System Requirements Specification

For

LED Video Player

Version 1.2

Prepared by Tell O'Neal & Alexander Elklund

University of Idaho Capstone Design

Last modified March 13th 2014

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Tell O'Neal	02/18/14	Initial Version	1.0
Tell O'Neal	02/26/14	Updated To Add Requirements for FPGA	1.1
Tell O'Neal	03/13/14	Updated Requirements for software interfaces	1.2

1. Introduction

1.1 Purpose

This document defines all software, hardware, and system level specifications that will guide the design of the FPGA Driven LED Matrix. This project contains several distinct subsystems, but each subsystem will be fully covered by this SRS document. This SRS document will specify all requirements of the system but will not describe how the system must be designed, nor contain the details of the design. The System Design Document (SDD) will contain all design choices and details. In addition the SDD may contain non-critical low level specifications not addressed in this SRS.

1.2 Document Conventions

Words ***bolded and italicized*** in standard sections of this document can be found in Appendix A, the glossary at the end of this document with an accompanying definition.

1.3 Intended Audience and Reading Suggestions

This document is intended to be used by the current developers of this project, Alex Elkland & Tell O'Neal to guide the design of the system. The document is also intended to be read and approved by the customer of this project, Dr. Robert Rinker of the University of Idaho, as well as Professor Bruce Bolden of the University of Idaho who is overseeing the project.

In addition to the current audience named above, this document may also be used as reference, or modified by future senior design groups that extend work done on this project.

All members of the intended audience are assumed to have a sufficient background in Computer Science or Engineering and this document will not attempt to provide supplementary explanations for terms and components that would be considered common knowledge for a person with such background.

This document does acknowledge that there is a sufficient hardware component in the system due to the use of an FPGA. If the reader is not familiar with ***FPGAs***, ***RTL***, and ***HDL*** languages, it is recommended that the reader fully view the Product scope and Overall Description sections of this document prior to reading other sections.

1.4 Product Scope

The system being developed is an ***LED*** Matrix capable of displaying video. 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, however the system will be developed so that it is capable of easily being expanded to drive a rectangular n x m 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.

1.5 References

Team RPLD Design Documents:
http://seniordesign.engr.uidaho.edu/2012-2013/raspberry_pi/documentarchive.html

Team Automaten Design Documents:
http://mindworks.shoutwiki.com/wiki/LED_Animation#Team:_Automaten

2. Overall Description

2.1 Product Perspective

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.

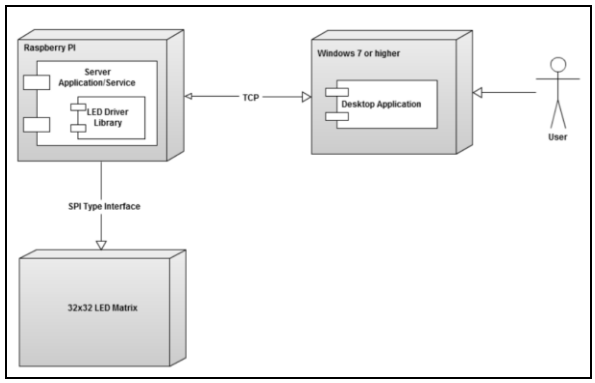


Figure 1- Team RPLD System Overview

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.

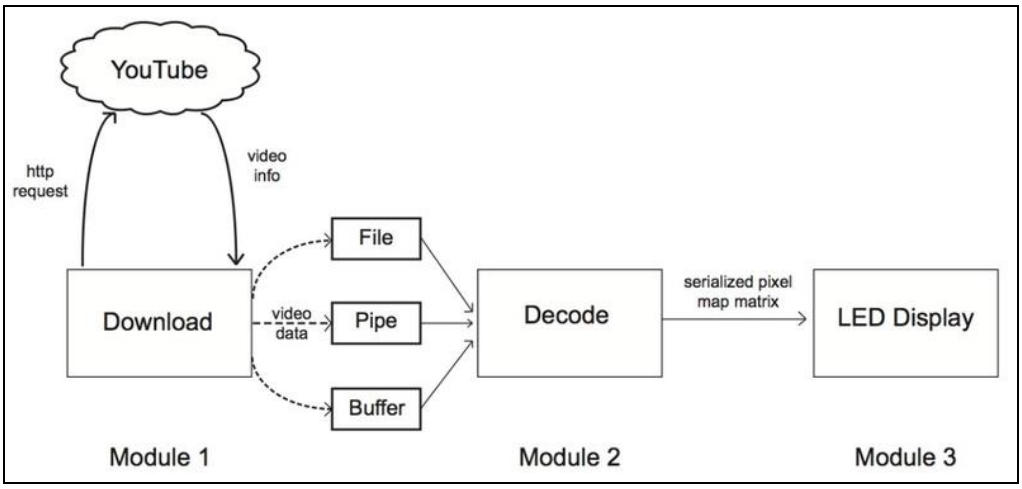


Figure 2- Team Automaten System Overview

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.

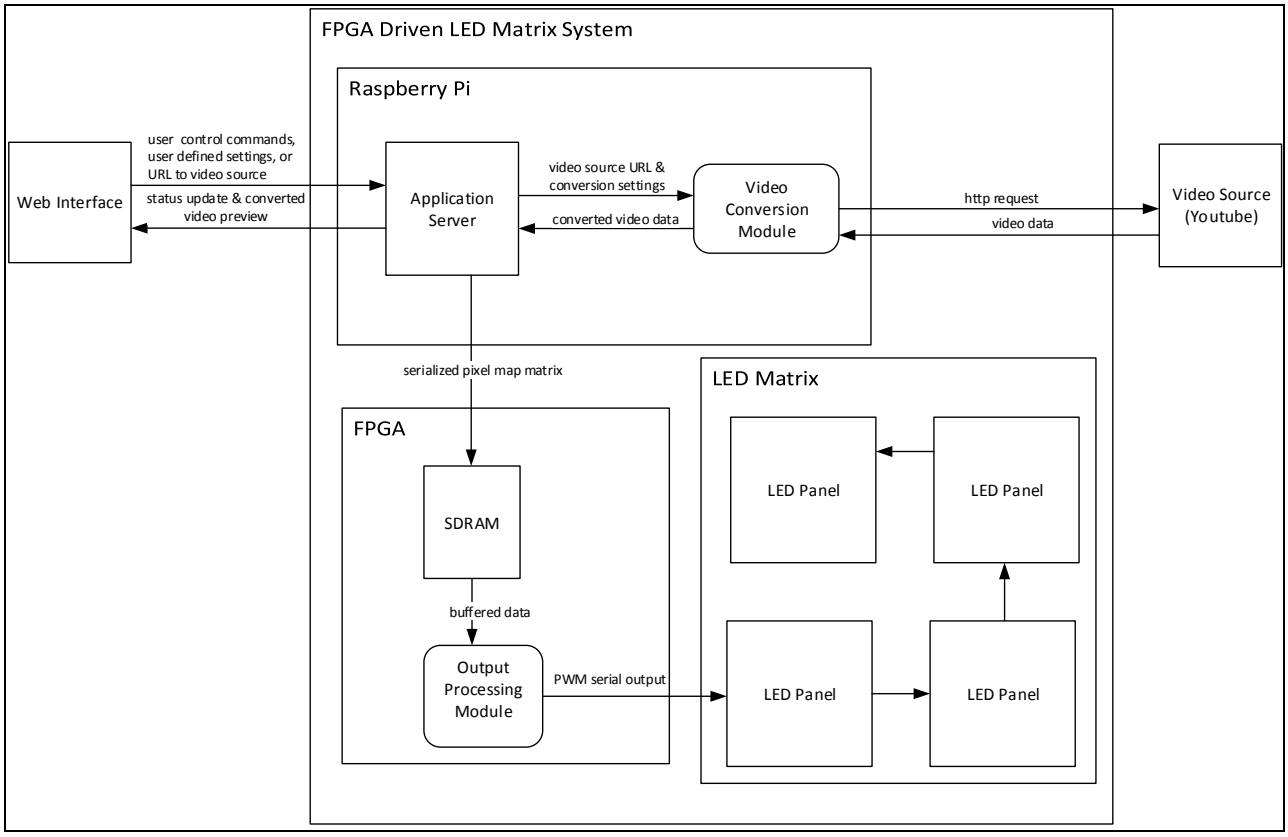


Figure 3- Current System Overview

2.2 Product Functions

LED Matrix Functions:

- Display Video on an LED matrix consisting of a minimum of 64 panels (8x8 array)
- Display Video with a minimum of 12 bit color (4096 distinct colors)
- Display Video at a minimum of 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

2.3 User Classes

- Dr. Rinker
- Future Design Groups
- Hobbyist duplicating our design.

2.4 User Documentation

The system shall include the following documentation:

- Guide for installing software dependencies, software application, and connecting hardware
- The software application shall have an internal help file that can be accessed while the user is operating the software through the user interface

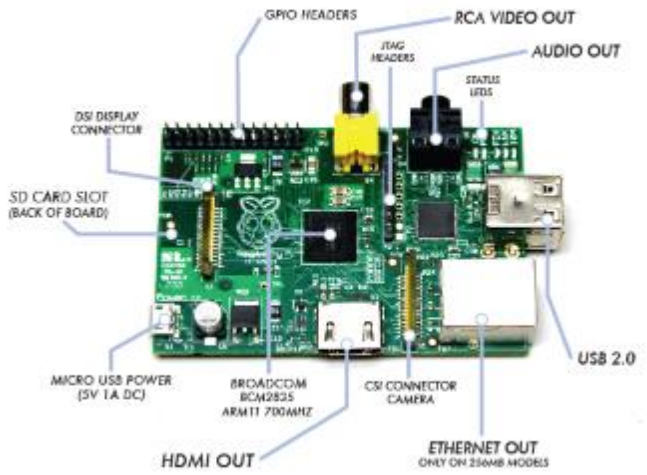
The documentation will be delivered to the customer in printed form and also accessible via the project wiki page.

Operating Environment

2.4.1 Operating Environment for Software

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

2.4.2 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

Assumptions and Dependencies

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

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

3. External Interface Requirements

3.1 User Interfaces

To support control of the system from as many types of devices as possible, the interface to the system will be accessed through a web interface. The interface will be provided by an application server running on the Raspberry Pi and accessible via http over port 80 or 8080.

The interface will be developed in HTML5 and CSS3. The interface should be accessible and fully functional from any modern device with a web browser supporting HTML5 and CSS3.

Particularly the interface should render, and perform accurately on the following devices:

- iOS Devices with Safari 6 or later (iPhones, iPads)
- Android 4.0 Devices with Google Chrome and Later
- Windows, Apple, and Linux computers with the following web browsers:
  - Internet Explorer 9+
  - Google Chrome
  - Safari
  - Firefox

The layout of the interface should be responsive to the resolution of the device and scale correctly on devices with a minimum horizontal resolution of 320px.

3.2 Hardware Interfaces

3.2.1 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 Serial Peripheral Interface (SPI)

The SPI bus will require 4 GPIO Pins to be used on the Raspberry Pi and 4 I/O pins on the FPGA.

Raspberry Pi Signal Name	**Data Flow Direction**	**FPGA Signal Name**	**Voltage Level**
SCLCK	→	SCLCK	3.3 V
MOSI	→	MOSI	3.3 V
MISO	←	MISO	3.3 V
SS (Slave Select)	→	SS (Slave Select)	3.3 V

3.2.2 FPGA Dev Board Interface to LED Panels

The FPGA also must interface directly to the first LED Panel. The interface is also a type of SPI (serial) bus, in that additional LED Panels are chained to the output of the first Panel. 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

3.3 Software Interfaces

3.3.1 Web Based GUI interface to Application Server

The Web Based GUI interface will be a client side application that is self contained in the browser. The interface to the Application server shall be a full-duplex asynchronous connection.

- The connection between the Application Server and clientside GUI application will be made using the Web Socket protocol over a TCP connection.
- Data transmitted across the web socket will be sent as JSON formatted objectes.

3.3.2 Application Server to LED Panel Driver

The Application server will interface to the LED Panel Driver be accomplished use TCP socket. This will allow the application server and the LED Panel Driver to be located on separate computers if needed. (Raspberry Pi only Runs LED Panel Driver, Networked PC runs Application Server)

3.4 Communications Interfaces

3.4.1 Raspberry Pi Network Connection

This system requires that the Raspberry Pi must be connected to an Ethernet network with Internet access. Internet access must be present to download videos from YouTube or other sources.

- The Raspberry Pi must be configured to connect to a wired Ethernet network via the onboard RJ45 connector **or**

  The Raspberry Pi may be connected to a Wireless Ethernet network via an 802.11.n WIFI adapter connected to a USB port on the raspberry pi.

3.4.2 LED Panel Driver SPI Interface to FPGA

The communication interface between the LED Panel Driver and the FPGA will be implemented using an SPI (serial peripheral interface) protocol.

The Raspberry Pi will function as the master device.
The FPGA will function as a slave device.

4. System Features

This section describes the system features required to fulfill the use cases listed in the diagram below.

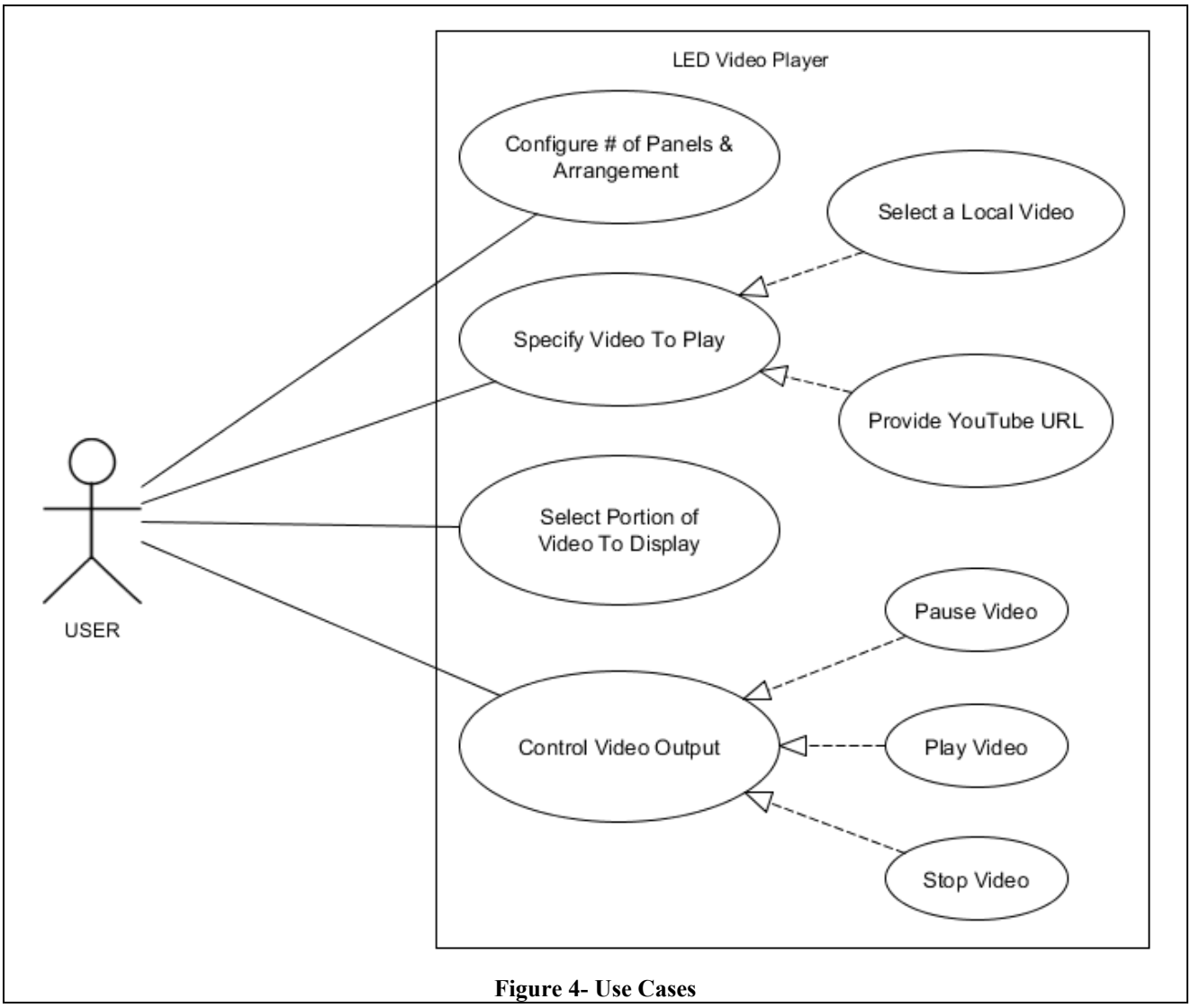


Figure 4- Use Cases

4.1 Feature 1: Ability to Configure LED Panels

4.1.1 **Description:** The user will be able to specify the number of LED panels that the software will control. The user shall be able to configure the arrangement by specifying the number of rows of panels x the number of columns of panels. The configuration of panels will be displayed so that the user can easily verify the panels are correct

4.1.1 **Priority**: High

4.1.2 **Stimulus/Response Sequences**

User logs into web interface -> Clicks Up or Down on Row or Col Widget -> Panel Arrangement Updates to show arrangement

4.1.3 **Functional Requirements**

REQ-1: Default Arrangement: 2 rows x 2 cols
REQ-2: Web Interface dynamically displays row and column array of panels immediately when user increases or decreases rows or cols in a panel

4.2 Feature 2: Ability to Specify Video Source

4.2.1 **Description:** The user will be able to specify the source of the video by pasting a YouTube url to a video that will be downloaded, or selecting from a list of previously downloaded videos.

4.2.1 **Priority**: High

4.2.2 **Stimulus/Response Sequences**

User logs into web interface -> User Pastes YouTube URL into input field OR User selects previously downloaded video from list -> System Downloads video and updates interface to show video OR Video updates interface to show selected video.

4.2.3 **Functional Requirements**

REQ-1: Input only accepts valid YouTube URL and warns user if URL is not valid

4.3 Feature 3: Ability to Select Portion of Video to Display

4.2.1 **Description:** The user will be able to select a portion of the video that will be sent to the display. The mechanism for selecting video will be a standard crop type control found in most image editing programs.

4.2.1 **Priority**: Medium

4.2.2 **Stimulus/Response Sequences**

User logs into web interface -> User Specifies Video Source-> Interface updates to display video preview-> Crop Box appears over video-> User Drags crop handles to specify Video area to use

4.2.3 **Functional Requirements**

REQ-1: Crop box ratio must be linked to Panel Arrangement so only valid aspect ratio of video may be selected

REQ-2: Crop box updates dynamically when Panel Rows or Columns Changes

4.4 Feature 4: Ability to Control Video Output

4.2.1 **Description:** The user should be able to send the video to the device and while the video is playing, the user can pause the video (which allows it to continue playing from where it is paused if play is pressed again), or stopped if the user presses stop.

4.2.1 **Priority**: Medium

4.2.2 **Stimulus/Response Sequences**

User logs into web interface -> User Specifies Video Source-> Interface updates to display video preview-> User Crops Video to Desired Selection-> User Clicks Play ->User Clicks Pause or Stop-> User Clicks Play If Paused… etc..

4.2.3 **Functional Requirements**

REQ-1: User Should not be able to press buttons until a video source is defined and selected.

REQ-2: Video should begin playing with less than 1s delay from time play button is pressed.

REQ-3: Video should respond to clicks of PAUSE, PLAY, or STOP in 1s or less.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

TBD

5.2 Security Requirements

The Web Interface to control the display shall require user authentication to prevent unauthorized users from controlling display – **FUTURE**

5.3 Software Quality Attributes

The software should be sufficiently validated to ensure it satisfies the major defects and is free from errors that prevent the user from using the system as intended, or without additional hardship. Each sub-module shall be validated, and the modules should be integrated and tested at a system level.

Appendix A: Glossary

FPGA:

RTL:

HDL:

Raspberry Pi:

SPI:

MOSI- Master Output, Slave Input:

MISO- Master Input, Slave Output:

SS- Slave Select, active low: