




Openauto

How to set up the initial image of Open Auto

- Install OpenAuto on microSD
- Insert microSD into RPI ZERO 2W and boot
- After powering off the RPI, remove the microSD and connect it to the external memory of the PC.
-  [Download boot folder from PI-SIGHT_SW_Openauto](#)
- Copy the config.txt file from the downloaded boot folder and paste it into the boot directory of the microSD.

Paste

- Modify and copy the crankshaft_env.sh file in the downloaded /boot/crankshaft folder and paste it into the /boot/crankshaft directory of the microSD (modify it with your own network SSID and PW)

```
/boot/crankshaft/crankshaft_env.sh # Wifi client mode (use this entry for home/work wifi -  
not for phone's hotspot) # If your SSID or password contains special chars or  
spaces make sure using quotation marks ="SSID" / ="password" WIFI_SSID="PIPA"  
WIFI_PSK="qwertyuiop" # Hotspot (if enabled the wifi client is disabled and a hotspot is  
opened) # Hotspot has now a default password (1234567890) -> changeable in /  
etc/hostapd/hostapd.conf if really needed! ENABLE_HOTSPOT=0
```

- Copy the config.txt file from the downloaded boot folder and paste it into the boot directory of the microSD. Put it in
- Copy the openauto.ini and hotspot.conf files from the /boot/crankshaft folder and paste them into the /boot/crankshaft directory on the microSD.
- Disconnect microSD from PC, connect it to RPI and boot it.
- RPI SSH connection from PC
- sudo raspi-config y camera on
- Edit file

```
#Modify contents nano /etc/resolv.conf nameserver 8.8.8.8 #Modify contents nano /  
etc/fstab /boot vfat defaults,noatime,nodiratime 0 2 #Add contents nano  
/etc/rc.local sh /boot/crankshaft/bootrun.sh
```

- Install required components

```
sudo apt-get update sudo apt-get upgrade sudo pip install --upgrade pip sudo pip3 install picamera keyboard pyautogui
```

- Setting display output related to pyautogui

```
sudo apt-get install python3-tk python3-dev sudo pip3 install python-xlib sudo echo "DISPLAY=:0" >> ~/.bashrc sudo source ~/.bashrc
```

- Bluetooth controller connection settings

```
sudo apt-get install bluetooth blueman bluez bluez-cups libbluetooth-dev pi-bluetooth python-bluez
sudo sed -i 's/Privacy = off/# Privacy = off/' /etc/bluetooth/main.conf sudo sed -i 's/ControllerMode =
bredr/#ControllerMode = dual/' /etc/bluetooth/main.conf sudo sed -i
's/FastConnectable = true/#FastConnectable = false/'
/etc/bluetooth/main.conf sudo systemctl disable tap2wake sudo systemctl disable csng-bluetooth sudo
systemctl disable btautoair sudo systemctl disable btdeviceconnect sudo systemctl disable btstore
sudo systemctl disable btdevice sudo systemctl enable bluetooth sudo systemctl enable hciuart sudo
systemctl enable ofono sudo sed -i 's/load-module module-bluetooth-discover headset=ofono/load-
module module-bluetooth-discover autotdetect_mtu=yes/' /etc/pulse/default.pa sudo sed -i 's/load-
module module-bluetooth-discover headset=ofono/load-module module-bluetooth-
discover autotdetect_mtu=yes/' /etc/pulse/system.pa sudo nano /etc/fstab -> /tmp/bluetooth ħ ħ sudo
rm -r /var/lib/bluetooth sudo mkdir /var/lib/bluetooth sudo chmod -R 777 /var/lib/bluetooth sudo
chown -R
pi /var/lib/bluetooth
```

- Pairing remote control with bluetoothctl manual command, trust test

- Reboot

- The remote control is normally stored in paired-devices using the bluetoothctl manual command.

Check it out

- Disable unnecessary services

```
sudo systemctl disable gpio2kbd sudo systemctl disable  
usbdetect.service sudo systemctl disable usbrestore.service sudo systemctl  
disable triggerhappy.service sudo systemctl disable triggerhappy.socket sudo  
systemctl disable gpsd
```

- Create LED off system service

```
sudo nano /etc/systemd/system/disable-led.service [ Unit ]  
Description=Disables the LED After=multi-user.target [ Service ]  
Type=oneshot RemainAfterExit=yes ExecStart=sh -c "echo 0 | sudo tee /sys/class/  
leds/led0/brightness > /dev/null" ExecStop=sh -c "echo 1 | sudo tee /sys/class/leds/led0/  
brightness > /dev/null" [ Install ]  
WantedBy=multi-user.target sudo systemctl enable disable-led.service
```

- LED off System service running

```
sudo systemctl enable disable-led.service
```

- Edit file

```
#Modify contents /etc/resolv.conf nameserver 8.8.8.8
```

- I2S MIC Settings

```
sudo pip3 install --upgrade adafruit-python-shell sudo wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2smic.py
sudo python3 i2smic.py
```

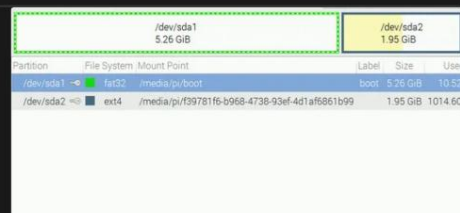
Install after checking that the RPI ZERO 2W device is loaded in the i2smic.py script.

- Power off the RPI
- Remove the microSD and connect it to another Linux device as external memory.
- Working with microSD GParted on other Linux devices (extending boot partition)

Resizing the Raspberry Pi Boot Partition

Though the Raspberry Pi computer is eminently networkable, some projects still just work best by physically moving the SD

★ <https://learn.adafruit.com/resizing-raspberry-pi-boot-partit...>



Partition	File System	Mount Point	Label	Size	Used
/dev/sda1	FAT32	/media/pi/boot	boot	5.26 GiB	10.5%
/dev/sda2	ext4	/media/pi/f39781f6-b968-4738-93ef-4d1aff5861b99		1.95 GiB	1014.60%

- After working with GParted, connect the microSD as external memory to your PC.
- After GParted is done, copy all the files in the downloaded /boot/crankshaft folder.
Paste it into the /boot/crankshaft directory on your microSD
- Create /boot/Video Converter directory and paste My MP4Box GUI.zip
- Disconnect microSD from PC, connect it to RPI and boot it.
- Remote control pairing, remote control key input, Android Auto operation, microphone input, camera operation, etc.
Test all features
- After powering off, remove the microSD and connect it to the PC's external memory.