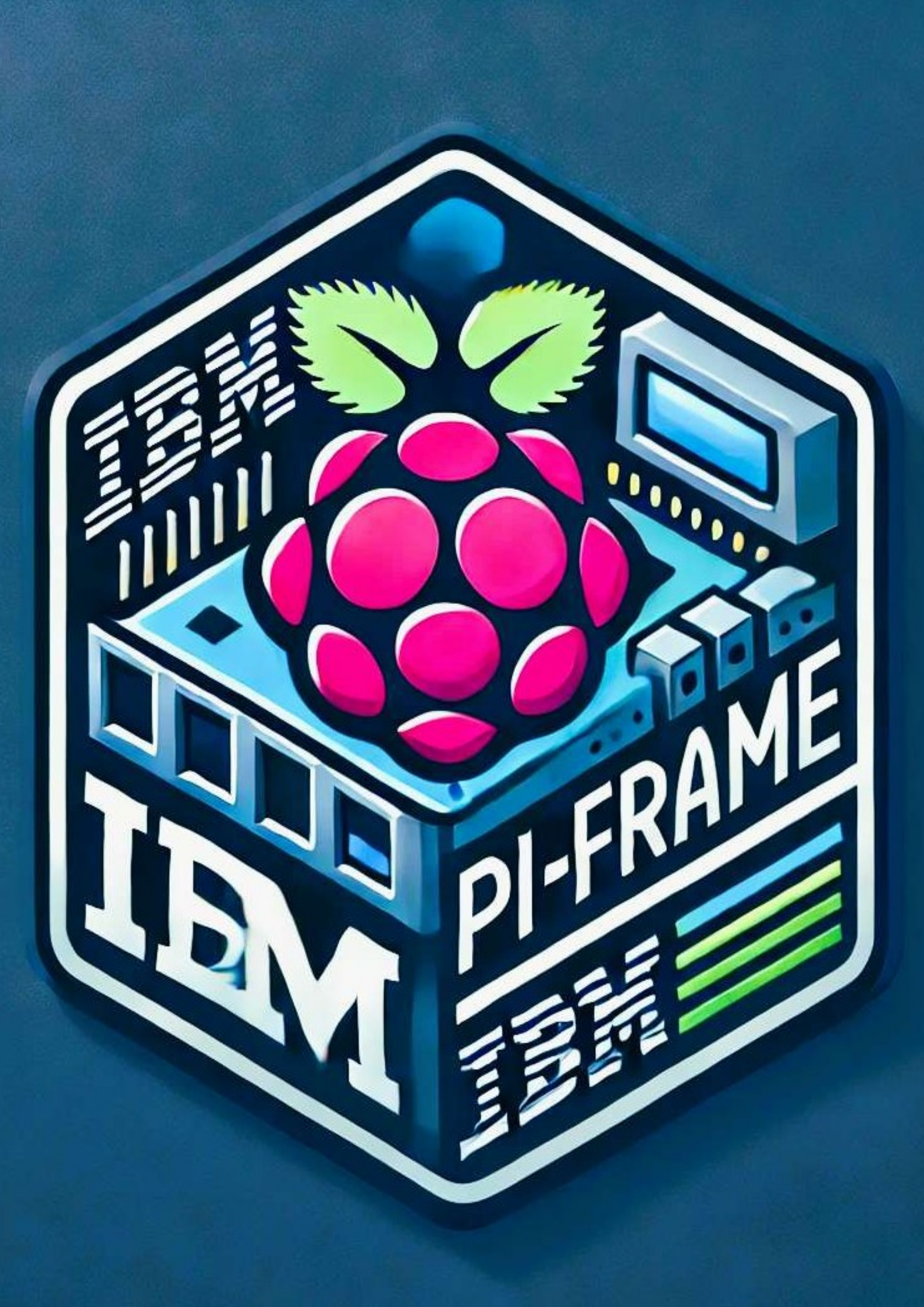

IBM
IBM
PI-FRAME
IBM

Table of Contents

Table of Contents 2
Introduction to the Pi-Frame 3
Why Build a PiFrame? 3
What You'll Find in This Manual 4
What you need – essential and cosmetic 4
Essential Parts 4
Parts for Decoration 4
Getting Started – Installing Pi OS 5
Download and Install Raspberry Pi Imager – Linux 5
Download and Install Raspberry Pi Imager - Windows 5
Prepare the MicroSD Card 6
Install Raspberry Pi OS 6
Eject the MicroSD Card 6
Boot the Raspberry Pi 7
Power Up for the First Time 7
Installing Cool-Retro-Term 9
Remote Access 9
Installing Hercules MVS-TK5 10
Downloading MVS-TK5 10
Unzip the MVS-TK5 Mainframe 10
Changing Permissions and Running MVS 11
Configuring the Arduino Nano 13
The LED and LCD Code 14
Arduino Code 14
Wiring the Components 16
3D printing 19
Assembly 19
Troubleshooting 20
Arduino Nano and Displays 20

Introduction to the Pi-Frame

Welcome to the **PiFrame** Technical Manual - your comprehensive guide to building, configuring, and enjoying the ultimate retro-modern mainframe experience. The **PiFrame** is a unique blend of modern computing technology, nostalgic aesthetics, and a sprinkling of open-source software. Designed to run a Hercules MVS-TK5 mainframe environment, the **PiFrame** transforms a humble Raspberry Pi into a powerful and captivating vintage-style computing platform.

The core of the **PiFrame** is a Raspberry Pi 4 or 5, equipped with a 32GB SD card running the versatile Raspberry Pi OS. This setup ensures the stability and flexibility required for the Hercules emulator, enabling you to run an authentic IBM mainframe operating system with all its functionality.

To enhance the user experience, **PiFrame** integrates the visually stunning *cool-retro-term* terminal emulator. This terminal faithfully replicates the look and feel of vintage CRT displays, providing an authentic retro interface to complement the mainframe environment.

But **PiFrame** doesn’t stop there. Adding an extra dimension of creativity and interactivity, an Arduino microcontroller powers LED matrices that serve as vibrant status indicators or customizable display panels. Paired with tailored software, these LEDs can be configured to show system activity, data traffic, or even custom animations, adding flair to the overall setup.

Why Build a PiFrame?

PiFrame is more than just a project; it's a journey into the history of computing and an opportunity to explore the possibilities of modern hardware and open-source software. Whether you're a retro computing enthusiast, a hobbyist, or a cybersecurity professional, the **PiFrame** offers a fascinating environment to tinker, learn, and create.

What You'll Find in This Manual

This manual is designed to walk you through every step of the process, including:

- Hardware assembly and wiring for the Raspberry Pi, Arduino, and LED matrices.
- Software installation and configuration, from the base Raspberry Pi OS to the Hercules MVS-TK5 environment.
- Setting up and customizing cool-retro-term for an immersive retro terminal experience.
- Arduino coding and LED matrix programming to add unique visual elements to your **PiFrame**.

By the end of this manual, you'll have a fully functioning mainframe environment housed in a compact, elegant Raspberry Pi setup - ready to tackle retro programming challenges, engage in system simulations, or simply impress your friends. What's more, it will fit neatly at the side of your desktop…unlike the 1970 IBM370 it emulates! We (*Kev Milne and Alisdair Gurney*) hope you enjoy building your very own **PiFrame**! You will need access to a PC running Linux or Windows and an Internet connection to do some of the configuration, but I will assume you have that.

What you need – essential and cosmetic

Essential Parts

- ELECROW Touch Screen Monitor for Raspberry Pi 5, 4, and so on.
- Raspberry Pi 4 Model B (4GB) or Pi 5.
- Micro HDMI Cable
- MicroSD card (32GB is recommended)
- Raspberry Pi Active Cooler – I do recommend this!

Parts for Decoration

Note Although some of these parts are used cosmetically in this project it is perfectly feasible, with a little extra coding, to connect the switches to the Pi or Arduino to provide additional functions.

- DAOKAI MAX7219 Dot Matrix Module 8 x 32 MCU Control Driver Board Green LED
- Dopro 4pcs USA(Imperial) LP Guitar Bell Knobs
- FREENOVE I2C LCD 1602 Module, New Type IIC TWI Serial 16x2 Display (any I2C 1602 module from your favourite bazaar)
- JINXIAN 4 Pcs ON-Off Rocker Switch DPS 4 Pin Red
- Heschen 16mm Square Push Button Switch
- mxuteuk 10Pcs Mini Toggle Switch SPDT 3 Position 3 Pins ON/OFF/ON
- An Arduino Nano (used to control the additional LCD displays)

Getting Started – Installing Pi OS

Let's get started. This section outlines the process of installing Raspberry Pi OS on a Raspberry Pi 5 using the Raspberry Pi Imager.

Note You can use Pi 4, but I will just refer to Pi or Pi 5 from here on.

The guide includes steps for both Linux and Windows operating systems.

Note This section describes a vanilla installation of the Pi OS. You can skip this section if you are familiar with the process or have a suitable Pi handy.

Download and Install Raspberry Pi Imager – Linux

Open a terminal and update your package manager if required on your Linux PC or Laptop. In the terminal at the $ prompt type:

```
$ sudo apt update
```

Install Raspberry Pi Imager using the following command:

```
$ sudo apt install rpi-imager
```

The imaging software will be installed and ready to run.

Download and Install Raspberry Pi Imager - Windows

Visit https://www.raspberrypi.com/software/ and download the Windows installer.

Run the downloaded *.exe* file and follow the on-screen instructions to install Raspberry Pi Imager. Once installed you should see the following screen:

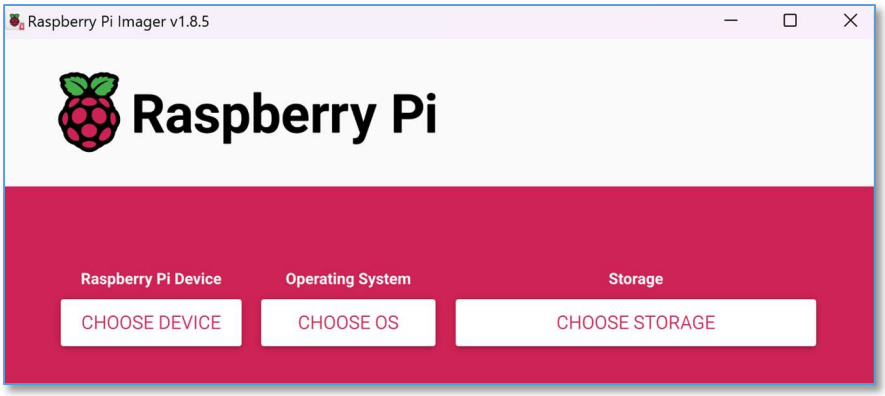


Figure 1- Raspberry Pi Imager

Prepare the MicroSD Card

Insert your MicroSD card into your computer (Laptop or Desktop) using the built-in card reader or an external adapter.

> **Note** Ensure the card is at least 16GB and does not contain critical data, as the process will erase its contents! 32GB is recommended.

Install Raspberry Pi OS

First, we need to launch the Raspberry Pi Imager:

Open the Raspberry Pi Imager from the application menu in Linux, (feel free to run **`rpi-imager`** in a terminal window); Windows users can use the Start menu.

Select the Operating System: Click the **Choose OS** button.
Select *Raspberry Pi OS (64-bit)* or another preferred version of Raspberry Pi OS from the list – Ubuntu will also work well.

Select the Storage Device: Click the **Choose Storage** button.
Select the MicroSD card from the list. If you have plugged it in, and you have made sure it is formatted you should be ready to go.

> **Note** Be careful not to overwrite the wrong device if you have more than one plugged in.

Configure Advanced Options: (Optional) Click the Gear icon (⚙) in the lower-right corner to configure advanced settings, such as:

- Enabling SSH
- Setting up Wi-Fi credentials
- Pre-configuring the hostname
- Setting the default username and password
- Save the settings and return to the main menu.

Write the Image: Click the **Write** button.
When you confirm the action the Raspberry Pi Imager will format the MicroSD card and write the selected Raspberry Pi OS image. Just be patient and wait for completion!

The process may take several minutes. When done, you will see a confirmation message indicating the operation was successful.

Eject the MicroSD Card

Safely eject the MicroSD card using the system's **Safely Remove Hardware** option (Windows) or `umount` command (Linux).

Boot the Raspberry Pi

We are now going to boot the Raspberry Pi to ensure it's up and running. We do have a few other things we will need to connect to it – so this is just to check that it is working as expected.

1. Insert the prepared MicroSD card into the Raspberry Pi's MicroSD card slot.
2. Attach a monitor using the Micro HDMI cable - it's always useful to be able to see what you are doing! If you have purchased the Touch Screen Monitor of your choice, now is a reasonable time to plug it in. Alternatively you could connect remotely using SSH…but that's another conversation.
3. Connect Peripherals such as a USB mouse and keyboard

So, what's next?

Power Up for the First Time

Connect the power supply to the Raspberry Pi. The device will automatically power on and begin the boot process.

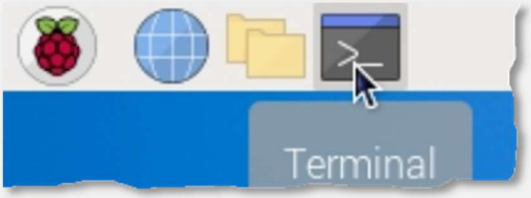
Initial Pi-OS Setup

Follow the on-screen instructions to complete the setup, including:

- Choosing a language and keyboard layout
- Connecting to Wi-Fi or Ethernet (*if you didn't do that as part of the Pi OS configuration*).
- Optionally, update the operating system to the latest version with the following command:

```
$ sudo apt update & sudo apt upgrade
```

Note To run the above command you will need to open up a terminal window:



When done you should have something that looks like the device shown in the Figures below – I added the IBM wallpaper:

Figure 2: Your booted up but naked Pi – with IBM wallpaper!

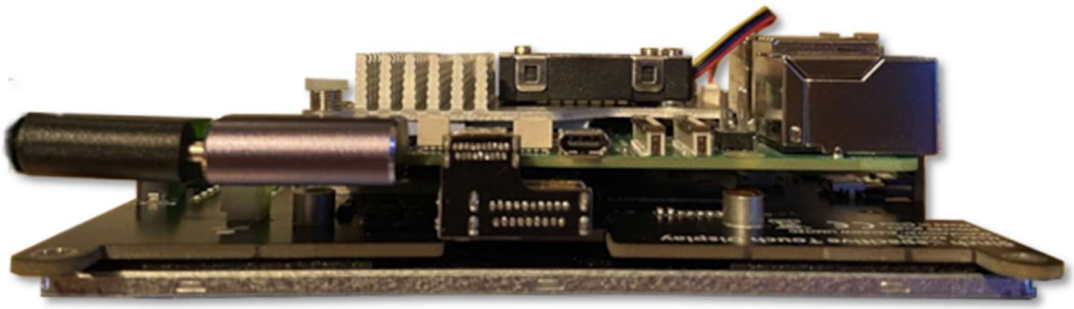


Figure 3: View from the top. Showing Power and HDMI Connections.

Installing Cool-Retro-Term

Cool-retro-term is a lightweight, aesthetically pleasing terminal emulator designed to mimic the look and feel of classic CRT monitors. With its customizable vintage-style themes, smooth animations, and authentic visual effects, I fell in love with it instantly. It adds that nostalgic touch to the command-line and console experience. To install from a terminal, enter the following:

```
$ sudo apt install cool-retro-term
```

That's it. Simple as that. To run it type:

```
$ cool-retro-term &
```

I always like to append **&** as this means it will keep running in the background without tying up the terminal. Considering this will form the MVS3.8J console screen and be presented on the small touch screen it lets you switch to another terminal on which you can run your `x3270` or `c3270` client. Now we have the touch screen, power, keyboard and all switched on, and Ethernet configured (whether by WiFi or, like me, cable) you should be ready to go. You can also plug in another Micro HDMI so you can view this on multi-screen and work from the larger monitor. Make sure you do the next part from the **PiFrame**.

Remote Access

It is convenient to be able to access the Pi in the **PiFrame** remotely from your usual PC or laptop, whether command line using PUTTY over SSH or full graphical view of the desktop.

For remote access you have the option of turning on Raspberry Pi Connect – for further information on setting that up head over to https://www.raspberrypi.com/software/connect/ where you will also set up a link (I think it is great to have remote access to your device from anywhere). Full docs are here - https://www.raspberrypi.com/documentation/services/connect.html

Alternatively you can simply set up access over Secure Shell (SSH) as follows:

```
$ sudo apt install ssh          // installs sshd
$ sudo service ssh start        // starts the service
$ sudo sysctl enable ssh        // enables the service on boot.
 --snip--
```

Note You do not need the above if you enabled SSH during the Pi OS configuration.
The forward slashes (`//`) are used for comments so you do not need to type those.

Then, from your desktop

```
$ ssh name@<IP address>         // First time hit 'Y' to agree.
```

Installing Hercules MVS-TK5

The Hercules MVS-TK5 Update 3, curated and enhanced by Rob Prins, is an exceptional mainframe emulation package. This update builds upon the robust Hercules emulator, which enables users to run IBM mainframe operating systems on low-cost hardware, by bundling a fully configured Turnkey MVS environment. It includes essentials like TSO, ISPF, NJE, and FTP Server and Web Server. It has a few preconfigured users, significantly reducing the setup time and complexity for users eager to explore the world of MVS (Multiple Virtual Storage). It has `RAKF` and `KICKS` as well as a thriving community writing utilities for it. It also works with `JCL`, `BREXX` and `CLIST`, amongst other languages - so you can customize it. With its easy installation, updated features, and comprehensive documentation, the TK5 Update 3 offers real accessibility to a historically significant computing environment – all wrapped up in 16mb of storage. I learned on this baby – great for when you can't get hold of a mainframe.

Downloading MVS-TK5

Assuming that this is your first time downloading the package just head to https://www.prince-webdesign.nl/index.php/software/mvs-3-8j-turnkey-5 and grab the download. Choose the download on the left to get the whole package.

Figure 4- Click on - Download MVS-TK5 with Update 3

Once the zip file has downloaded, make sure you are in your **`/home`** folder. You can do this by typing:

```
$ cd ~
$ pwd
/home/kev
```

Note Check where the file is downloaded to, it will very likely be found in ~/Downloads. Feel free to copy it to your home directory.

Unzip the MVS-TK5 Mainframe

We are going to extract it into your **`/home`** folder as follows:

```
$ unzip mvs-tk5.zip
```

Next, change into the newly created mvs-tk5 directory.

```
$ cd mvs-tk5
```

Changing Permissions and Running MVS

We are now going to change the settings to ensure that the files are executable. As we are also going to set up the console in unattended mode, we will also make the files in the unattended directory executable also. Add the commands below (any text following `//` is a comment, so don't enter that).

```
$ cd unattended           // change to the unattended directory.
$ chmod -R 755 *          // change permissions to execute.
$ ./set_console_mode      // run console mode.
```

Your Hercules console mode is now activated for unattended operation.

Hit Enter again

```
$ cd ..                   // move up to the mvs-tk5 directory
$ chmod -R 755 *          // changing permissions in the mvs-tk5 directory
```

If your `cool-retro-term` isn't already started then start it back up with the command:

```
$ cool-retro-term
```

Make sure you are in the `mvs-tk5` directory and run:

```
$ ./mvs                   //  this will start MVS
```

You should see the following screen 😊

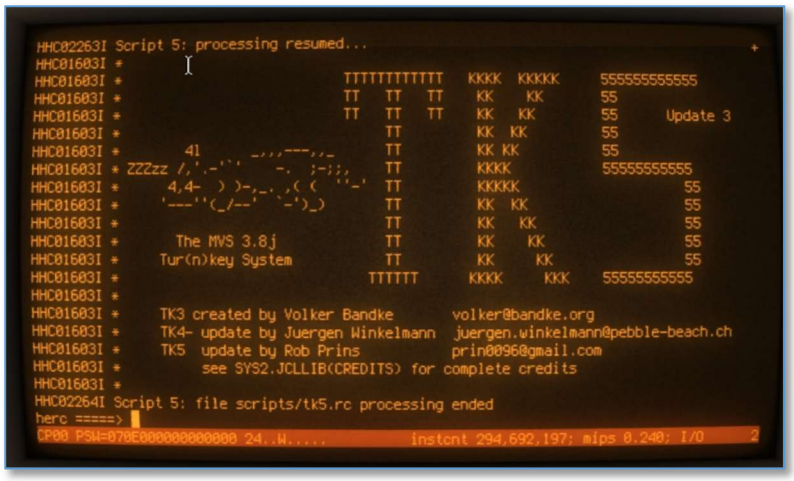


Figure 5: MVS-TK5 all set up.

You can switch back and forth between screens with the ESC key.

Figure 6: Pressing ESC at the console.

Feel free at this point to enter some commands such as `/d t` or `/d a,l` or even start up the FTP server `/s ftpd`. You can right click on the screen and play with the profiles to get the look you want. Pretty much at this point, if you want you can add a case and call it a day. However, if you want to go further into the **PiFrame** and add some bells and whistles read on!

Configuring the Arduino Nano

The Arduino Nano is a compact and powerful microcontroller, ideal for integrating into projects like the **PiFrame**. We'll use it to manage the LED matrix and LCD matrix, key components of the PiFrame's functionality. I'll walk you through configuring the board and uploading the necessary code, ensuring your **PiFrame** setup is seamless and ready to go.

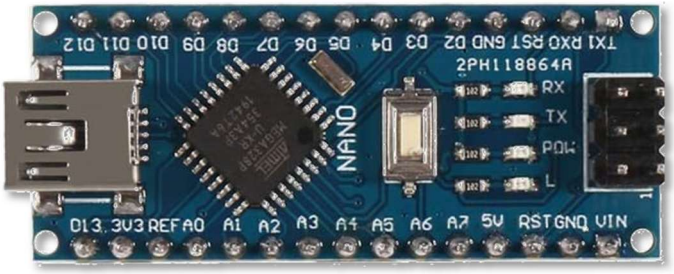


Figure 7 Arduino Nano - LED and LCD controller.

First things first. You'll need to install the Arduino IDE on your desktop PC or laptop. On Windows you can find the Arduino IDE in the Microsoft App Store. On Linux you can simply type:

```
$ sudo apt install arduino
```

Start the Arduino IDE by finding it on the Start Menu in Windows; on Linux, simply type:

```
$ arduino
```

Connect your Nano to a USB port and make sure your system detects the device. I found the following site excellent, so I am pointing you there for the next steps as there is no point in re-inventing the wheel! Head on over to https://www.instructables.com/Arduino-Nano-1/ for further information about using the Arduino IDE to program your Nano. This is not the end of this section, as once we have the Nano ready to go we will still need to complete the coding AND wire the parts together.

The LED and LCD Code

Download the code and upload to the Arduino IDE. You will also have to add the following libraries:

```
MD_MAX7anc2XX
LiquidCrystal_I2C
millisDelay
MD_Parola
```

The full code is available on my GitHub at - https://github.com/kmilne40/PiFrame/blob/main/piframe.ino as a .ino file.

I've also added the code below, whilst also highlighting for simplicity where you can change some of the output if you want the LCD matrix to say something other than Pi-Frame: ACTIVE.

Arduino Code

```
#include <MD_MAX72xx.h>         // Library for controlling LED matrices
#include <MD_Parola.h>          // Advanced library for LED matrix animations
#include <SPI.h>                // SPI communication library
#include <LiquidCrystal_I2C.h>  // Library for I2C-controlled LCD displays
#include <millisDelay.h>        // Non-blocking delay library

// Code by Alisdair Gurney and Kev Milne - Tightened up and checked by CGPT.

// LCD Configuration
#define LCD_I2C_ADDRESS 0x27    // I2C address for the LCD
#define LCD_WIDTH 16            // LCD width in characters
#define LCD_HEIGHT 2            // LCD height in rows
LiquidCrystal_I2C lcd(LCD_I2C_ADDRESS, LCD_WIDTH, LCD_HEIGHT);  // LCD instance

// LED Matrix Configuration
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW  // Hardware type for LED matrix
#define MAX_DEVICES 4                      // Number of connected LED matrix modules
#define CLK_PIN 13                         // Clock pin for LED matrix
#define DATA_PIN 11                        // Data pin for LED matrix
#define CS_PIN 10                          // Chip Select pin for LED matrix
MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES); // LED matrix instance

// Timer for LCD updates
millisDelay lcdUpdateTimer;                // Timer for periodic LCD updates
const unsigned long LCD_UPDATE_INTERVAL = 4000;  // Update interval in milliseconds

// Variables
int lcdRow = 0; // Tracks the current row for alternating messages on the LCD

// Function Prototypes
void setupLCD();             // Initialize the LCD
void updateLCD();            // Update the LCD display content
void setupLEDMatrix();       // Initialize the LED matrix
void randomLEDFlashing();    // Create a random LED flashing effect

void setup() {
  Serial.begin(115200);      // Initialize serial communication for debugging
  setupLCD();                // Set up the LCD
  setupLEDMatrix();          // Set up the LED matrix
  lcdUpdateTimer.start(LCD_UPDATE_INTERVAL);  // Start the LCD update timer
}

void loop() {
  // Check if it's time to update the LCD
  if (lcdUpdateTimer.justFinished()) {
    updateLCD();             // Update the LCD display
    lcdUpdateTimer.restart();// Restart the timer
  }
```

```
  // Continuously run the random LED flashing effect
  randomLEDFlashing();
}

void setupLCD() {
  lcd.init();                // Initialize the LCD
  lcd.begin(LCD_WIDTH, LCD_HEIGHT); // Set LCD dimensions
  lcd.backlight();           // Turn on the LCD backlight
  lcd.clear();               // Clear any existing content on the LCD
  lcd.setCursor(0, 0);       // Set cursor to the first row
  lcd.print("Pi-Frame: ACTIVE "); // Display initial message
  lcd.setCursor(0, 1);       // Move cursor to the second row
  lcd.print("MVS38J-81-IPL150"); // Display additional information
}

void updateLCD() {
  lcd.clear();               // Clear the LCD before updating
  lcd.setCursor(0, lcdRow);  // Set cursor to the current row
  lcd.print("Pi-Frame: ACTIVE  "); // Display active message
  lcd.setCursor(0, 1 - lcdRow);    // Alternate to the other row
  lcd.print("MVS38J-81-IPL150");  // Display secondary message
  lcdRow = 1 - lcdRow;       // Toggle between rows for the next update
}

void setupLEDMatrix() {
  mx.begin();                // Initialize the LED matrix
  mx.control(MD_MAX72XX::INTENSITY, 5); // Set matrix brightness to medium
  mx.clear();                // Clear any previous data on the matrix
}

void randomLEDFlashing() {
  static unsigned long lastUpdate = 0; // Track the last update time
  const unsigned long updateInterval = 350; // Time between updates in milliseconds

  // Check if enough time has passed for the next update
  if (millis() - lastUpdate >= updateInterval) {
    lastUpdate = millis();   // Update the last update time
    mx.clear();              // Clear the LED matrix

    int ledCount = 64;       // Number of LEDs to light up randomly
    uint8_t brightness = random(1, 3); // Random brightness level (1 to 3)
    mx.control(MD_MAX72XX::INTENSITY, brightness); // Apply the brightness

    // Randomly light up LEDs on the matrix
    for (int i = 0; i < ledCount; i++) {
      uint8_t x = random(0, 8); // Random row (0-7)
      uint8_t y = random(0, mx.getColumnCount()); // Random column
      mx.setPoint(x, y, true); // Turn on the LED at (x, y)
    }

    mx.update();             // Update the LED matrix with new data
  }
}
```

Wiring the Components.

Overall, the front panel was initially laid out on paper as in the diagram below. You can clearly see on the bottom right the where the wires are positioned. It will become clearer on the figure of the inside of the **PiFrame** *(Figure 13):*

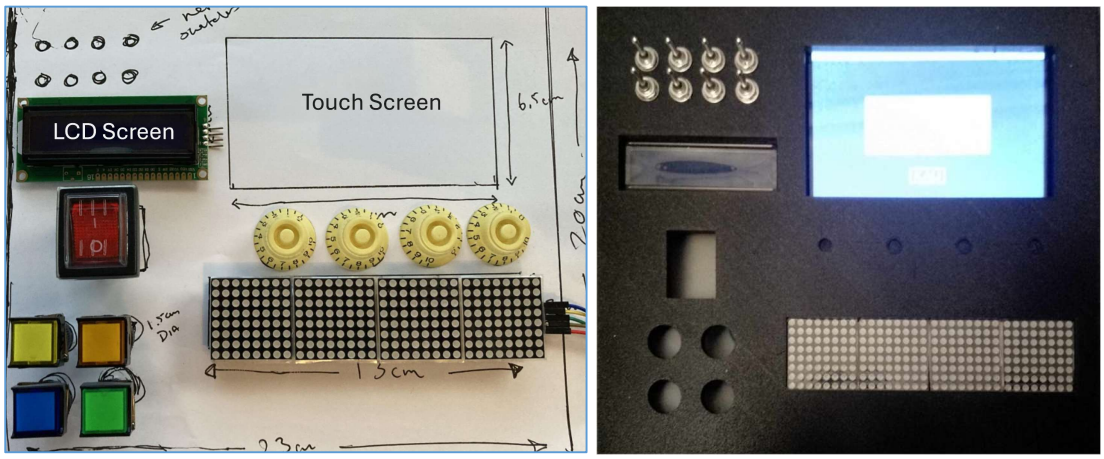


Figure 8 PiFrame front panel design.

Most of the switches and knobs are just cosmetic, but could easily be used to perform additional functions, whether on the Raspberry Pi or the Arduio Nano. Feel free to add to the project!

The internal structure of the **PiFrame**, including the wiring, is shown in Figures 9 to 12. Due to the significant power demands of the Pi 5, a touchscreen, an Arduino, and the LED/LCD components, an additional power supply has been incorporated to support the Arduino and the LED/LCDs. To ensure proper operation and connectivity between devices via the Arduino, it is necessary to split the cables, allowing shared access to the same power pins.

Now, let's look at the wiring for the 64x4 LED.

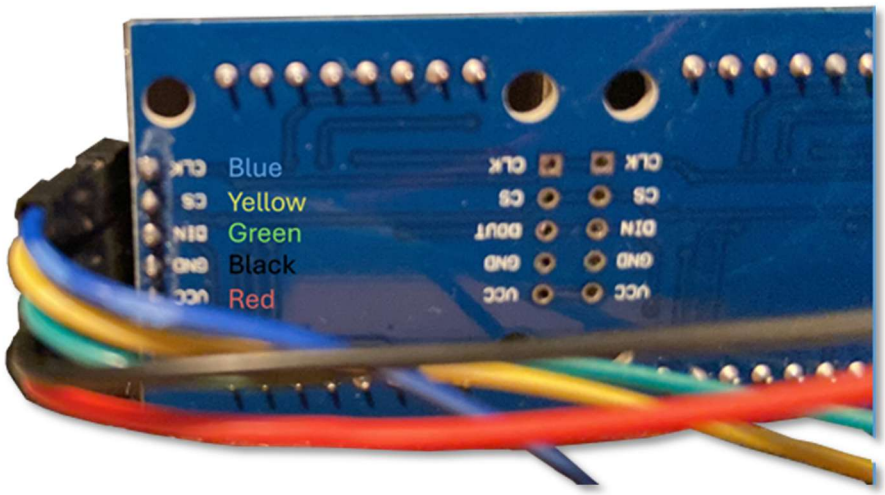


Figure 9 - LED Wiring

Next, from there, we move to the Arduino connection. Note that the Micro-USB connector is on the right-hand side. Insert the pins accordingly as shown in Figure 10.

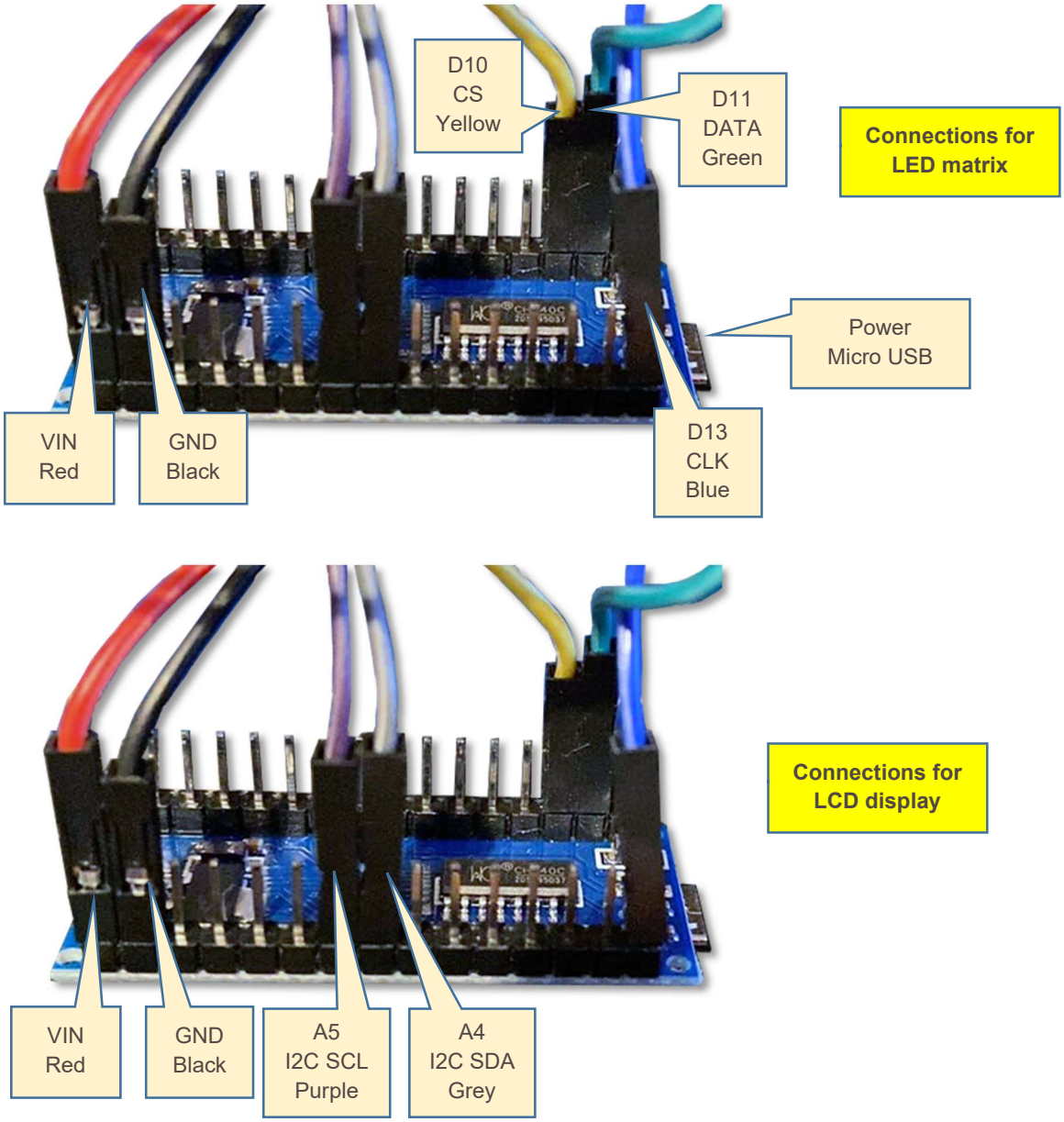


Figure 10 - Arduino Connections

The red and black cables on the Arduino connect to both the LCD and LED devices. Figure 11 shows a close-up of the LCD device:

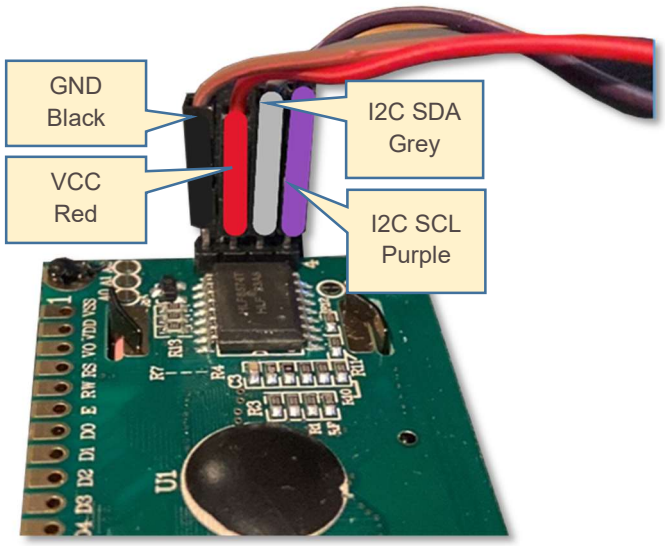


Figure 11 - LCD connections.

Note Be aware that pinouts can vary between suppliers and manufacturers, so please confirm with the labels on the board.

Finally, we can look at the full picture so we can see how the wiring should be connected to power these items. See Figure 12.

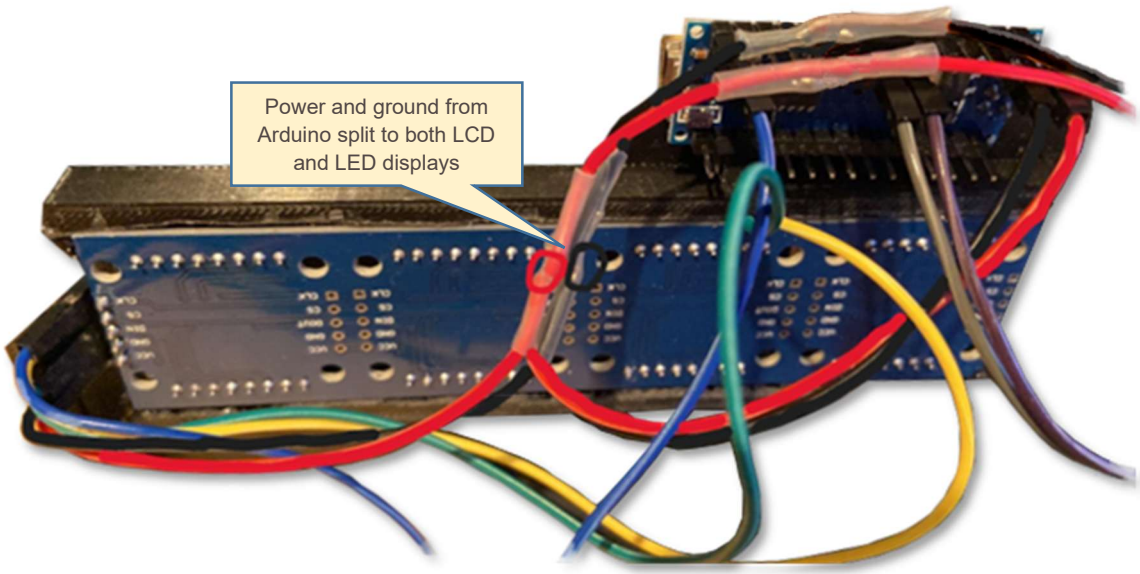
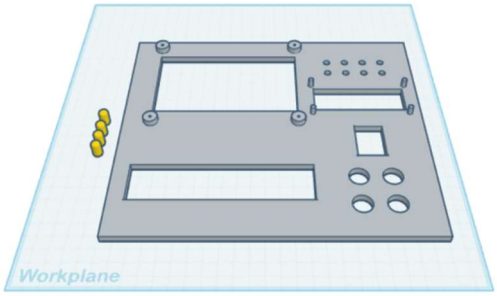


Figure 12 - Connections for Power

3D printing

A 3D design front panel for the **PiFrame** was created by Alisdair Gurney. The STL file can be downloaded from the github page - https://github.com/kmilne40/PiFrame/blob/main/PiFrame-face.stl, or the design can be copied and edited in Tinkercad (https://tinyurl.com/8cc29mw5)



Note that this will only give you the front panel. Once you have the panel printed out you decide how to create the container, or should I say, the box for the front panel.

Assembly

Although the old IBM systems had side panel options of Yellow, Red and Black I decided to obtain a clear acrylic display case - *1 Pack Acrylic Display Case Box Transparent Dustproof Showcase, 7.9x7.9x3.9 Inch for Collectibles.*

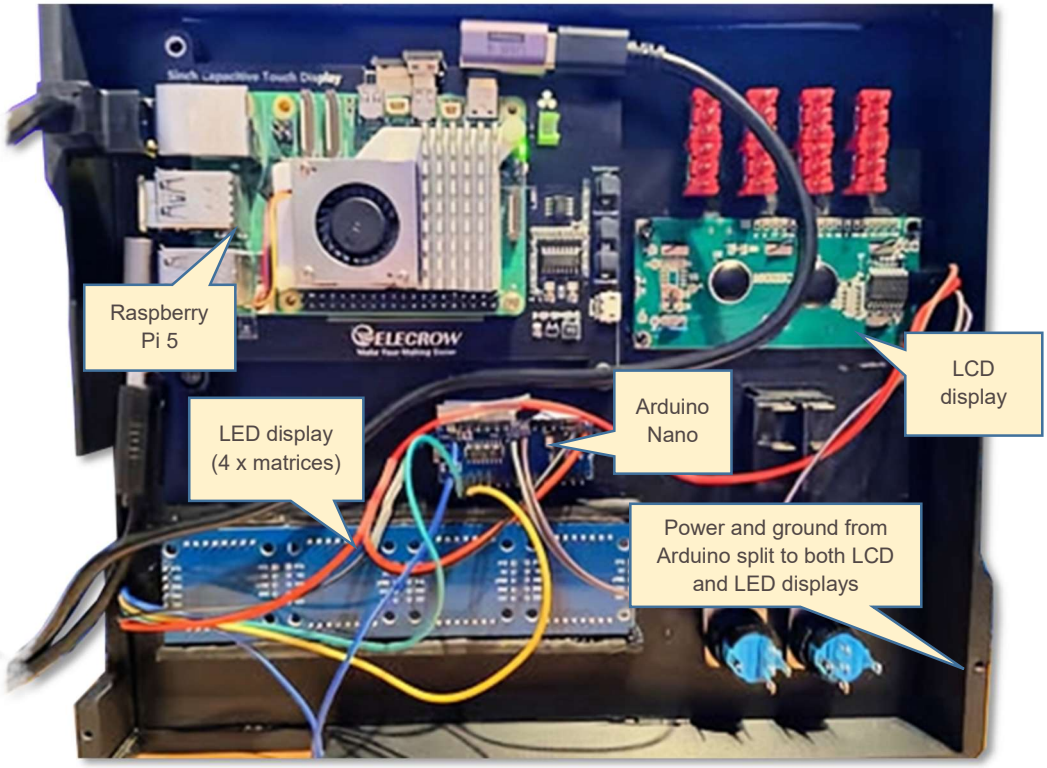


Figure 13 - The completed internals.

Note I recommend getting the USB-C right angled adapters as well as the Ethernet adapter. This is to ensure that everything fits nicely.

Troubleshooting

Arduino Nano and Displays

Don't panic if everything compiles and uploads correctly but nothing seems to work as you expect.

- The I2C addresses for the LED matrices can vary depending on the manufacturer and supplier. 0x27 is usual but 0x3F may also be encountered.
- If you want to write text to the LED matrices, rather than just random flashing, be aware that some displays write to LEDs row by row (for example left to right), some are column by column (for example top to bottom) while others use a zigzag either horizontally or vertically .. you never quite know what you're going to receive, but the Arduino web site (https://www.arduino.cc/) is a good place to find solutions.
- You can add some debug lines to see what's going on; use something like:

```
Serial.print("some debug text : ");
Serial.println(<variable name>);
```

  You can monitor the debug output in the Arduino IDE's serial console.
- Multi-line LCD displays usually have a small screw on the back for changing the contrast, so if you can't see text (or it's too bright) that's a good place to start.
- You can also add the ino code to your friendly AI of choice (I use ChatGPT) get it to analyze it and then tell it how you would like it changed (if you are short on time).