MCE 6699 Camera Project Notes

Nolan Chandler

Fall 2022

Contents

Project description

The goal of the project is to spec out and configure a wireless camera system with multiple camera modules that can be easily mounted near the end effector of a robot. The camera feeds need to be accessible to use in computer vision applications, such as motion detection and object identification and tracking. A setup like this can also be used to provide additional safety measures to people in the vicinity of the robot, as well as provide views for remotely operating the robot.

Raspberry Pi Zero configuration

Quick Note

All the files mentioned in these instructions can be found on my personal GitHub account (TODO TODO TODO)

Current configuration

`root` account is disabled. Use `sudo` to run administrative tasks.

- Username: `pi`
- Password: `idahostateMCE`
- IP: `10.42.0.1` on WiFi, `10.43.0.1` on Bluetooth

Some initial configuration

All of the following configurations assume a freshly flashed Raspberry Pi OS "Bullseye" image, which can be found here:

https://www.raspberrypi.com/software/

Once the image has been flashed, boot the Pi with the SD card inserted. Once booted, open the terminal and enter the command:

```
sudo raspi-config
```

The following configurations will be done using this tool.

Change hostname

1. System Options, S4 Hostname
   - change from "raspberry" to whatever, I used "robocam-00"
   - password can be changed from here too, if needed.

Boot to command line instead of graphical desktop

1. System Options, S5 Boot / Auto Login
   - Console Autologin, this automatically logs in default user for userland tasks
   - Desktop session can be started anytime with command `startx`

Saves a lot of system memory not having a desktop running that no one is using

Enable SSH

3. Interface Options, I2 SSH
    - Enable. This allows remote logins while connected to Pi's network

While connected to the Pi with any well-configured network, a tool like PuTTY can reach the Pi by its hostname, in the domain `.local`. Otherwise, it can be reached by either of its static IP addresses.

```
ssh pi@robocam-00.local
ssh pi@10.42.0.1 # only on wifi
ssh pi@10.43.0.1 # only on BT
```

Enable Glamor (for viewing camera streams)

6. Advanced Options, A8 Glamor
    - Enable. This will allow `libcamera` apps to draw with graphics acceleration either from the CLI or from a desktop session. Absolutely necessary for Pi 0

Enable Network Manager

6. Advanced Options, AA Network Config
    - Choose "NetworkManager". Older Pi images used "dhcpd" to handle networking. NetworkManager makes setting up the Pi as a network host much, much simpler
    - If NetworkManager is not available, install with `sudo apt install networkmanager`

Wifi Network Host

In the in included folder "wifi-cfg", there are a couple of files including a setup.txt explaining some installation steps. More importantly, there is a '.nmconnection' file with the configuration for creating a WiFi hotspot. The current configuration is:

- SSID: `robocam-00`
- PSK (passphrase): `idahostateMCE`

Under the block `[ipv4]`, you may also want to configure the Pi's static IP address on this network. The current configuration sets up the Pi as `10.42.0.1`, but different subnets can be chosen if desired.

Once edited, the connection profile should be copied into NetworkManager's folder and given the proper, restricted permissions:

```
sudo cp robocam-00.nmconnection /etc/NetworkManager/system-connections/
sudo chmod 600 /etc/NetworkManager/system-connections/robocam-00.nmconnection
```

After a reboot, the network should start automatically.

Bluetooth Network Host (PAN)

In `/etc/bluetooth/main.conf`, find and uncomment line:

```
#DiscoverableTimeout = 0
```

This sets the Pi's bluetooth to always be discoverable to devices around it.

btpan-cfg

There are a number of files that need to be placed in different locations in the Pi's filesystem to automatically start and manage a bluetooth network. A "makefile" is provided in the folder as a templated list of each of these files and their destinations, although some may change based on the available version of Python, for example.

```
install:
    sudo cp ./pan0.nmconnection /etc/NetworkManager/system-connections/
    sudo chmod 600 /etc/NetworkManager/system-connections/pan0.nmconnection
    sudo cp ./btpan /usr/local/sbin/
    sudo cp ./bluezutils.py /usr/local/lib/python3.9/dist-packages
    sudo cp ./btpan.service /etc/systemd/system/
    sudo systemctl enable btpan.service
    sudo chmod +x /usr/local/sbin/btpan
```

If the destinations exist, the makefile can be used as an installation script simply by issuing the command `make`. This will move the scripts to those locations and enable a system-controlled service to handle adding and removing clients to the network. Once installed and operational, the network can be stopped at anytime using `systemctl`:

```
sudo systemctl stop btpan.service
```

As with the WiFi network configuration, a static address can be configured for the Pi on this network, located in the file `pan0.nmconnection`. Currently the Pi is set up as `10.43.0.1` so as to not conflict with the WiFi network. The other scripts rely on the existing configuration in this file, so other changes (such as the name of the bridge interface, "pan0") must be reflected in the other files.

In order to use the bluetooth network, your device must first be paired to the Pi. The process differs for different devices, but all pairing must be accepted manually from the Pi. This is done as follows, from a command line:

1. Start `bluetoothctl`. This will start the interface on the Pi.
2. Make your own device discoverable. This will vary on different platforms.
3. On the Pi, issue the command `scan on`. This will begin a search that will find your device and display some information about it, most notably its MAC address.
4. Once found, establish trust using the command `trust [MAC_ADDRESS]`, substituting your own MAC (hint: start typing the address and press Tab to autocomplete). This will prevent having to manually authorize connection attempts from your device. Once this is done, your device no longer needs to be discoverable.
5. On your own device, search for the Pi (it will broadcast itself as its hostname), and issue a pairing request (may vary)
6. On the Pi, *quickly* accept the pairing request, verifying any codes it presents.
    - Your own device might ask for confirmation; be sure to accept these as well.
7. Once paired and connected, follow your system's directions for connecting to a bluetooth Personal Area Network (PAN).

This network connection provides the same functionality as a WiFi connection, with different caveats such as a much smaller data bandwidth, but decreased power consumption.

USB Ethernet Connection

With the Pi acting as a network host on both WiFi and Bluetooth, it became important for me to be able to provide internet access to the Pi to update to run updates and retrieve tools. The Pi 0 is able to connect directly to a computer using a micro-USB to USB-A cable connected to its USB port (not power port) and establish a network with that computer as if an ethernet cable was connected.

I used this connection to share my laptop's wireless internet from the school network without needing to disable the Pi's managed networks and provide internet access.

Only a few configuration changes are required to enable this interface:

1. `/boot/config.txt`
   - Add a line reading `dtoverlay=dwc2` at the bottom, under the `[all]` block.
2. `/boot/cmdline.txt`
   - On the same line and following `rootwait`, add the argument `modules-load=dwc2,g_ether`
3. NetworkManager configuration
   - Copy the included configuration to NM's directory:

   ```
   sudo cp EthernetGadget.nmconnection /etc/NetworkManager/system-connections/
   sudo chmod 600 /etc/NetworkManager/system-connections/EthernetGadget.nmconnection
   ```
4. `/etc/rc.local`
   1. Add the following line to the file, before `exit 0`:

   ```
   $(nmcli c up EthernetGadget &>/dev/null) &
   ```
5. Reboot to enable changes

Once the Pi has been set up, your connected device will need to be configured to share its internet connection over the new "Ethernet" interface. This process will vary based on your system. The following link shows instructions that will probably work on Windows 10/11:

https://www.tomshardware.com/how-to/share-internet-connection-windows-ethernet-wi-fi

Camera setup

A quick note

The Raspberry Pi camera is well-documented, and all of the following information can be found in the official documentation:

https://www.raspberrypi.com/documentation/accessories/camera.html

`libcamera` installation

Up to date Raspberry Pi images are no longer using the `Raspicam` libraries that have been around for some time, and have instead switched to a more cross-compatible library, `libcamera`. It is included on newer Pi images, but if not can be installed using `apt`:

```
sudo apt install libcamera-apps
```

`libcamera` test

A simple test can be run using `libcamera-hello`. This will open a preview window of what the camera sees. This test can help diagnose connection issues between the Pi and the camera module, as it prints out some diagnostic information about the detected (or not) sensor.

`picamera2`

`picamera2` is the Raspberry Pi foundation's newest user-friendly Python API for camera functions. It is installed by default on new images, but can be installed manually:

```
sudo apt install python3-picamera2
```

The entire project is very well-documented and easy to use. The official documentation detailing its usage can be found on the official Raspberry Pi website:

https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf

Simple Web Server

For testing purposes, I wanted to setup the Pi to host an HTTP server/webpage displaying the camera's view, so that any network connected device would be able to view the stream. Thankfully, the authors of `picamera2` wrote an example script, `mjpeg_server.py`, that does exactly this.

The easiest way to obtain this script is by cloning the `picamera2` GitHub repo.

```
sudo apt install git
git clone https://github.com/raspberrypi/picamera2
cd picamera2/examples
```

Once the script has been cloned, it can be run to start the server:

```
python3 ./mjpeg_server.py
```

Once this has been started, any network connected device should be able to open the webpage being served by the Pi using a web-browser and the following URL:

http://robocam-00.local:8000

If your device is unable to resolve the hostname (common on Windows, but untested), either of the Pi's static IP addresses can be substituted for the hostname according to the network your device is connected to:

- WiFi: http://10.42.0.1:8000
- Bluetooth: http://10.43.0.1:8000

Alternate Hardware Considerations

For the motion detection function, a camera with a wide-angle/fish-eye lens will provide a wider view.

Hardware	Price (MSRP)	Considerations
NVIDIA Jetson Nano 4GB	$99.00	• 2 MIPI CSI-2 connectors, 12 lanes – Able to run 4 cameras simultaneously • Large list of supported camera sensors • Designed for AI/CV processing applications • Relatively high power draw, up to 10W
M5stack ESP32 Unit Cam DIY Kit	$18.95	• Inexpensive • Kit provides OV2640 (2MP) sensor and wide-angle lens • Low power • Powered by standard LiPo battery, charging integrated • Requires other system to process images
PiHut ZeroCam FishEye	$14.02	• 5MP sensor with wide-angle lens • Intended for use with RPi Zero, other Pi models require an adapter