Raspberry Pi User Manual

A Guide by Aman Balam

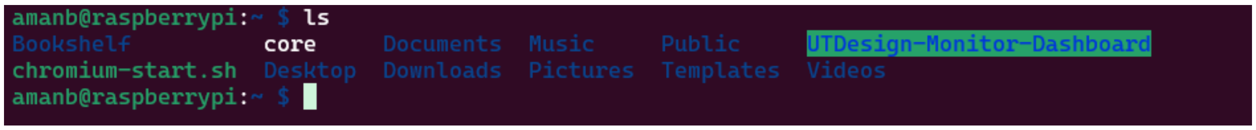
This is a guide and will help you get up to speed on what has been completed involving the raspberry pi

Honestly not much needs to be edited this is just for reference

If there are any issues or if you want to modify the script take a look at the startup script located in the home directory

You want to edit the chromium-start.sh program

nano chromium-start.sh



```
amanb@raspberrypi:~ $ ls
Bookshelf          core     Documents  Music     Public     UTDesign-Monitor-Dashboard
chromium-start.sh  Desktop  Downloads  Pictures  Templates  Videos
amanb@raspberrypi:~ $ 
```

Overview: How the Raspberry Pi Boots and Loads the Application

When the Raspberry Pi starts, it automatically runs a series of setup scripts that ensure your **UTD Monitor Dashboard** is displayed and running correctly on **http://localhost:3000**.

1. **Boot Process**:
   - When the Raspberry Pi is powered on or restarted, it first loads the operating system (Raspberry Pi OS).
   - Upon booting, a **startup script** is executed automatically to configure the environment and launch the necessary services.
2. **Startup Script**:
   - The script is typically set up using **pm2**, a process manager that ensures the application starts on boot and keeps running. This is crucial to ensure that even if the Raspberry Pi is rebooted, the application will restart without needing manual intervention.
   - The script also makes sure that **Chromium** is launched in kiosk mode to open the **UTD Monitor Dashboard** at **http://localhost:3000**. This means that as soon

as the Pi boots, the browser window is opened, and your application is displayed automatically.

3. **Autostart with pm2**:

    - **pm2** is used to manage the backend application that runs on the Raspberry Pi. It keeps the application running in the background and ensures it automatically starts when the Pi boots up.

    - The application, which is a **Node.js app**, is started by pm2 and listens on **http://localhost:3000**.

4. **Launching Chromium**:

    - Once the application is running, a separate script (usually chromium-start.sh) launches the **Chromium** browser in full-screen mode, which displays the dashboard on the Pi's connected monitor. Chromium is configured to open the application at **http://localhost:3000**, where it serves the **UTD Monitor Dashboard**.

Note that the majority of this is done already (prisma configuration; rsync has the project loaded from 4-27-25)

1. Note on .env File Setup

Before starting, make sure that your **.env** file is set up correctly, as this configuration is essential for your application to function as expected.

2. Purpose of the Raspberry Pi

The main function of the Raspberry Pi is to display the **UTD Monitor Dashboard** on **http://localhost:3000**.

To achieve this, several startup scripts have been implemented to automatically start the Pi on boot and display the application.

3. Accessing the Application

Once the Raspberry Pi has booted and started the application, you can log in and interact with it via your browser.

4. Updating the Project on the Raspberry Pi

To update the project on the Raspberry Pi, follow these steps:

1. First, sync the latest commits in your VS Code repository to ensure you have the latest code.

Use the rsync command to transfer the updated files to the Pi.

Important: If you are using Windows, you will need to have the **Windows Subsystem for Linux (WSL)** installed. After SSHing into the Pi (which is already configured to accept SSH connections), run the following rsync command to sync your files.

The rsync command might look like this

rsync -avz /mnt/c/Users/YourUsername/Desktop/EPICS/UTDesign-Monitor-Dashboard/ user@<raspberry-pi-ip>:/home/user/

2. **Explanation**:

    - This command uses WSL to locate the project files and sends them to the Pi over SSH using the Pi's IP address.
    - Replace user and the file paths as appropriate for your setup.

5. Configuring Prisma for ARM on Raspberry Pi

Since the Raspberry Pi runs on an ARM architecture, Prisma needs to be configured to use the correct binary for ARM.

Step 1: Set the PRISMA_QUERY_ENGINE_BINARY in the .env File

In your project's **.env** file, add the following line to point to the ARM-specific Prisma query engine binary:

PRISMA_QUERY_ENGINE_BINARY=https://github.com/prisma/prisma-engines/releases/download/2.29.0/prisma-query-engine-linux-arm64

This line instructs Prisma to use the appropriate query engine binary for ARM architecture (linux-arm64).

Step 2: Configure binaryTargets in schema.prisma

In the **schema.prisma** file, ensure that the binaryTargets are set to include ARM architecture. Add the following under the generator client block:

```
prisma
CopyEdit
generator client {
  provider = "prisma-client-js"
  binaryTargets = ["native", "linux-arm64"]
}
```

Step 3: Regenerate Prisma Client

Once the configurations are set, regenerate the Prisma client by running the following command:

```
npx prisma generate
```

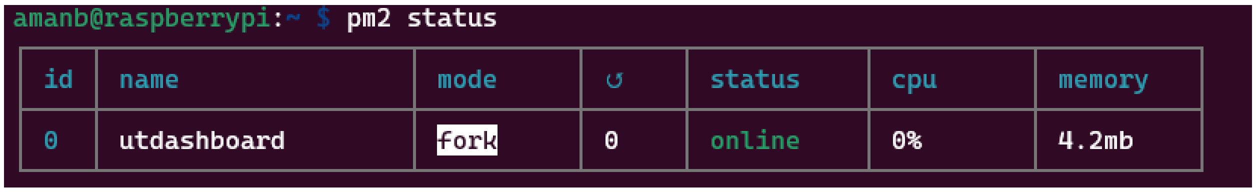
This ensures Prisma generates the client using the ARM-compatible binary.

6. Testing the Application

After updating the project and making sure everything is configured correctly, you can test if the application is running by using the following command:

```
pm2 status
```

Should output similar to:



```
amanb@raspberrypi:~ $ pm2 status
┌────┬──────────────┬──────┬───┬────────┬─────┬────────┐
│ id │ name         │ mode │ ↺ │ status │ cpu │ memory │
├────┼──────────────┼──────┼───┼────────┼─────┼────────┤
│ 0  │ utdashboard  │ fork │ 0 │ online │ 0%  │ 4.2mb  │
└────┴──────────────┴──────┴───┴────────┴─────┴────────┘
```

If everything is set up correctly, the output should indicate that the application is running.

Additionally, Chromium will open automatically, displaying the application at **http://localhost:3000**.

Additional Considerations

- **Autostart**: The Raspberry Pi is set up to start the application automatically on boot. The necessary startup scripts and services (like pm2) ensure that the application is running after reboot.

- **Updates**: Every time you update your project, you should sync it with the Pi using the rsync command as shown above.

- **Prisma**: If you’re using Prisma with a database, remember that the query engine must be compatible with the ARM architecture. The steps provided ensure Prisma works correctly on the Raspberry Pi.