Building the S.H.U.G.R. PI

The Official Guide

Introduction

The Stormwrecker Handheld Undersized Gaming Raspberry Pi (or SHUGRPI) is a Pygame-Powered handheld entertainment system specifically designed to run Pygame apps. These instructions will tell you how to build one using a **Raspberry Pi 5.** In this project, you will

1. Set up the Raspberry Pi with basic configurations
2. Build the software components
3. Assemble the hardware components

Let's start!

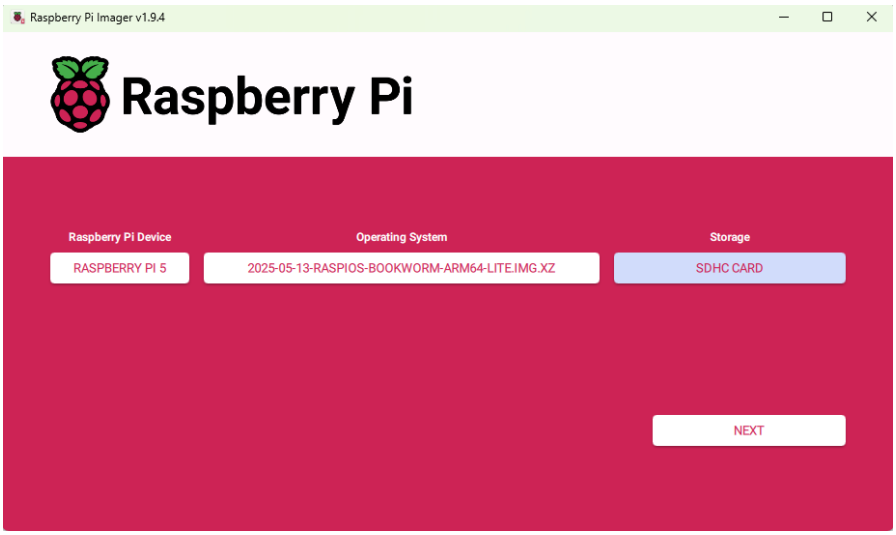
Section 1: Software

Step 0: Booting the Pi

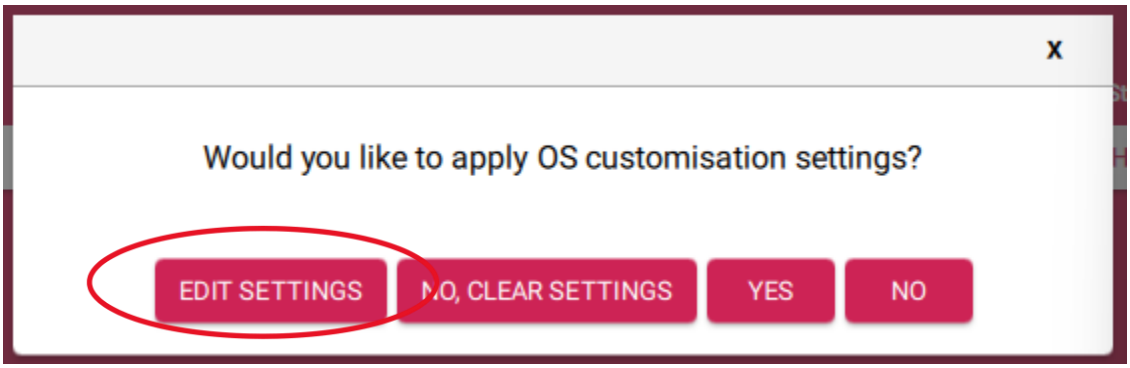
Open up Raspberry Pi Imager (I'm using v1.9.4), and for device, choose Raspberry Pi 5. For OS, choose this one:

https://downloads.raspberrypi.com/raspios_oldstable_lite_arm64/images/raspios_oldstable_lite_arm64-2025-11-24/2025-11-24-raspios-bookworm-arm64-lite.img.xz

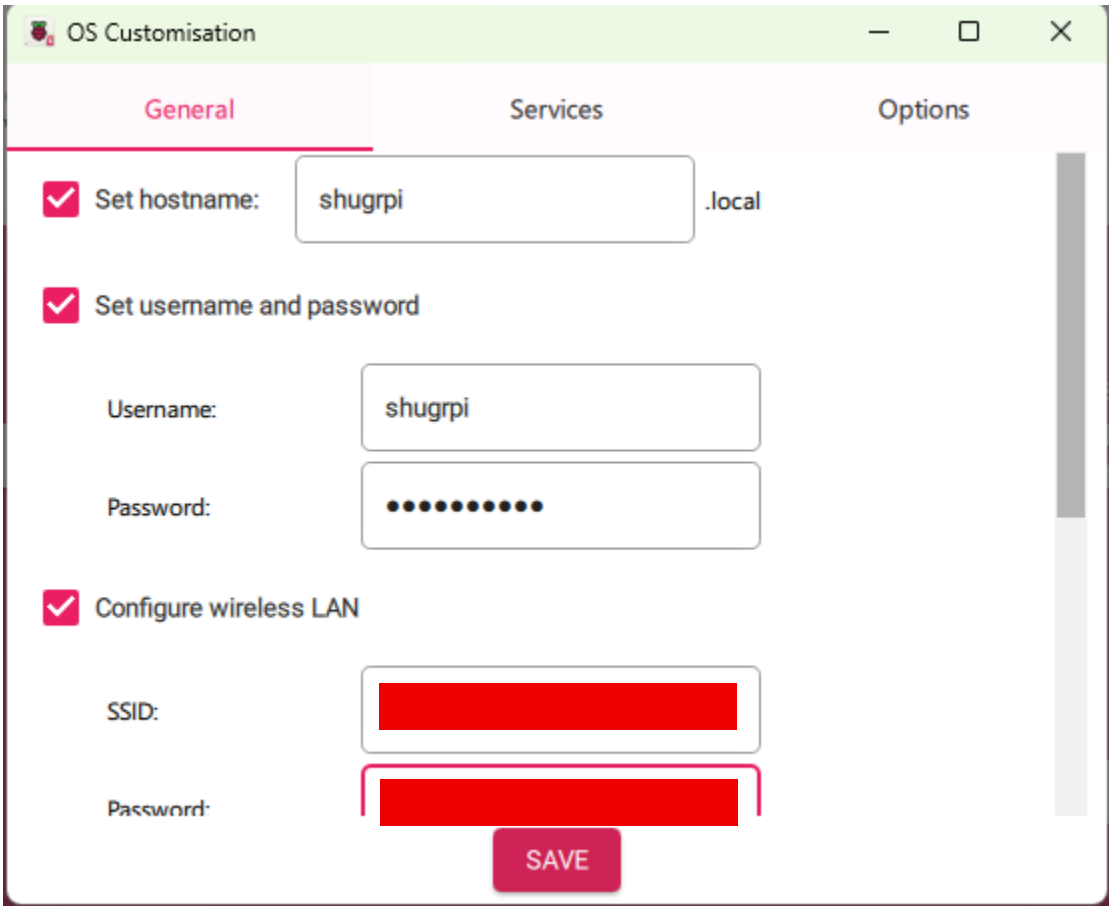
This is Raspberry Pi OS Lite (64-bit) Debian-12 Bookworm. You do **not** want to use Debian-13 Trixie. Next, select your SD card for storage. You should see something like this:



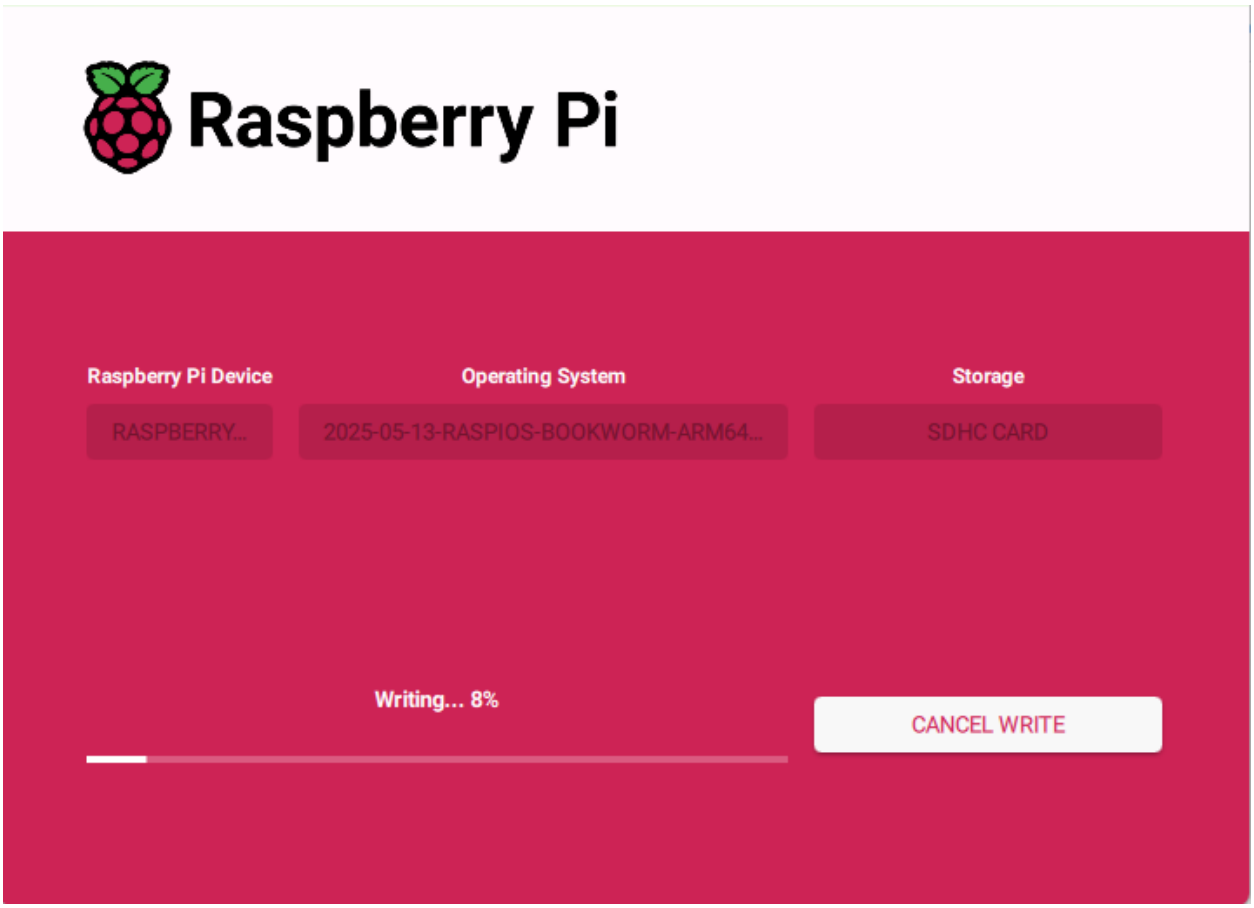
Before flashing your SD card, make sure to edit OS customization settings.



Set hostname to “shugrpi”, then set username to "shugrpi” and set password to whatever you want. Enable wireless LAN and enter in SSID with its password.



In “Services”, enable SSH and use password authentication. In Options, leave “Enable telemetry” unchecked. Now, you are ready to flash the SD card!



Now it is time to put the SD card in the Pi and boot it up for the first time!

Step 1: Connecting to the Pi

Now your Pi is up and running and there is no way to talk to it from your PC. Additionally, there is no way for the Pi to access the internet. This means you cannot install or download any packages for this project. Let's rectify this!

Your first command you'll type in is this:

```
sudo raspi-config
```

This brings up an interface for general configurations for your Pi. Go to "System Options" -> "Wireless LAN" and enter in your SSID and password. If you get an error or warning, be sure to go to "Localization Options" -> "WLAN Country" and select the country you are in. If these steps have worked, you have connected to the internet.

Make sure after these changes that you run:

```
sudo reboot
```

One last thing, run these lines to avoid conflicts with outdated packages:

```
sudo apt update && sudo apt full-upgrade -y
```

Now you are ready for the next step!

Step 2: Mounting a Static IP on the Pi

For ease of access via PuTTy, it is a good idea to mount a static IP. For this step, we will transition to using dhcpcd5 instead of NetworkManager.

First, disable NetworkManager:

```
sudo systemctl disable NetworkManager --now
```

Then install dhcpcd5:

```
sudo apt install dhcpcd5
```

Next, configure dhcpcd:

```
sudo nano /etc/dhcpcd.conf
```

Inside of the file, write this:

```
interface wlan0
static ip_address=192.168.1.100/24 # replace with your actual IP
static routers=192.168.1.1 # replace with your actual IP
static domain_name_servers=8.8.8.8 8.8.4.4 # replace with your actual DNS
```

Restart dhcpcd:

```
sudo systemctl restart dhcpcd
```

At this point, all the background stuff is up and running. Wi-fi has been configured, a static IP has been mounted for internet access, and PuTTy can be used to SSH into the Pi. Now you are ready for actually getting the software going.

Step 3: Installing Dependencies (pyenv first)

The packages we will be needing are these:

- **python** (to run the OS scripts)
- **pyenv** (for multiple versions of Python)
- **openbox** (to render the OS/games)
- **git** (for OS updates; optional)

Although Python 3.11 is normally installed on the Raspberry Pi 5, it's good to ensure that it is installed before doing anything else.

```
sudo apt install python3
```

We will **not** be explicitly using the system's Python for the OS, but it is still a good idea to just validate that it's still there. What we will be using is **pyenv.** This is a Python manager that gives you the capability to have multiple versions of Python. Even then, it isn't actually used for runtime. It is used to create virtual environments when necessary.

Before installing pyenv, download all the prerequisites:

```
sudo apt install -y build-essential libssl-dev zlib1g-dev libbz2-dev
libreadline-dev libsqlite3-dev curl git libncursesw5-dev xz-utils tk-dev
libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
```

These are necessary for pyenv to run without issues. Notice that **git has been installed** as part of the prerequisites, so that's out of the way. Once installed, you can move on to pyenv:

```
curl https://pyenv.run | bash
```

Also, to make pyenv work, you need to put the following at the end of `~/.bashrc`:

```
export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
eval "$(pyenv init --path)"
eval "$(pyenv init -)"
```

Then:

```
source ~/.bashrc
```

Now you can install different versions of Python. The version we will be defaulting to is 3.13.

```
pyenv install 3.13
```

This can take up to an hour. Once installed, set 3.13 to be the default version:

```
pyenv global 3.13
```

To avoid Python issues when using SSH, put this in `/ home/shugrpi/.bash_profile` (via nano; not sudo nano):

```
if [ -f "$HOME/.bashrc" ]; then
    source "$HOME/.bashrc"
fi
```

Now Python is fully ready to use for the project.

Step 4: Setting up the OS Scripts

Create a master directory to house all of the OS files:

```
mkdir /home/shugrpi/master
```

Move into the right directory:

```
cd /home/shugrpi
```

First, write this in the command line to prevent future bugs:

```
git config --global init.defaultBranch main
```

Install the OS files:

```
git clone https://github.com/Stormwrecker/shugrpi_os_master.git master
```

Move into the right directory:

```
cd /home/shugrpi/master
```

Now, create the venv and install OS dependencies:

```
python -m venv .venv
```

```
source .venv/bin/activate
pip install -r requirements.txt
```

Now all the OS scripts are in the right place, and you are ready to start running these in Openbox.

Step 5: Setting up Openbox (and Automating it)

To install Openbox, run this:

```
sudo apt install -y \
  xserver-xorg \
  xinit \
  openbox \
  x11-xserver-utils \
  unclutter
```

and this:

```
sudo apt install -y wmctrl
```

These are all the packages required for Openbox.

Note: when writing the following files, do **NOT** use `sudo nano`! Use plain `nano`.

In `/home/shugrpi/.config/openbox/autostart`:

```
#!/bin/sh

# Disable screen blanking
xset -dpms
xset s off
xset s noblank

unclutter &

/home/shugrpi/master/run_os.sh &
```

This is what will run whenever Openbox starts up. The file referenced on the bottom does not exist yet, so keep reading.

In `/home/shugrpi/.xinitrc`:

```
#!/bin/sh
exec openbox-session
```

In `/home/shugrpi/master/run_os.sh`:

```
#!/bin/bash

# Ensure DISPLAY is set
export DISPLAY=:0

# Optional but recommended for SDL clarity
export SDL_VIDEODRIVER=x11

# Change to project directory
cd /home/shugrpi/master/

# Activate virtualenv
source ./.venv/bin/activate

# Run your main script
python main.py

# If SHUGRPI exits, exit Openbox too
openbox --exit
```

This is the file that actually runs the OS. This is also the file that is referenced in the autostart file. Now make it executable:

```
chmod +x /home/shugrpi/master/run_os.sh
chmod +x /home/shugrpi/.xinitrc
```

Now the Openbox scripts are ready to run on boot. All that is left is to polish up the boot-up process and take care of some small details.

Step 6: Automating the Pi

Now it's time to automate the whole process. The desired flow is to

- Boot up the Pi
- Auto-login
- Run Openbox
- Run OS

Go to `/boot/firmware/config.txt` (**sudo nano**) and add these:

```
[all]
dtoverlay=pwm-fan
disable_splash=1
```

In `/boot/firmware/cmdline.txt` (**sudo nano**), add this**:**

```
rootwait quiet loglevel=3 vt.global_cursor_default=0 splash
```
(**MAKE SURE EVERYTHING IS ON ONE LINE!**)

This will silence the boot-up process. Next, make the following directory:

```
sudo mkdir -p /etc/systemd/system/getty@tty1.service.d
```

Then create an override file for the login process:

```
sudo nano /etc/systemd/system/getty@tty1.service.d/override.conf
```

Add this:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin shugrpi --noclear %I $TERM
```

Then reload the service:

```
sudo systemctl daemon-reexec
sudo systemctl restart getty@tty1
```

In `/home/shugrpi/.bash_profile` (nano, **not** sudo nano), add these to the bottom of the file:

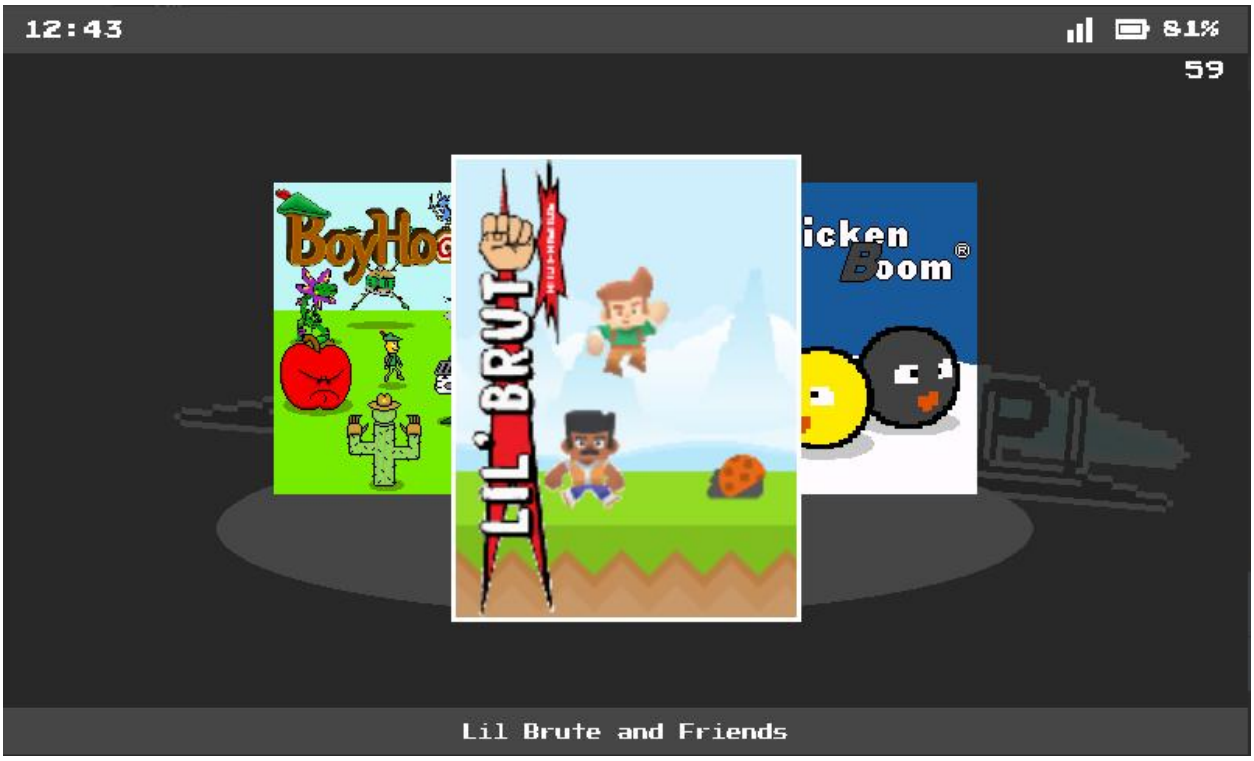
```
if [ -z "$DISPLAY" ] && [ "$(tty)" = "/dev/tty1" ]; then
    while true; do
        startx
        sleep 1   # prevent tight loop
    done
fi
```

To test everything:

```
sudo reboot
```

What now should happen is that the Pi boots up quietly, logs in, starts Openbox, and runs the OS automatically.

Congratulations! You now have the software setup of the SHUGRPI. This is a good spot to stop as now both the OS and Linux are good and ready to go. You could move on to the hardware end of things, but at this point, you are able to plug-and-play. Happy gaming!



The End