Dual-Pi Visualizer Design Document

Table of Contents

1 Introduction

This document details the design considerations for the Dual-Pi Visualizer project, including its constraints and dependencies, methodology, a high-level system overview, and an overview of the system architecture design. The introductory section details the scope and purpose of this document, its intended audience, and definitions of some terminology used throughout.

1.1 Purpose and Scope

The purpose of this document is to provide the details of how the Dual-Pi Visualizer project will achieve the functionality required of it.

1.2 Target Audience

The target audience for this document are stakeholders and developers. This document is intended for all who are involved in the completion of this project, and therefore, its scope is broad.

1.3 Terms and Definitions

The following terms will be utilized throughout this document.

Term	**Definition**
User	A client that interacts with the Pi
Use Cases	The interaction between the product and the users of the product.
Stakeholder	A person that is involved in the system, that is not part of development.
Milestone	The date on which a work product will be completed.
Deliverable	A measurable outcome of the project.
GUI	Graphical User Interface
Functional Requirement	An action that the product must be able to perform.
Nonfunctional Requirement	Specifications of the properties of the product.

2 Design Considerations

This section is for describing the considerations that will need to be taken in designing the Dual-Pi Visualizer project in order for it to meet all of its functionality requirements.

2.1 Constraints and Dependencies

In order to meet the minimal requirements for the creation of the Dual-Pi Visualizer project, it must:

- 222
- Sync up with at least one other Raspberry Pi, at which point both will be performing the exact same actions.
- Be capable of being controlled solely with the various buttons available on a standard wireless mouse, with no interface.
- Automatically power down between 7am and 7pm.

2.2 Methodology

The development of the Dual-Pi Visualizer project will require the usage of VLC's API in order to deal with a majority of the program's functionality, such as populating video-based playlists automatically, trimming intros and outros from videos greater than a certain length, and playing them in full-screen mode across two different devices simultaneously.

This project will also require the usage of shell commands from within it in order to automatically power itself down between 7am and 7pm.

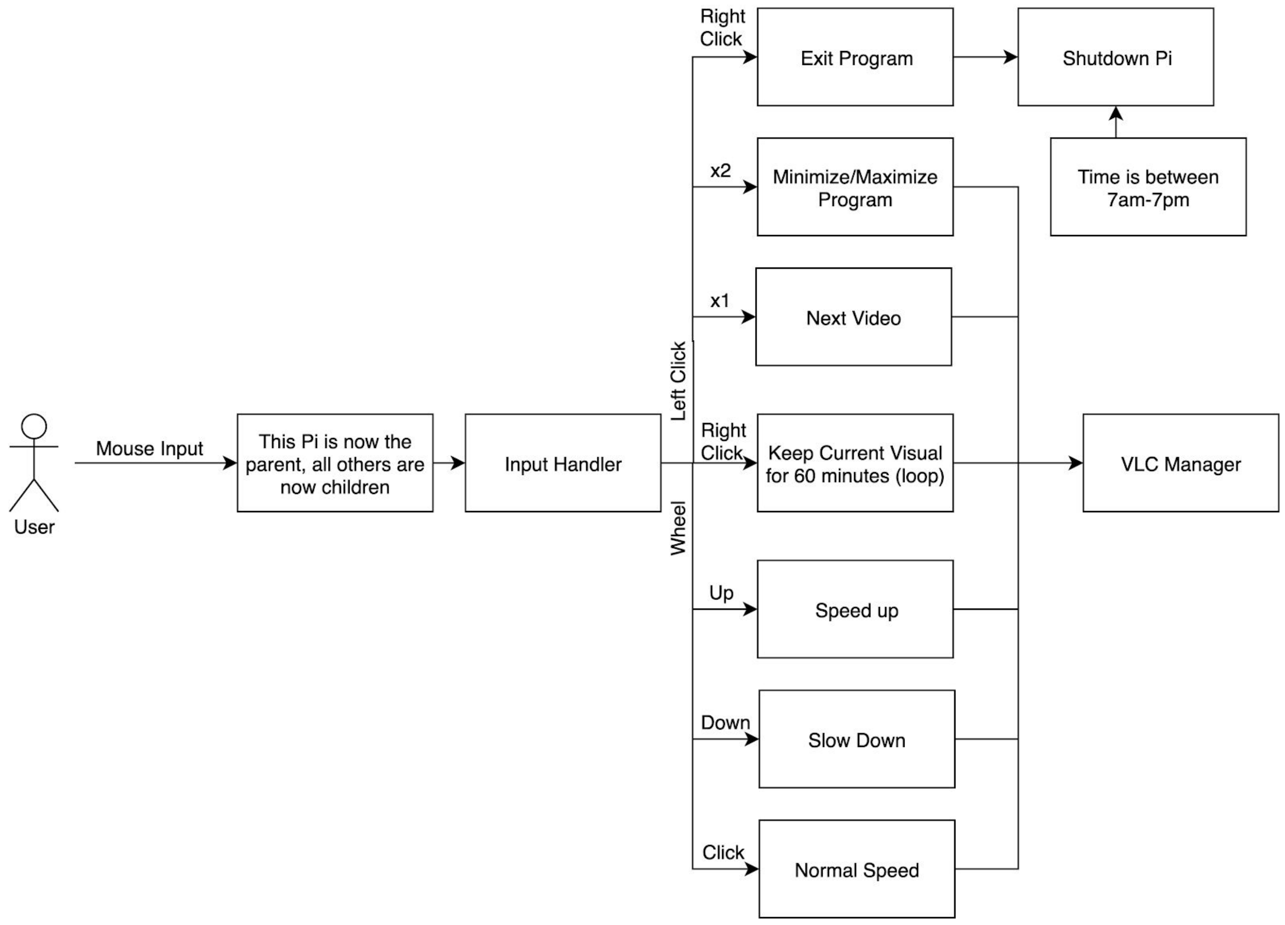
3 System Overview

The diagram below shows a high-level abstraction of the entire Dual-Pi Visualizer project, and its purpose is to prepare one for the system architecture portion of this design document. The system that makes up the entirety of the Dual-Pi Visualizer project is comprised of the following components:

- **Parent Handler** - Necessary for dealing with having multiple Pi units carrying out the same tasks. Whichever Pi unit most recently received mouse input becomes the parent, and all other Pi units are children which perform the same actions as the parent.
- **Input Handler** - Deals with the various key bindings available to the user.
- **VLC Manager** - Deals with VLC API and handles playlist populating, video trimming, audio visualization, and turns user input into actual VLC-related actions such as proceeding to the next video in the playlist.

- **Shutdown Handler** - Shutdown can occur due to power loss, pressing the left and right mouse buttons at the same time, or due to the time being between 7am-7pm. We need to ensure that shutdown is handled properly.

There's a bunch of other components not in this overview or the diagram below that will show up in the next section.

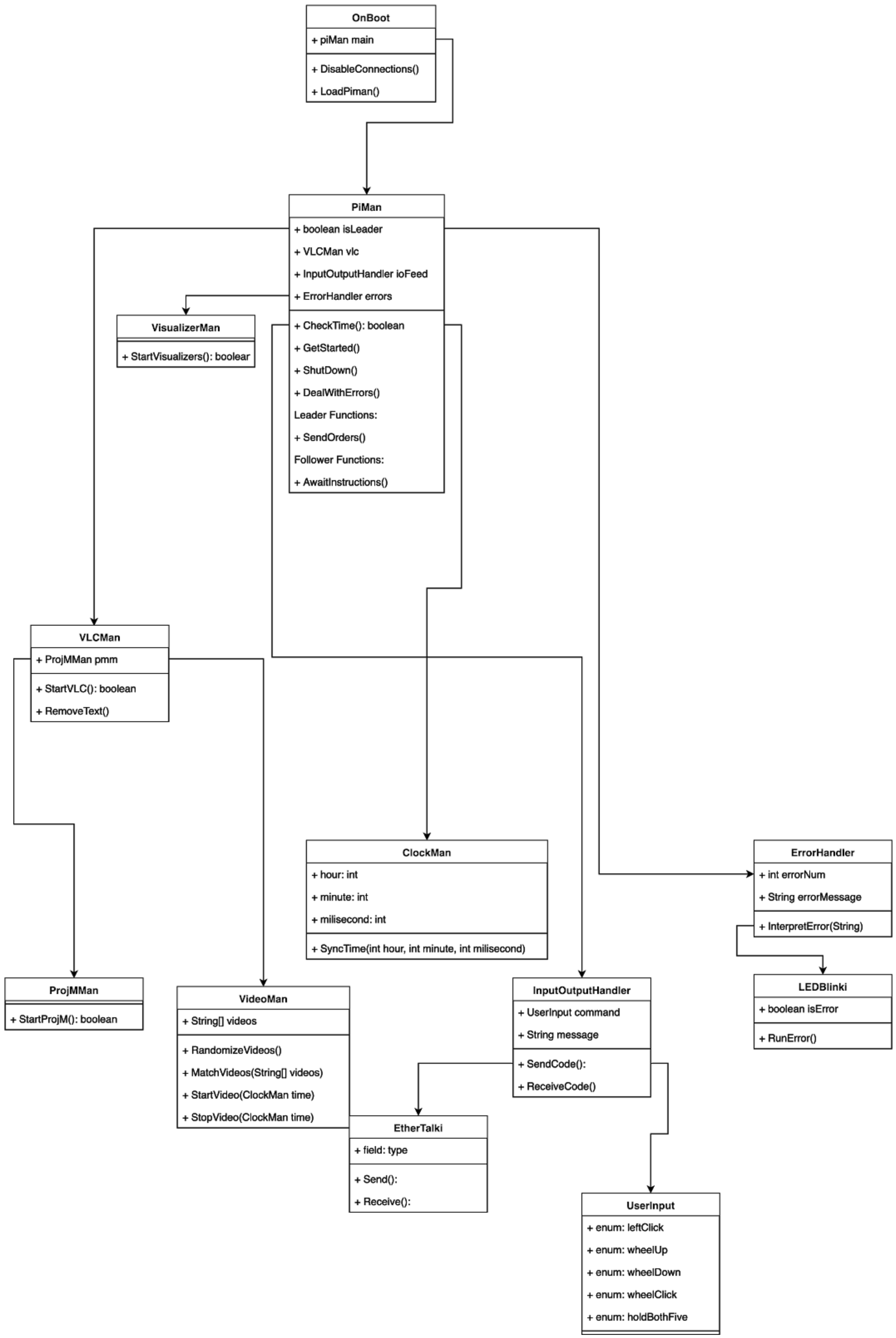


4 System Architecture

As briefly introduced in the previous section regarding the Dual-Pi Visualizer project's system overview, this section dives deeper into these systems and the components that make up these systems on a lower-level scale.

4.1 Overview

Below is a diagram of the system architecture plan. Beyond this section is a detailed explanation behind each of the components in this diagram.


OnBoot
+ piMan main
+ DisableConnections()
+ LoadPiman()
PiMan
+ boolean isLeader
+ VLCMan vlc
+ InputOutputHandler ioFeed
+ ErrorHandler errors
+ CheckTime(): boolean
+ GetStarted()
+ ShutDown()
+ DealWithErrors()
Leader Functions:
+ SendOrders()
Follower Functions:
+ AwaitInstructions()
VisualizerMan
+ StartVisualizers(): boolear
VLCMan
+ ProjMMan pmm
+ StartVLC(): boolean
+ RemoveText()
ClockMan
+ hour: int
+ minute: int
+ milisecond: int
+ SyncTime(int hour, int minute, int milisecond)
ErrorHandler
+ int errorNum
+ String errorMessage
+ InterpretError(String)
LEDBlinki
+ boolean isError
+ RunError()
ProjMMan
+ StartProjM(): boolean
VideoMan
+ String[] videos
+ RandomizeVideos()
+ MatchVideos(String[] videos)
+ StartVideo(ClockMan time)
+ StopVideo(ClockMan time)
InputOutputHandler
+ UserInput command
+ String message
+ SendCode():
+ ReceiveCode()
EtherTalki
+ field: type
+ Send():
+ Receive():
UserInput
+ enum: leftClick
+ enum: wheelUp
+ enum: wheelDown
+ enum: wheelClick
+ enum: holdBothFive

4.2 OnBoot

Handles the startup of our program and handling of any processes that must take place in preparation. The OnBoot will disable all wifi and bluetooth connections to the Pi unit. This will begin with the powering up of the RaspberryPi, without any user inputs needed.

4.3 PiMan

PiMan manages all functionality of the Pi and our program's processes post-boot. Retains boolean representation of status as "Leader" or "Follower", and will run all functionality based on that designation. If "leader", listens for mouse inputs and sends commands out through ethernet via the InputOutputHandler. If "follower", listens for ethernet data passed from the InputOutputHandler and follows those commands.

4.4 VLCMan

The VLC Manager component will deal directly with the VLC API necessary to populate a playlist with the videos in the folder we direct it to, cutting the intro and outro of videos which exceed a certain length, and translate user input into VLC-specific commands such as skipping to the next video, or altering the speed at which the video is playing.

4.5 ProjMMan

The ProjectM Manager will deal directly with the ProjectM audio visualizer. Starting/stopping visualizations. Starting/stopping music file playback (used only to effect visualization effects).

4.6 VideoMan

Manages Videos through the VLC program. Creation of the video playlist. Starting/stopping videos. Determining where in videos to start playback. Disabling of all captions and text in videos.

4.7 InputOutputHandler

Manages EtherTalki and UserInput, passing messages back to PiMan.

4.8 EtherTalki

Receives/interprets input messages from other Raspberry Pis through the ethernet connection and passes command back through the InputHandler. I/O between Pis go through TCP/IP network utility. Leader sends status.

4.9 UserInput

Receives/interprets input messages from user’s mouse/keypad control and passes command back through the InputHandler.

4.10 ErrorHandling

Interprets Error codes for the PiMan and passes back the results for PiMan to handle. Indicates fatal/significant errors displayed through physical LEDs on the Pi unit.

4.11 LEDBlinki

The lightup of LEDs physically attached to the RaspberryPi unit that will indicate error status of our program.

4.12 ClockMan

Manages information from the physical clock unit. Has a PowerDown boolean that is true if the time is between 7am and 7pm, to let the PiMan know to go into shut down mode.