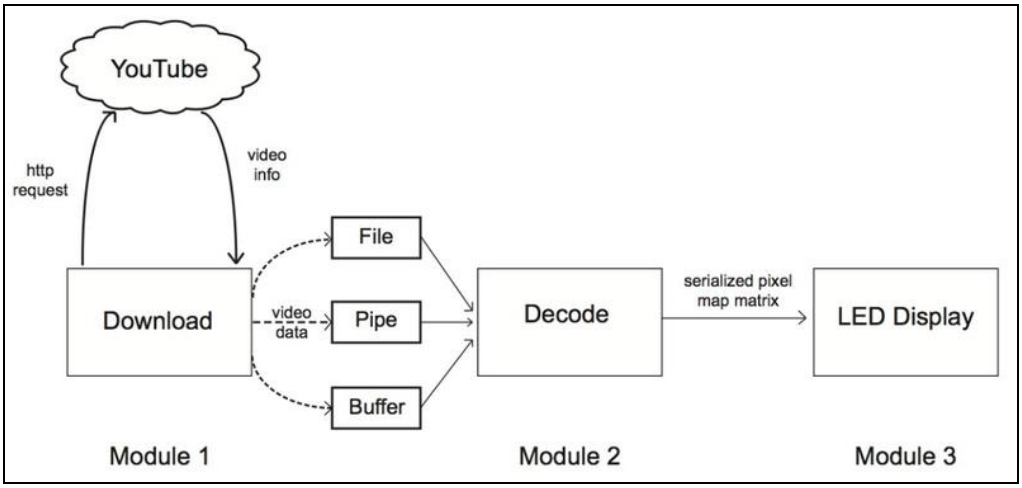


Figure 2- Team Automaten System Overview

Both previous projects provided useful prototypes that illustrated the concept of displaying content on the LED displays, but both left significant opportunities to improve and expand the system. The

next logical progression of this system is to allow for expandability to allow the use of additional LED panels to create a larger display with higher resolution. Initial research, and feedback from the previous teams has confirmed that the ***Raspberry Pi*** is not a suitable platform to base this system on if expandability is a goal, especially if video display at a reasonable frame rate is expected.

Displaying video at desirable frame rate (>30 FPS) on even a small LED matrix consisting of only 4 LED panels with a resolution of 64x64 LEDs requires precise, dependable timing from the hardware driving the LED display. The general purpose input/output (GPIO) pins of the Raspberry Pi provide a suitable interface for connecting to and driving the LED panels, but the rate at which data can be dependably pushed to the LED panels is bottlenecked by the Linux operating system running on the Raspberry pi.

Early research into Real Time Operating Systems (RTOS) running on the raspberry pi indicated improvements could made over the Linux OS used by the other teams, but even the best RTOS options for the PI could only guarantee timing requirements to ±20µs of accuracy. This would be an improvement over the current implementation, but would still significantly limit future expandability, and very little work done by the previous teams could be reused.

Faced with the fact that much of the originally system would need to be redesigned anyway by using an RTOS on the Raspberry Pi, and even that option would not provide a suitable platform for all of the desired goals for the project, another option that would provide greater performance was desired.

To address the current and future needs of this project, an FPGA was selected as the primary component used to drive the LED Matrix. The Raspberry Pi will still be used to process and convert video, but rather than interface directly with the LED panels it will output the processed data stream to the FPGA which will buffer and output the video with precisely defined timing.

In addition to adding the FPGA, this project will bring back a GUI interface such as the one used by Team RPLD. The interface will be web based. The server for the web-application will run on the Raspberry pi and improves upon the previous projects as it will not require any software to be installed on the PC or device used to control the displays; it will be accessible via a web browser.

6. System Architecture

The current system meets all goals but playing synced audio.

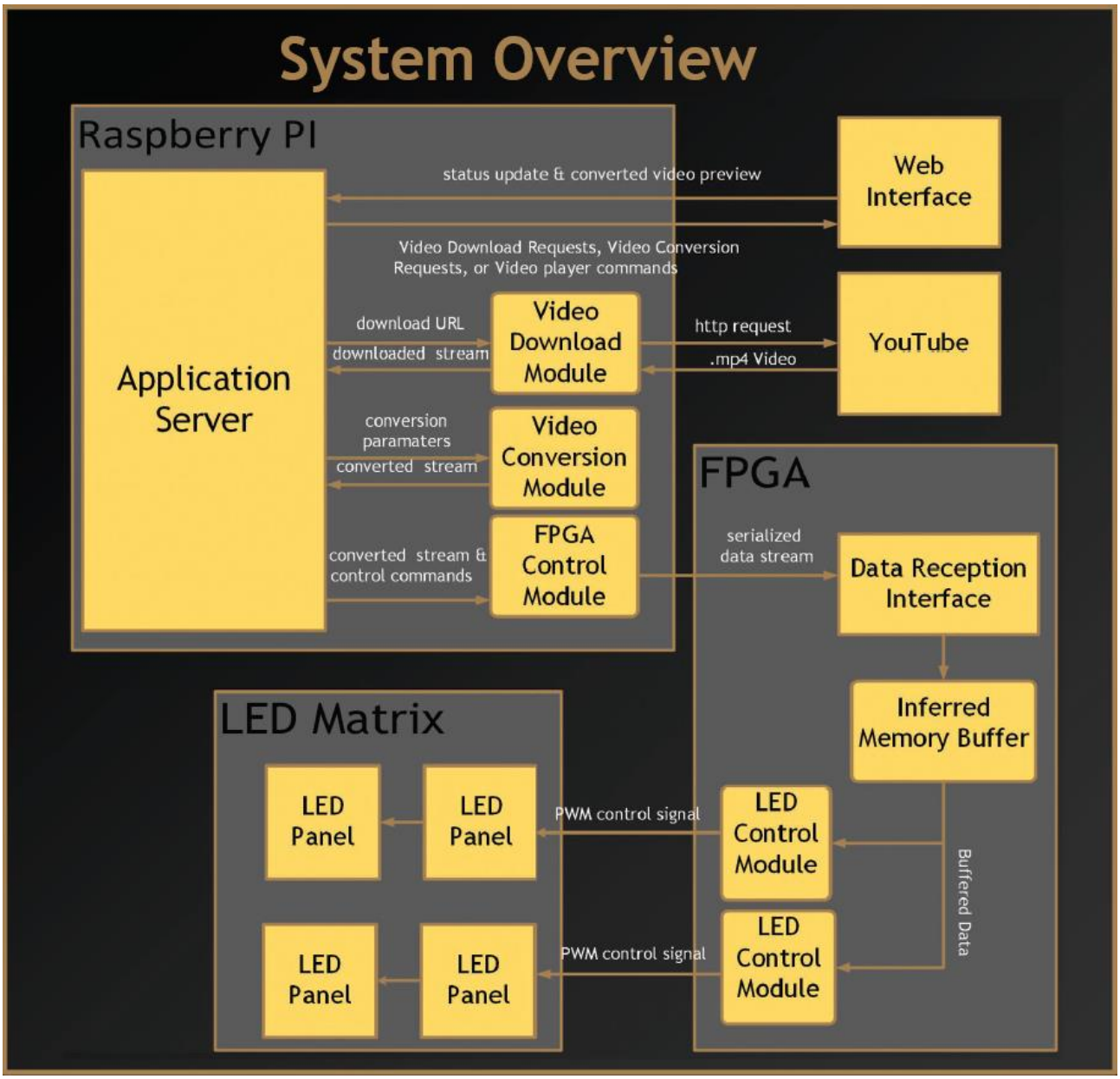


Figure 3 - System Overview

6.1 GUI and Video Conversion Software

The GUI is a client side Angular JS application that follows a MVC pattern.

The GUI is served by an application server built in Node.js. The application server and GUI communicate over a TCP web socket using a client/server model. Communications are asynchronous.

The application server utilizes the Node.js modules YTDL, and Fluent FFMPEG.
The YTDL module is responsible for sending an http request to youtube and saving the returned MP4 video.

The Fluent FFMPEG module handles cropping and resizing the video as specified by the settings in the user interface.

The Conversion Module is a C++ application using the Open CV video processing library. The conversion is a stand-alone command line application.

The Application server spawns a child instance of the Conversion module and passes all of the video conversion parameters received from the GUI, along with a file stream for the downloaded video.

The conversion module converts the video stream to a serialized data format suitable for use to be passed to the driver code.

6.2 Data Transmission and VHDL

Data Transmission - Data is sent from the Pi 8 parallel bits at a time. It is received by a module in the FPGA that shifts each new byte into a receiver register. A flag is raised every six bytes, at which point the data is stored in the memory module. Each panel writes two pixels at once, so the data for each pixel pair is stored together.

Memory Module - For our current design, a single frame is buffered. The LED control modules each provide the address of the pixel pair they are going to write next, and the output to each module is updated at every clock cycle.

LED Control Module - The control module is a state machine that keeps track of what pixel pair to write next, and when to write it. To write a pixel pair, the module grabs the memory module's output, splits it up, calculates whether the LEDs should be on or off, puts the correct values on the color and addressing wires to the panel, and pulses the signal that tells the panel to write the data to the LEDs.

6.2.1 Raspberry Pi Interface to FPGA Dev Board

The interface from the Raspberry Pi to the FPGA Dev Board will be made from the GPIO pins on the **Raspberry PI.**

This communication between the Raspberry Pi and the FPGA will be facilitated using a parallel 8 bit bus. The parallel bus requires 10 GPIO Pins to be used on the Raspberry Pi and 10 I/O pins on the FPGA.

Raspberry Pi Signal Name	**Data Flow Direction**	**FPGA Signal Name**	**Voltage Level**
GPIO 8	→	PI_cntl	3.3 V
GPIO 9	→	PI_rst	3.3 V

GPIO 0	→	PI_dat[0]	3.3 V
GPIO 1	→	PI_dat[1]	3.3 V
GPIO 2	→	PI_dat[2]	3.3 V
GPIO 3	→	PI_dat[3]	3.3 V
GPIO 4	→	PI_dat[4]	3.3 V
GPIO 5	→	PI_dat[5]	3.3 V
GPIO 6	→	PI_dat[6]	3.3 V
GPIO 7	→	PI_dat[7]	3.3 V

6.2.2 FPGA Dev Board Interface to LED Panels

The FPGA also must interface directly to the first LED panel in each chain of two. The following pin assignments will be required for the FPGA to interface to the LED Panel.

FPGA Signal Name	**Data Flow Direction**	**LED Panel Signal Name**	**Voltage Level**
Red1	→	R1	3.3 - 5V
Blue1	→	B1	3.3 - 5V
Green1	→	G1	3.3 - 5V
Red2	→	R2	3.3 - 5V
Blue2	→	B2	3.3 -5V
Green2	→	G2	3.3 -5V
Clock	→	CLK	3.3 -5V
OE	→	OE (output enable)	3.3 -5V
Latch	→	LAT(data latch)	3.3 -5V
AddrA	→	A	3.3 -5V
AddrB	→	B	3.3 -5V
AddrC	→	C	3.3 -5V
AddrD	→	D	3.3 -5V

7. Future Work

7.1 Recommendations for GUI

7.1.1 Authentication System

A critically needed addition to the GUI is a user based authentication system. Currently anyone can log into the user interface served on the raspberry pi. This was fine for proof of concept and testing on a local network with controlled access, but is completely unsuitable for access over the internet which is the intended use. In addition from securing the UI from un authorized access the Authentication system needs to manage sessions. Currently an unlimited number of clients may connect to the server, and send conflicting commands. The authentication system should limit (at least control) to a single session.

7.1.2 REST API for Video Conversions

Currently all video processing is performed on the PI. This works fine but is quite slow, and bottlenecked by the processor of the PI. Offloading the processing to a more powerful computer (even a moderate webserver) can reduce conversion times tenfold. The Pi already needs to be connected to the web to download files from video. These large source files are then passed to the conversion module. An alternative would be to create REST Api that the PI could send an http request with the YouTube url and conversion parameters. The REST application would download the video, convert it, and return the URL to converted video. This would then allow the pi to only have to download a much smaller converted video, rather than a large source video from you tube. This option would improve the user interface by greatly speeding up the download and conversion process, but it will add an extra layer of complexity.

7.2 Recommendations for the Hardware:

Another route to go for video conversion is to create a DVI converter for the system. This module would be plugged into any computer, and follow standard DVI protocol for the connection with the computer. It would ask for video data in as close a resolution as the computer could give it to the resolution of the LED display, receive the data, convert it to the frame format our LED display driver accepts, and send it. Thus, our LED display would work plug and play style with most computers. This conversion module could be implemented on an FPGA. This route would not only be a silver bullet for frame rate issues and audio syncing, but would also bring the system one step closer to being readily usable for entertainment purposes. I believe one person could accomplish this over the course of a semester.

Another logical step for the project is creating a modular system based on nodes of 2x2 or 4x4 LED panels. This way a compact power supply could be designed for each node, as well as cases to hold each node together. 2x2 nodes are probably a better route at this point, as each node would be cheaper and more manageable for hobbyists. Also our current FPGA cannot support the number of logic gates required to drive 16 panels, but has been shown to work excellently for 4 panels.

Other considerations for the future include timing constraints analysis of VHDL, designing a framing system to build variable size aggregations of nodes that can safely stand or be hung, and weatherization.