JavaTheHutts

**Project Thisplay**

Software Design Document

Name (s):

Jocelyn Mallon

Michael DeSantiago

Shrey Gahlawat

Erick Barron

Fall 2023 Comp 490/L

Instructor:

Edmund Dantes

Date: 12/15/2023

**TABLE OF CONTENTS**

***1.0*** ***INTRODUCTION 4***

***1.1*** ***Purpose 4***

***1.2*** ***Scope 4***

***1.3*** ***Overview 4***

***1.4*** ***Definitions and Acronyms 4***

***2.0*** ***SYSTEM OVERVIEW 4***

***3.0*** ***SYSTEM ARCHITECTURE 4***

***3.1*** ***Architectural Design 4***

***3.2*** ***Decomposition Description 5***

***3.3*** ***Design Rationale 5***

***4.0*** ***DATA DESIGN 5***

***4.1*** ***Data Description 5***

***4.2*** ***Data Dictionary 5***

***5.0*** ***COMPONENT DESIGN 5***

***6.0*** ***HUMAN INTERFACE DESIGN 5***

***6.1*** ***Overview of User Interface 5***

***6.2*** ***Screen Images 6***

***6.3*** ***Screen Objects and Actions 6***

***7.0*** ***REQUIREMENTS MATRIX 6***

# **INTRODUCTION**

## Purpose

This software design document describes the architecture and system design of ‘Project Thisplay’ our wireless ‘smart’ display and accompanying client applications.

## Scope

Project Thisplay is a ‘smart’ display (i.e. a graphical output device) that users can access over local networking (wifi/ethernet) or BLE, and show custom images and text. The display firmware is intended to run in two different modes, a full-featured/stand alone mode for deployment on Raspberry Pi 4B/5 models, accessible via any Chromium based web browser, and a lighter weight version that runs on the Raspberry Pi Pico W, and requires the use of separate Web and/or iOS apps.

## Overview

This SDD will outline and explain the structure of both versions of the firmware, along with the separate web and iOS applications. This document will include class diagrams, flowcharts, and UI mockups, for each separate component, broken down by firmware/client app categories, then further broken down into subcategories as necessary (e.g. separate iOS and web app subcategories under the client app category.)

## Definitions and Acronyms

BLE - Bluetooth Low Energy

WIFI - 802.11n compatible 2.4ghz wireless networking.

LCD - Liquid Crystal Display panel

EINK/EPAPER - Electronic Ink/Electronic paper display panel

RPi - Raspberry Pi 4B or 5 series

Pico/RP2040 - Raspberry Pi Pico W

TBD - To Be Determined

WIP - Work In Progress

# **SYSTEM OVERVIEW**

Project Thisplay is a user friendly, programmable/customizable ‘smart’ display, that allows users to display custom images and/or text. The display consists of two main components, a physical LCD or EINK display panel, and an accompanying RPi or RP2040 ‘driver’ board. Depending on the driver board in use, the user may need to use a separate client application (i.e. the web or ios apps) for interacting with the display. The driver

# **SYSTEM ARCHITECTURE**

## Architectural Design

This section is largely finalized, though the actual diagrams/figures are still WIP. A general explanation of the architecture follows, to be replaced with more diagrams as they’re finalized over winter break.

* Firmware design:
  + The firmware can be broken into two categories, full-featured, intended for use with the RPi 4B/5+ models, and the lightweight/barebones, intended for use with the rp2040.
  + The full featured firmware effectively bundles both the ‘barebones’ firmware, and client web app together, and runs both locally on the RPi. Depending on the configuration, it can expose both the web app, and the barebones firmware endpoints to local network connections over https server, and BLE respectively.
  + The barebones firmware is intended to accept minimal data uploads from the client applications, over BLE or local networking, according to a pre-defined JSON schema, which communicates the type of display panel in use, the display resolution and color depth, amount of free storage on device, etc.
  + The barebones firmware is composed of three main modules:
    - The ‘**Output’** module handles actually outputting to the display, it simply reads file path and time/duration information from a local (on device) JSON store, and displays the file on the screen for the specified amount of time.
    - The ‘**Persistence’** module handles creation of local file storage, and appending new entries to the on device JSON store, to later be read by the Output module.
    - The ‘**Server’** module, accepts the incoming JSON configuration data, along with any file binary data, then verifies any/all file integrity using SHA1 hashes (matched against provided hashes in the JSON data), before handing the file and duration data off to the Persistence module.
* Client Apps:
  + The client applications consist of two separate applications, one a python web app, the other an iOS app, using SwiftUI.
  + The client applications are intended to be functionally identical, but will utilize local/native frameworks and controls where possible, so their UI’s will NOT be identical.
  + The client applications will consist of two base modules, ‘Communication’ and ‘Editing’ though there are some extensions for both, e.g. BLE and HTTP modules for communication, and Text and Image modules for the editor.
    - The ‘**Communication’** module will handle any network-level communication with the firmware/driver board. This also includes the computing of SHA1 hashes for binary data, and creation of the JSON config data to be sent to the firmware/driver.
    - The Editor module will handle the user input, file processing and verification in respect to device display resolution and color depth, along with presenting minimal image editing/compositing options, including cropping, flipping and/or rotating images, and adding text layers., etc. Final options are contingent on time and may include ‘stretch’ goals not outlined here.

A diagram of a software application

Description automatically generated

Figure 2. Basic overview of the client app interaction with the hardware, when used on the rp2040 driver board.

## Interface Design

TBD/WIP – mockups have been under slow progress for several weeks, but nothing is really finalized enough at this point. Again, over winter break with no assignments/projects for other classes, we should be able to hone in on the finalized iOS and web app mockups.

## Decomposition Description

WIP – I think we’ve covered this already under section 3.1 so I’m not entirely sure what should be moved here, outside of the final diagrams once they’re completed and exported from OmniGraffle.

## Design Rationale

The main decision for the architecture design was to allow for flexibility and user choice. During the COVID19 pandemic, it became increasingly harder to find full-fledged Raspberry Pi models, so we wanted to ensure that the most basic functionality can run on the rp2040, which is widely available at a low cost of ~$10 USD. Based on the limited compute power and flash storage of the rp2040, it made sense to avoid relying on a relational DB, hence we opted to use local JSON files for data persistence instead. This then allows us to easily bundle the web app along with the barebones firmware when running on the more powerful RPi 4/5 models, while decoupling the editing and more compute heavy client application functions when running on the lower power rp2040.

# **DATA DESIGN**

TBD/WIP – This is WIP as we’re waiting for the arrival of a new/alternate EINK display panel. The original purchased panel while functional used a non-standard 50-pin connector and we had to replace it with a standard 24-pin model. Once the panel arrives during winter break and we confirm that it operates comparably to the original panel, we should be better able to flesh out the data design.

# **COMPONENT DESIGN/DETAILED DESIGN**

## Class Diagrams

## TBD/WIP - Progress on this section was blocked due to the holidays and finals in other classes, but will be worked on during the winter break by team lead Jocelyn, and finalized early in Comp 491/L. See section 3.1, architecture design for the overview of how we’ve designed the project. Note from team lead/Jocelyn: we had some basic UML started in draw.io, but frankly I’m not very happy with them and I find the web interface frustrating and slow to work with, so now with a few weeks of no other classes, I intend to redo them with OmniGraffle over winter break.

# **HUMAN INTERFACE DESIGN**

## Overview of User Interface

TBD/WIP - As with section 3.0 and 4.0, this is maybe 60% complete, but final interface design hinges on the verification of the new/replacement eink panel, and just a bit more free time over winter break.

## Screen Images

TBD

## Screen Objects and Actions

TBD

1. **REQUIREMENTS MATRIX**

TBD/WIP - This section goes hand in hand with Section 3 & 4, and is currently being in tracked and fleshed out in a separate excel spreadsheet. This should also be finalized over winter break, before 491/L starts.