COMP 490/L Senior Design Project Fall 2023 490/L Project Presentation

Group Name: JavaTheHutts
Team Leader: Jocelyn Mallon

CSUN

Agenda

- Project Overview
- Problem Definition
- Application screenshots
- Software overview / Diagrams
- Sample Code Snippets
- Progress and Future Work (491/L)
- Questions

Project Overview

- Create a cross platform (mobile and web based) application that allows users to upload/serve images and text to a wireless display, powered by several different Raspberry pi models.

- Create a simple smart display, accessible over bluetooth and wifi, using affordable, readily available off the shelf hardware.

- Create iOS and web-based applications to interact with/control the display.

Problem Definition

- Existing options for programmable displays are either expensive, suffer from barebones interfaces and/or hard to understand software.
- Multi-range use can help with marketing, decorating, information, etc.
- Our goal is to create an affordable and user friendly ‘smart’ display that will appeal to ‘tinkerers’ and non-tech-savvy users alike.

Technology

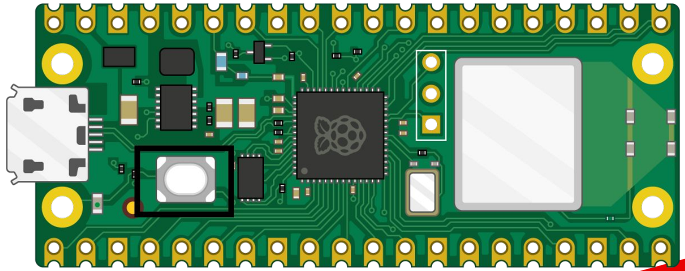
Given the combined hardware & software nature of our project, the technology used broadly breaks down into separate ‘firmware’ and ‘app’ categories, though there is some further differentiation/ segmentation based on the model of Raspberry Pi in use as well.

Technology, continued

On the Raspberry Pi 4B & 5, we're using native python and Flask to run our web interface on the Pi itself, allowing for almost anyone with a web browser to access the display, regardless of their device.

Technology, continued

On the Raspberry Pi Pico W, we run the standard micropython firmware with wireless modules to run a barebones (non Flask) http server on device, and additionally accept and respond to incoming BLE connections from the iOS and web apps.



CSUN

Technology, continued

iOS/Mobile app:

- The iOS application is built with Swift, using Apple’s SwiftUI GUI framework.
- Minimum iOS version targetted is iOS 16.
- Utilizes native controls and views, to be a ‘first class citizen’ on iOS.
- Stores previously configured device information using OS provided ‘defaults’ framework.
- Does not request unnecessary permissions.

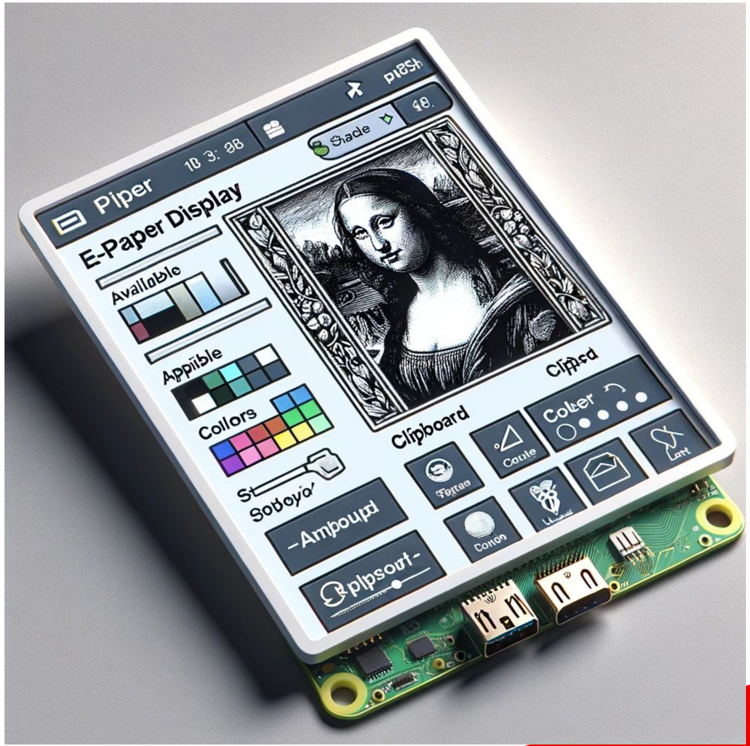
Frontend Vision

- Envision a sleek and clean UI which allows even the most tech illiterate users
- Our product name and logo will be placed on top with the sign in option placed next to it.
- The middle will have the prompt and button to either add a file or drag it if you are a web user.
- Under the upload button will be bars that will show the progress of the upload and once complete will either show that the file was uploaded successfully or an error message.

Frontend Vision Continued

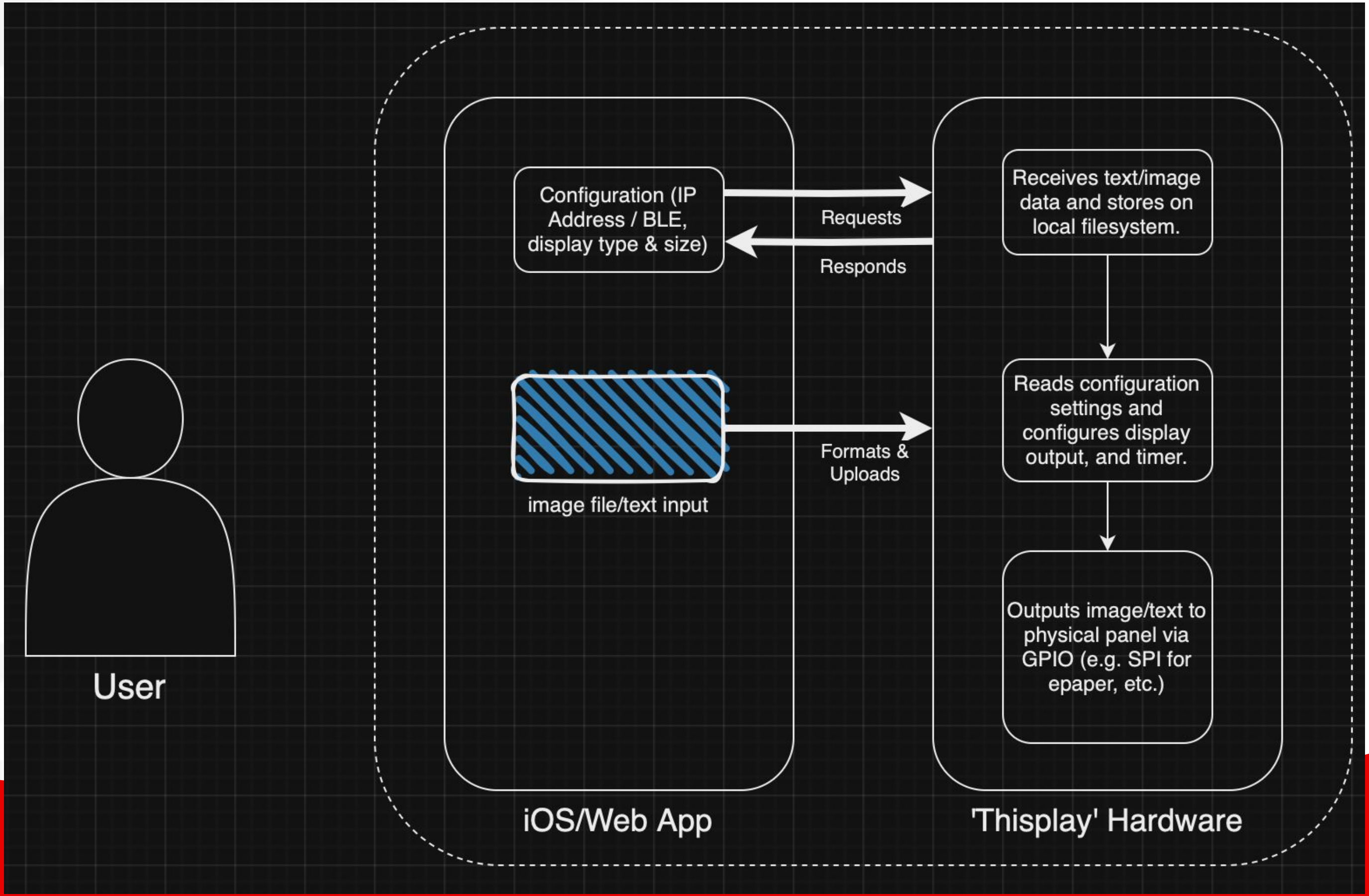
- Once in the user profile you will be able to see all your saved uploads with option to delete or favorite.
- If you choose an uploaded file you will be taken to another screen which will give you a wide range of options to customize your file as you see fit.(still thinking about all options, but think resizing, colo, etc.)
- And from there you may choose display the file to your e-paper display.

Application Screenshots (mockups)



CSUN

Software Overview

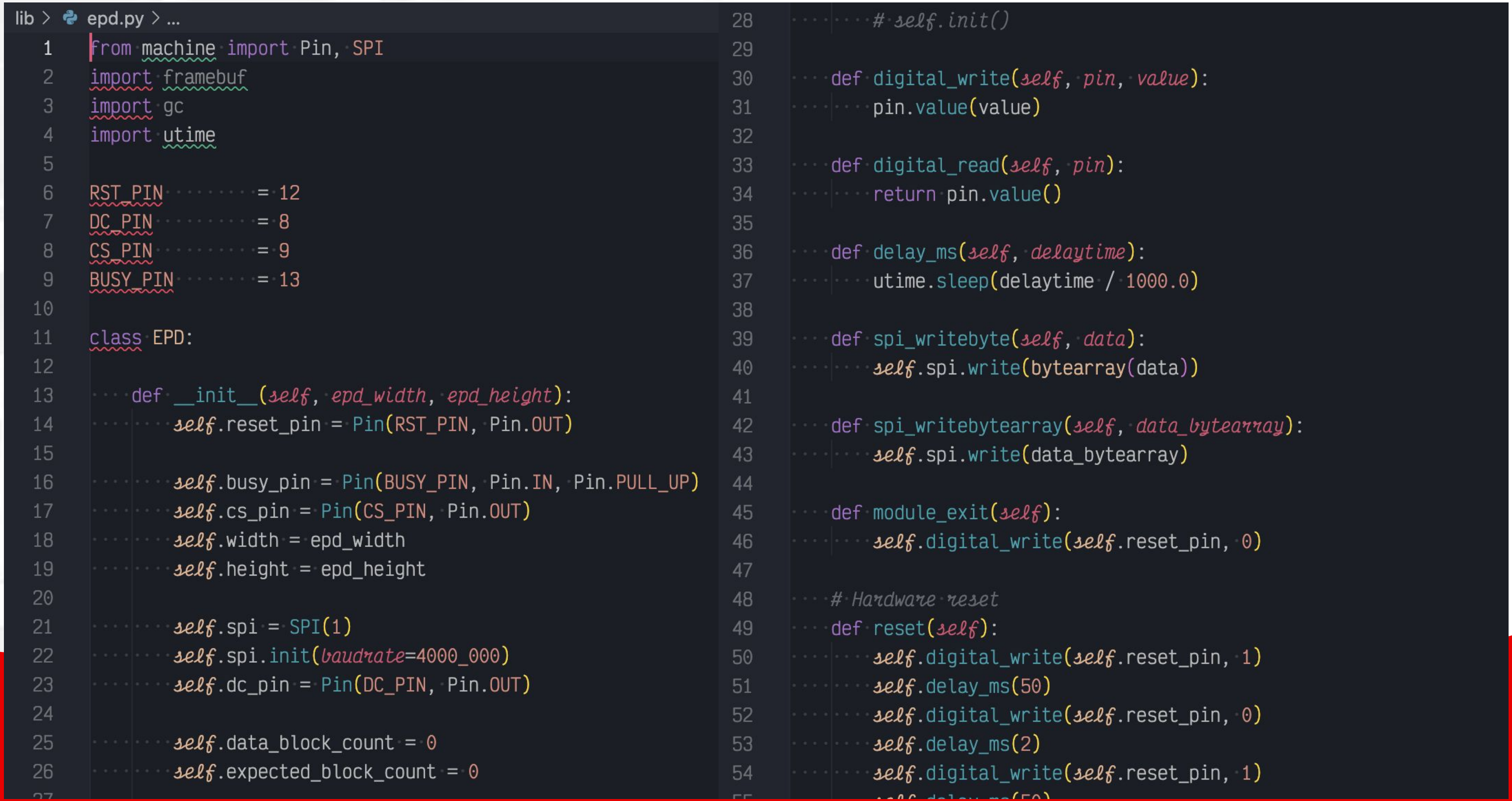


User
Configuration (IP Address / BLE, display type & size)
image file/text input
iOS/Web App
Requests
Responds
Formats & Uploads
Receives text/image data and stores on local filesystem.
Reads configuration settings and configures display output, and timer.
Outputs image/text to physical panel via GPIO (e.g. SPI for epaper, etc.)
'Thisplay' Hardware

CSUN

Sample Code Snippets

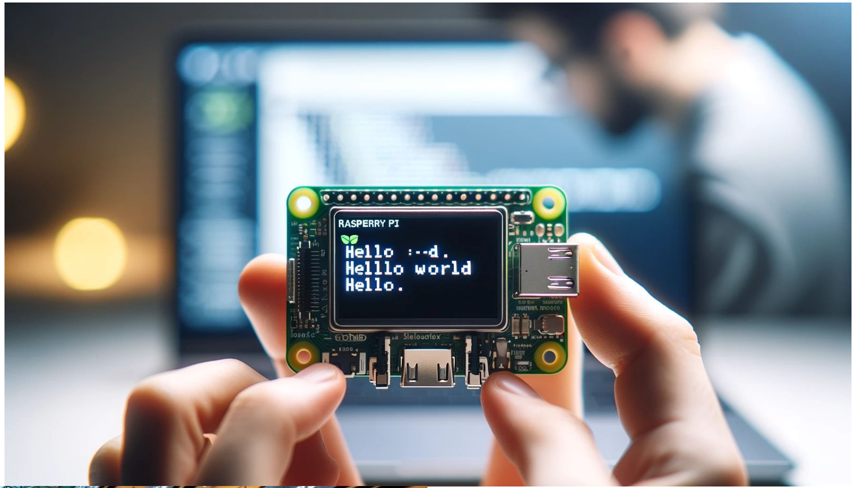
For our initial testing with the Pi Pico W & epaper display, we started by adapting existing code from the open source 'e-paper-frame' project on github: https://github.com/jtiscione/e-paper-frame

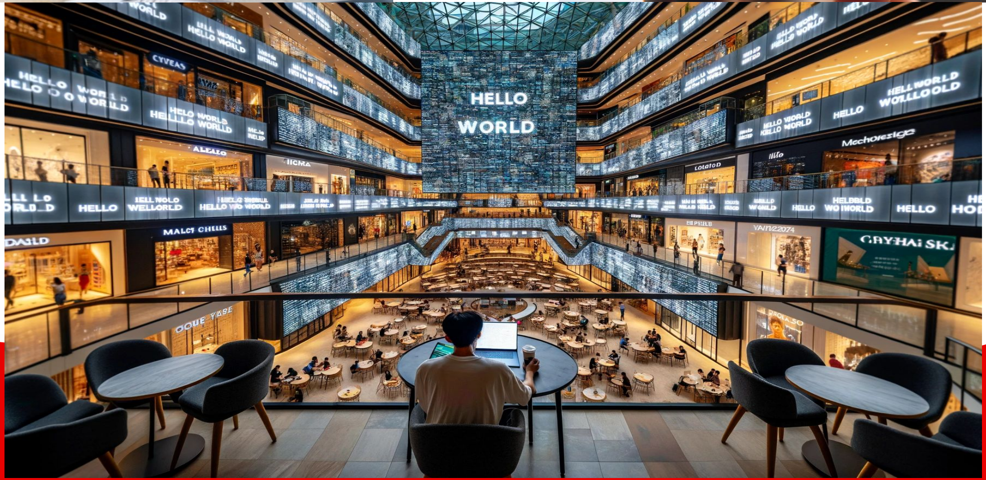


```python
from machine import Pin, SPI
import framebuf
import gc
import utime

RST_PIN         = 12
DC_PIN          = 8
CS_PIN          = 9
BUSY_PIN        = 13

class EPD:

    def __init__(self, epd_width, epd_height):
        self.reset_pin = Pin(RST_PIN, Pin.OUT)

        self.busy_pin = Pin(BUSY_PIN, Pin.IN, Pin.PULL_UP)
        self.cs_pin = Pin(CS_PIN, Pin.OUT)
        self.width = epd_width
        self.height = epd_height

        self.spi = SPI(1)
        self.spi.init(baudrate=4000_000)
        self.dc_pin = Pin(DC_PIN, Pin.OUT)

        self.data_block_count = 0
        self.expected_block_count = 0
```

```python
        # self.init()

    def digital_write(self, pin, value):
        pin.value(value)

    def digital_read(self, pin):
        return pin.value()

    def delay_ms(self, delaytime):
        utime.sleep(delaytime / 1000.0)

    def spi_writebyte(self, data):
        self.spi.write(bytearray(data))

    def spi_writebytearray(self, data_bytearray):
        self.spi.write(data_bytearray)

    def module_exit(self):
        self.digital_write(self.reset_pin, 0)

    # Hardware reset
    def reset(self):
        self.digital_write(self.reset_pin, 1)
        self.delay_ms(50)
        self.digital_write(self.reset_pin, 0)
        self.delay_ms(2)
        self.digital_write(self.reset_pin, 1)
```

App Visualization





CSUN

Progress and Future Work

Current Progress:

- Hardware has been ordered, delivered, and tested to verify it all works as expected.
- Samples from existing open source projects have been compiled and run to validate the proof of concept.
- Planning of software stack and architecture is largely complete.
- UI Mockups for web application and iOS application are still in progress.

CSUN

Progress and Future Work

Future Work (491/L):

- **GUI design needs to be finalized for iOS and web apps.**
- **JSON schema used to communicate hardware configuration back to client apps needs to be finalized.**
- **Most of the actual software coding for both firmware and client apps still needs to be done.**
- **Software Test Plan.**
- **Software Test Report.**

CSUN

Questions?

CSUN