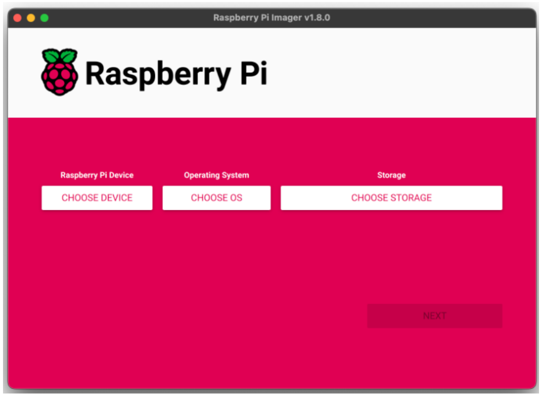
D

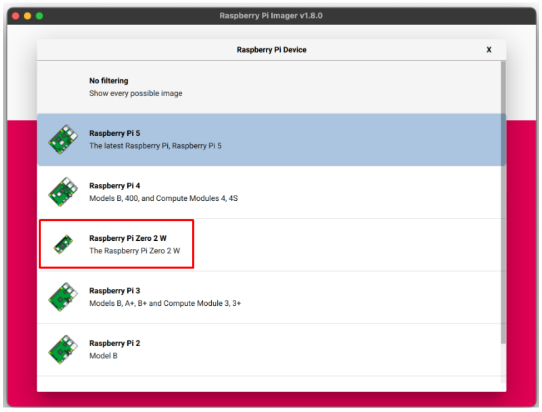
System Setup Manual

D.1. Operating System

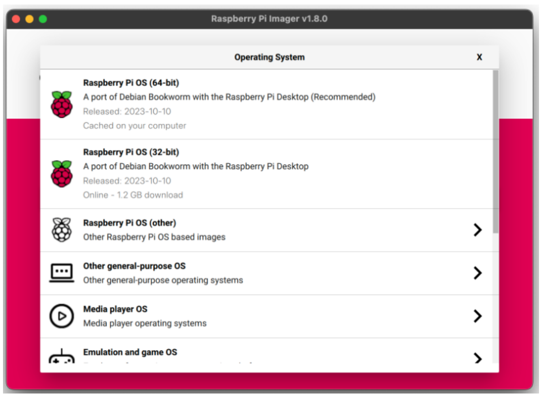
The Raspberry Pi OS is flashed onto the microSD card using the official *Raspberry Pi Imager*, available via raspberrypi.com. After launching the Imager, select the Raspberry Pi Zero 2 W (Figure D.1b), followed by the 64-bit Debian Bookworm with desktop OS image (Figure D.1c). Insert the microSD card using a card reader (Figure D.1d). Before writing, optional settings such as username, Wi-Fi, hostname, locale, keyboard layout, and SSH/VNC access can be preconfigured. Once applied, the OS is written to the card, producing a ready-to-use environment.



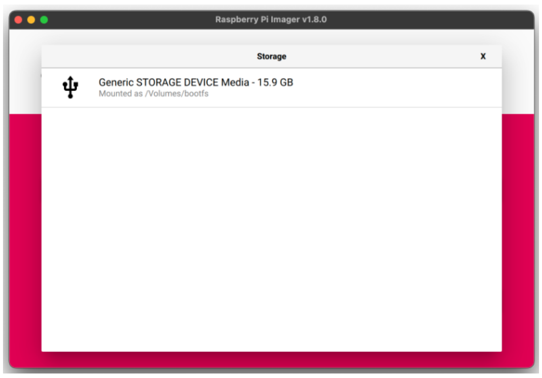
(a) Select target device.



(b) Select processor module.



(c) Select operating system.



(d) Select target storage.

Figure D.1: Overview of the SD-card imaging process.

D.2. First-Boot Configuration: Swapfile and Updates

To allow initial GUI-based configuration, the Pi Zero 2 W can be connected to an external monitor (HDMI → mini-HDMI) and controlled via USB hub (micro-USB → USB-A) with keyboard and mouse. After booting into the desktop, the default 100 MB swapfile is increased to 1024 MB to improve stability under image-processing load; care should be taken to avoid excessive SD-card wear (by increasing the swapfile size too much) (source). This can be achieved using the terminal as follows:

1. Disable swap temporarily:

   ```
   sudo dphys-swapfile swapoff
   ```

2. Edit the swap configuration file:

   ```
   sudo nano /etc/dphys-swapfile
   ```

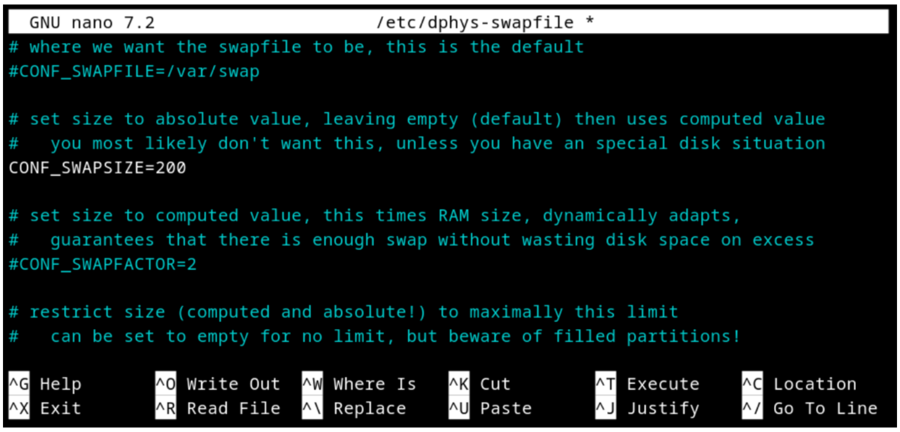


Figure D.2: Swapfile configuration.

3. Increase swap size: Change the line `CONF_SWAPSIZE=100` to `CONF_SWAPSIZE=1024`. Save changes with `CTRL + O`, `CTRL + X`, and confirm with `Y + ENTER`.
4. Apply the new swap size:

   ```
   sudo dphys-swapfile setup
   ```

5. Re-enable swap:

   ```
   sudo dphys-swapfile swapon
   ```

6. Reboot the system:

   ```
   sudo reboot
   ```

After reboot, the system is stable for all GUI and processing tasks. Regular package updates are then performed:

```
sudo apt update
sudo apt upgrade
```

D.3. Package Installation

The water-level detection software depends on both system-level camera drivers and a set of Python libraries. Before operation, ensure the following packages are installed on Raspberry Pi OS:

System & camera features

- libcamera-apps
- libcamera-dev
- v4l-utils

Core Python libraries

- python3-picamera2 (Picamera2 API & libcamera bindings)
- python3-pillow (PIL image handling)
- python3-numpy (array operations)
- python3-matplotlib (plotting backend)
- python3-scipy (signal processing & statistics)
- python3-requests (HTTP for OCR API)
- python3-psutil (system metrics: CPU, memory)
- python3-rpi.gpio (GPIO control)

This can be achieved by running the following line in the terminal command prompt:

```
sudo apt update
sudo apt install -y libcamera-apps libcamera-dev v4l-utils \
python3-picamera2 python3-pillow python3-numpy \
python3-matplotlib python3-scipy python3-requests \
python3-psutil python3-rpi.gpio
```

D.4. Auto-Start Service

To enable the water-level detection algorithm to launch automatically after any reboot, create a `systemd` service unit. This ensures the system is self-recovering: if it loses power or crashes, it will come back online and resume measurements without manual intervention. In combination with a power-management board, energy use can be minimized by only powering the Pi when a measurement cycle is due (see Section 8).

Create and edit the service file with:

```
sudo nano /etc/systemd/system/<filename>.service
```

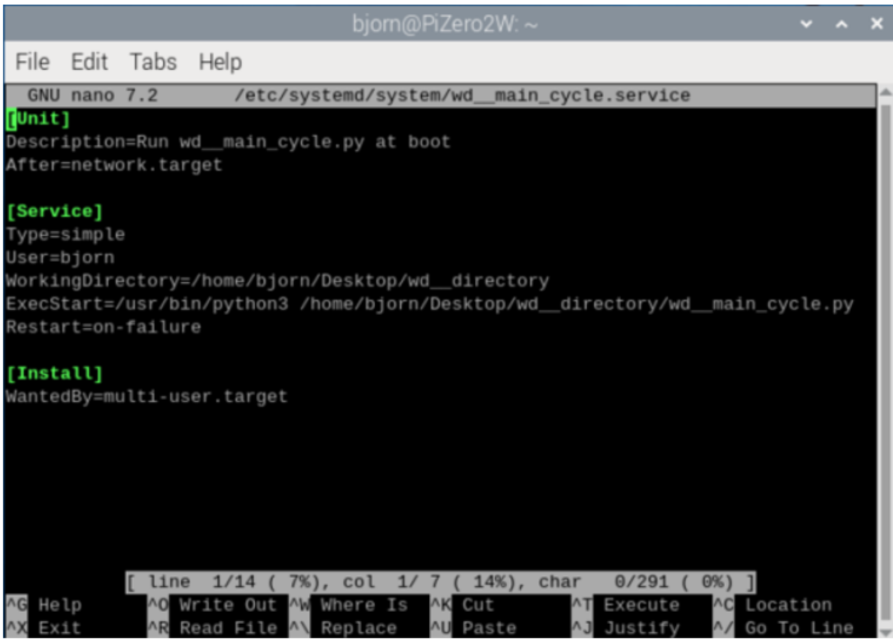


Figure D.3: Example of .service file

This file will be available on <GitHub link>.

The service file begins with a [Unit] section that names the task and tells systemd to wait until the network is online before starting. In the [Service] section, it specifies that systemd should run the Python interpreter on your wd__main_cycle.py script as the pi user, from the project's working directory, and automatically restart it if it ever crashes. Finally, the [Install] section hooks the service into the normal multi-user boot sequence so that enabling it causes the script to launch on every reboot without any further intervention.

After placing the service unit file into /etc/systemd/system/, the systemd manager must reload its configuration so that the new unit becomes available. The service can then be configured to start automatically on boot or prevented from doing so, and it supports manual start and stop operations as well as status inspection. This ensures that the water-level detection algorithm recovers automatically after power interruptions and can be managed interactively for troubleshooting or maintenance.

The relevant commands are:

```
sudo systemctl daemon-reload
sudo systemctl enable <filename>.service
sudo systemctl disable <filename>.service
sudo systemctl start <filename>.service
sudo systemctl stop <filename>.service
sudo systemctl status <filename>.service
```