

2021

Raspberry Pi Payload

USER MANUAL
RONNEL WALTON

Table of Contents

Hardware.....	2
Raspberry Pi Parts List	2
Ubiquiti Parts List.....	2
Other Utilities:	3
Software	3
Starting out.....	3
Connecting through SSH	5
Updating Fresh Raspberry Pi OS	6
Camera System setup	7
Getting the Camera Working	7
Camera Stream on the Network	8
Automation Time.....	14
Creating Script File	14
Creating Service File	16

Hardware

Raspberry Pi Parts List

Name	#	Link	Price (\$) [Single]
Raspberry Pi 4 B 8GB (Kit that includes USB C Power Supply)	1	https://www.raspberrypi.com/products/raspberry-pi-4-model-b/	83.00 [83.00]
Arducam Synchronized Stereo Camera Bundle Kit	1	https://www.arducam.com/product/arducam-mini-12mp-2-synchronized-stereo-camera-bundle-kit-for-raspberry-pi-and-pi-zero-two-12-3mp-imx477-camera-modules-with-m12-lens-and-camarray-stereo-camera-hat-b0347r/	204.99 [204.99]
32 GB Micro SD Card	1		8.00-11.00 [8.00-11.00]
Arducam 190 Degree Fisheye 1/2.3" M12 Lens	2	https://www.arducam.com/product/arducam-190-degrees-lens-for-hq-camera-ln074/	60 [30]
Raspberry Pi 4 Aluminum Case	1	https://www.amazon.com/dp/B07VD568FB?ref_=cm_sw_r_cp_ud_dp_HKQ1H6H0NRKXGAZB9K6P	13.99 [13.99]
Subtotal:			369.98 – 372.98

Ubiquiti Parts List

Name	#	Link	Price (\$) [Single]

Ubiquiti airMAX BaseStation (ROCKETM5)	1	https://store.ui.com/collections/operator-airmax-devices/products/rocket-m5	89

Other Utilities:

Name	#	Link	Price (\$) [Single]
USB Mouse*	1		
USB Keyboard*	1		
Micro HDMI to HDMI*	1	https://www.amazon.com/Rankie-Micro-Supports-Ethernet-Return/dp/B00Z07JYLE	8.99 [8.99]

[*These are used for raspberry pi 4]

Software

Starting out

Raspberry Pi is a small computer that can be plugged in to a monitor, mouse, and keyboard. Raspberry Pi is extremely versatile can be used for number of uses. For BOREALIS's case, it is used as a camera system that host camera's stream and saves the camera footage.

The operating system (OS) can be downloaded from the official Raspberry Pi website (<https://www.raspberrypi.com/>). The download link can be found at <https://www.raspberrypi.com/software/>. The installed OS version at the time of this manual was version 10 buster(Legacy). This version may not be available at the time this manual is used (2022-04-04).

To get started, the OS needs to be loaded to the micro-SD card.

1. Go to <https://www.raspberrypi.com/software/>. Download and Install Raspberry Pi Imager.
2. Once installed, open Raspberry Pi Imager.
3. Select the gear icon at the bottom right of the imager.
4. Check the “Enable SSH” checkbox. This is how other computers can be used to connect to the raspberry pi while being the same network. More on this later.
5. Select Choose OS and then Raspberry Pi OS (Legacy).
6. Select Choose Storage and then select the inserted micro-SD card. The micro-SD card will be overwritten so make sure there is a backup of anything needed from the card.
7. Select Write. This will take some time. With a good computer, it will take about 15-25 minutes. The micro-SD card will be automatically unmounted by the imager.

The micro-SD card is ready to be inserted into the raspberry pi 4. Connect the raspberry pi to a monitor with the Micro HDMI to HDMI cable. Plug in the mouse and keyboard as well. Plug in the power supply and the raspberry pi 4 should come to life and show some booting process on the monitor. Check Figure 2 for the connections. It should come up with the Raspberry Pi OS desktop. If it doesn't, switch the port for the display or reimage the micro-SD card and try again. It may come up with a warning regarding SSH being enabled due to the password being set to default. Go through the Welcome screen steps.

1. Set Country, Language, and Timezone
2. Set Password
3. Check if the screen is properly shown
4. Login to a network
5. Select Next to do Software Update. **This will take up to an hour depending on the network speed.**
6. Select Restart

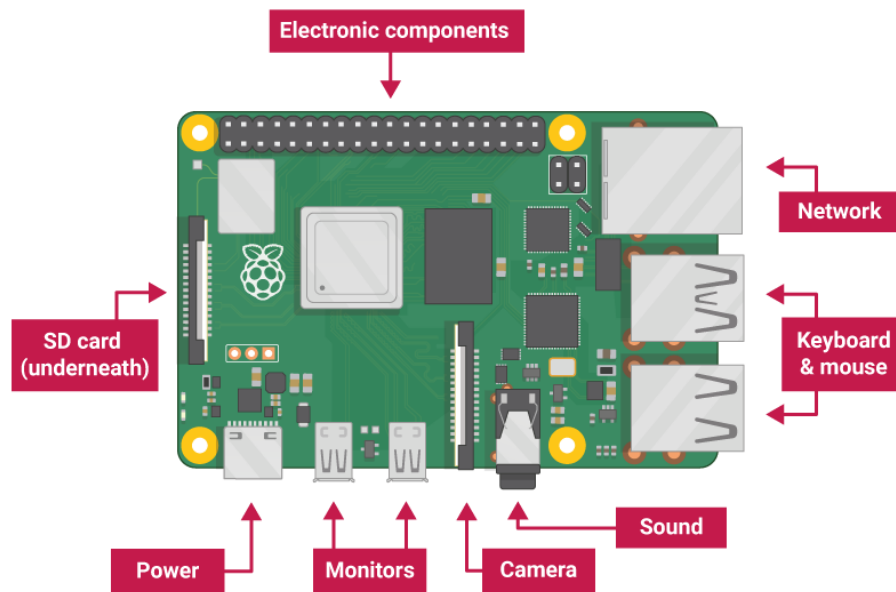


Figure 1 Raspberry Pi 4 Connections

[Insert Desktop Screenshot]

Connecting through SSH

Secure Shell Protocol (SSH) is a way for another computer to connect to the raspberry pi and be able to configure it wirelessly. Two recommended software that can be used are PuTTY (<https://www.putty.org/>) and MobaXTerm (<https://mobaxterm.mobatek.net/>).

1. Download and Install PuTTY or MobaXTerm.
2. On the Raspberry Pi, open the terminal. On the Terminal, Type in "ifconfig" and then press Enter.
 - a. Take note of inet. Since it is configured to connect to the WiFi, check inet under wlan0.
3. PuTTY
 - a. Open PuTTY
 - b. In Host Name, Type in the inet.
 - c. Select Open. Select Accept when prompt to trust.

- d. Enter the credentials
4. MobaXTerm
- a. Open MobaXterm
 - b. Select Session on the top left.
 - c. Select SSH. In the Remote host, Type in the inet. Select OK.
 - d. Select Accept when prompt to trust.
 - e. Enter the credentials

Updating Fresh Raspberry Pi OS

After login, the OS will require software update. This may take up to an hour depending on how outdated the image and the network download speed. **Please make sure to have a stable network during this process.**

1. Open the terminal (Ctrl + Alt + T or Raspberry Pi Start button > Accessories > Terminal).
 - a. Terminal is a magical place where most of our work will be done. You will use the terminal A LOT.
2. Type in “sudo apt-get update -y && sudo apt-get upgrade -y” and then press Enter. **This will take up to an hour depending on the network speed.** Or no time if updated at the welcome screen.
 - a. sudo means running the command with elevated permissions. Use with caution. Will be used a lot in installing software and compiling. Most of the time, will ask for the root user password. If the current user is pi, it is a root user. Use the password for pi.
 - b. apt-get is one of the package managers of the OS. This is used to install or remove software.
 - c. update option for apt-get means updating the repository of the package manager.
 - i. Repository is where software files and folders live.
 - d. upgrade option for apt-get means upgrading the existing software on the OS installed by the apt-get.

- e. -y means answer yes to user prompts.
 - f. && means running both commands. One command will be run at a time. The order will be from left to right. The next command will only run if the first one ran successfully.
3. Type in “sudo reboot -h 0” to reboot the raspberry pi right away.

The raspberry pi is ready to be configured for other uses. In our case, the camera system.

Camera System setup

Depending on the camera kit obtain, the connection of the camera may differ. Most manufacturers provide a manual how to do so. In this manual, the Arducam Stereo Hat kit is used. Please check the Arducam’s documentation/video on putting the kit together (<https://www.arducam.com/docs/cameras-for-raspberry-pi/synchronized-stereo-camera-hat/camarry-12mp-synchronized-stereo-hat/#8-hardware-setup>). The software part will be summarized in this manual. Some troubleshooting is included in Arducam’s documentation as well.

Getting the Camera Working

Note: At the time of the creation of this manual, Raspberry Pi is moving forward from the sunny camera/legacy camera. Following steps may not work in the future.

1. Open Terminal (Ctrl + Alt + T).
2. Type in “sudo raspi-config” and then press Enter.
 - a. raspi-config opens the raspberry pi preference meaning in the terminal
3. Select (up/down arrow keys to move up and down) Interface Options and then Camera. Select Yes (left/right to move left and right) when prompt to enable camera.
4. Repeat the same thing for I2C.

5. Select Performance Options and then GPU Memory. Type replace the current memory with 256 and select Ok.
6. Select Finish. When prompt to Reboot, select Yes.
7. Open Terminal (Ctrl + Alt + T).
8. Type in “raspistill -t 0” and then press Enter
 - a. raspistill is one the built-in camera application.
 - b. -t 0 means open the camera indefinitely
9. The camera preview should show. Press Ctrl + C to exit the camera preview if it’s still showing.
 - a. Ctrl + C is Keyboard Interrupt. In case you want to cancel anything you ran, this will be your best friend.

Camera Stream on the Network

At this point, the camera can be confirmed to be properly working. Next part will be for configuring the camera to stream in the network. This part will involve A LOT of Terminal commands. Be careful of typos or download this manual (pdf version) into the raspberry pi OS. Another option is using SSH (Check chapter SSH for more info). Copy-Paste is a thing. All highlighted text is entered in the terminal. If you like to cancel the operation at anytime use Ctrl + C.

The software used to stream and save the camera footage is GStreamer (<https://gstreamer.freedesktop.org/>). GStreamer is an open-source software for multimedia processing. The process of installing the software was adapted from Q-engineering (<https://qengineering.eu/install-gstreamer-1.18-on-raspberry-pi-4.html>). This process will be to install GStreamer version 1.18 and its plugins (good, bad, and ugly). Check out the documentation for GStreamer at <https://gstreamer.freedesktop.org/documentation/>. Grab some coffee (or tea) and a snack before you start, this will take a while (For me, it was about 1-2 hours).

1. Create GStreamer directory and enter it: `cd ~ && mkdir GStreamer && cd GStreamer`
 - a. ~ means home directory of the current user
2. Install utilities: `sudo apt-get install cmake meson ninja-build flex bison -y`
3. Install dependencies: `sudo apt-get install libglib2.0-dev libjpeg-dev libx264-dev libgtk2.0-dev libcanberra-gtk* libgtk-3-dev -y`
4. Install GStreamer Core:
 - a. Create Core directory and enter it: `mkdir Core && cd Core`
 - b. Download the GStreamer Core: `wget https://gstreamer.freedesktop.org/src/gstreamer/gstreamer-1.18.4.tar.xz`
 - c. Extract the downloaded tar.xz: `sudo tar -xf gstreamer-1.18.4.tar.xz`
 - d. Enter the newly extracted: `cd gstreamer-1.18.4`
 - e. Create and Enter build directory: `mkdir build && cd build`
 - f. Prepare to build (Configure how to build it): `meson --prefix=/usr --wrap-mode=nofallback -D buildtype=release -D gst_debug=true -D package-origin=https://gstreamer.freedesktop.org/src/gstreamer/ -D package-name="GStreamer 1.18.4 BLFS" ..`
 - g. Build (This will take a while): `ninja -j4`
 - h. Install the libraries: `sudo ninja install && sudo ldconfig`
5. Install GStreamer Base Plugins:
 - a. Install other dependencies: `sudo apt-get install libegl1-mesa-dev libglfw3-dev libgles2-mesa-dev -y`
 - b. Return to GStreamer Directory: `cd ~/GStreamer`
 - c. Create and Enter Base Directory: `mkdir Base && cd Base`

- d. Download GStreamer Base Plugin: `wget`
<https://gstreamer.freedesktop.org/src/gst-plugins-base/gst-plugins-base-1.18.4.tar.xz>
 - e. Extract the downloaded tar.xz file: `sudo tar -xf gst-plugins-base-1.18.4.tar.xz`
 - f. Enter the extracted folder: `cd gst-plugins-base-1.18.4`
 - g. Create and Enter Build folder: `mkdir build && cd build`
 - h. Prepare to build (Configuring how to build it): `meson --prefix=/usr -D gl_winsys=wayland -D buildtype=release -D package-origin=https://gstreamer.freedesktop.org/src/gstreamer/ ..`
 - i. Build (This will take a while): `ninja -j4`
 - j. Install libraries: `sudo ninja install && sudo ldconfig`
6. Install GStreamer Good Plugins:
- a. Install other dependencies: `sudo apt-get install libjpeg-dev -y`
 - b. Return to GStreamer Directory: `cd ~/GStreamer`
 - c. Create and Enter Good Directory: `mkdir Good && cd Good`
 - d. Download GStreamer Good Plugin: `wget`
<https://gstreamer.freedesktop.org/src/gst-plugins-good/gst-plugins-good-1.18.4.tar.xz>
 - e. Extract the downloaded tar.xz file: `sudo tar -xf gst-plugins-good-1.18.4.tar.xz`
 - f. Enter the extracted folder: `cd gst-plugins-good-1.18.4`
 - g. Create and Enter Build folder: `mkdir build && cd build`
 - h. Prepare to build (Configuring how to build it): `meson --prefix=/usr -D buildtype=release -D package-origin=https://gstreamer.freedesktop.org/src/gstreamer/ -D package-name="GStreamer 1.18.4 BLFS" ..`
 - i. Build (This will take a while): `ninja -j4`

- j. Install libraries: `sudo ninja install && sudo ldconfig`
7. Install GStreamer Bad Plugins:
 - a. Install other dependencies: `sudo apt-get install librtmp-dev libvo-aacenc-dev -y`
 - b. Return to GStreamer Directory: `cd ~/GStreamer`
 - c. Create and Enter Bad Directory: `mkdir Bad && cd Bad`
 - d. Download GStreamer Bad Plugin: `wget https://gstreamer.freedesktop.org/src/gst-plugins-bad/gst-plugins-bad-1.18.4.tar.xz`
 - e. Extract the downloaded tar.xz file: `sudo tar -xf gst-plugins-bad-1.18.4.tar.xz`
 - f. Enter the extracted folder: `cd gst-plugins-bad-1.18.4`
 - g. Create and Enter Build folder: `mkdir build && cd build`
 - h. Prepare to build (Configuring how to build it): `meson --prefix=/usr -D buildtype=release -D package-origin=https://gstreamer.freedesktop.org/src/gstreamer/ -D package-name="GStreamer 1.18.4 BLFS" ..`
 - i. Build (This will take a while): `ninja -j4`
 - j. Install libraries: `sudo ninja install && sudo ldconfig`
 8. Install GStreamer Ugly Plugins:
 - a. Install other dependencies: `sudo apt-get install libx264-dev`
 - b. Return to GStreamer Directory: `cd ~/GStreamer`
 - c. Create and Enter Ugly Directory: `mkdir Ugly && cd Ugly`
 - d. Download GStreamer Ugly Plugin: `wget https://gstreamer.freedesktop.org/src/gst-plugins-ugly/gst-plugins-ugly-1.18.4.tar.xz`
 - e. Extract the downloaded tar.xz file: `sudo tar -xf gst-plugins-ugly-1.18.4.tar.xz`
 - f. Enter the extracted folder: `cd gst-plugins-ugly-1.18.4`
 - g. Create and Enter Build folder: `mkdir build && cd build`

- h. Prepare to build (Configuring how to build it): `meson --prefix=/usr -D buildtype=release -D package-origin=https://gststreamer.freedesktop.org/src/gststreamer/ -D package-name="GStreamer 1.18.4 BLFS" ..`
 - i. Build (This will take a while): `ninja -j4`
 - j. Install libraries: `sudo ninja install && sudo ldconfig`
9. Install omxh264enc (Only for 32-bit OS):
- a. Return to GStreamer Directory: `cd ~/GStreamer`
 - b. Create and Enter Omxh264enc Directory: `mkdir Omxh264enc && cd Omxh264enc`
 - c. Download GStreamer Omxh264enc Plugin: `wget https://gststreamer.freedesktop.org/src/gst-omx/gst-omx-1.18.4.tar.xz`
 - d. Extract the downloaded tar.xz file: `sudo tar -xf gst-omx-1.18.4.tar.xz`
 - e. Enter the extracted folder: `cd gst-omx-1.18.4`
 - f. Create and Enter Build folder: `mkdir build && cd build`
 - g. Prepare to build (Configuring how to build it): `meson --prefix=/usr -D header_path=/opt/vc/include/IL -D target=rpi -D buildtype=release ..`
 - h. Build: `ninja -j4`
 - i. Install libraries: `sudo ninja install && sudo ldconfig`
10. Install RTSP Server:
- a. Return to GStreamer Directory: `cd ~/GStreamer`
 - b. Create and Enter RTSPServer Directory: `mkdir RTSPServer && cd RTSPServer`
 - c. Download RTSP Server source code: `wget https://gststreamer.freedesktop.org/src/gst-rtsp-server/gst-rtsp-server-1.18.4.tar.xz`
 - d. Extract tar.xz file: `tar -xf gst-rtsp-server-1.18.4.tar.xz`

- e. Enter the newly extracted folder: `cd gst-rtsp-server-1.18.4`
- f. Modify test-launch.c to change where the stream will be viewed (This is what you put in VLC later. Make sure to take note): `nano examples/test-launch.c`
 - i. Press down arrow till `"gst_rtsp_mount_points_add_factory (mounts, "/test", factory);"` is found. Change `"/test"` to `"/payload"` or something else preferred.
 - ii. Press `Ctrl + X`. Press `y` and then `Enter`.
- g. Create and Enter Build/Compilation Directory: `mkdir build && cd build`
- h. Build: `meson --prefix=/usr --wrap-mode=nofallback -D buildtype=release -D package-origin=https://gstreamer.freedesktop.org/src/gstreamer/ -D package-name="GStreamer 1.18.4 BLFS" ..`
- i. Compile: `sudo ninja -j4`
- j. Install the compiled code: `sudo ninja install && sudo ldconfig`

At this point, the camera can stream through the network now. Pat yourself in the back for taking the time. Time to test!

1. Create Streaming folder: `cd ~ && mkdir BOREALIS_Cam && cd BOREALIS_Cam`
2. Copy the software for streaming: `cp ~/GStreamer/RTSPServer/gst-rtsp-server-1.18.4/build/examples/test-launch ./`
3. Create a file for the GStreamer parameter: `nano test.txt`
 - a. Type in `"rpicamsrc bitrate=8000000 preview=false ! video/x-h264, width=640, height=480, framerate=30/1 ! h264parse ! rtpH264pay name=pay0 pt=96"`
 - b. Press `Ctrl + X`, type `y`, and then press `Enter`
4. On the Raspberry Pi

- a. Open Terminal (Ctrl + Alt + T)
- b. Go to the Streaming folder: `cd ~/BOREALIS_Cam`
- c. Start Stream: `./test-launch "`cat test.txt`"`
 - i. `.` meaning run the object
 - ii. `test-launch` is the object modified in rtsp server and compiled
 - iii. ``` is a backtick not single quotation. It allows the output of the command inside the backticks to be used as a variable.
 - iv. `cat` is a command for reading the data in the file
- d. When done with testing. Ctrl + C to stop streaming.

At this point, the system is functional as a camera streaming system. Next part is automations and saving footage.

Automation Time

This part of the manual will help in automating the camera system. After this part, no need to touch the raspberry pi again. Except when retrieving the saved footage.

Creating Script File

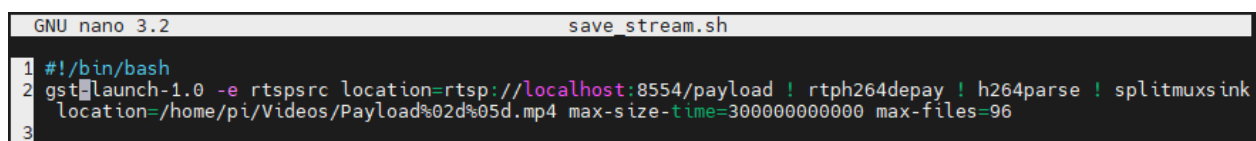
Tired of typing long commands? This part will teach you how to create scripts files. These script or shell files can be ran instead of typing the long commands. All highlighted text is entered in the terminal. If you like to cancel the operation at anytime use Ctrl + C.

Let's make the script file for saving the camera stream footage.

1. Return to Streaming folder: `cd ~/BOREALIS_Cam`
2. Create and edit a script file: `nano save_stream.sh`
3. Type in the following:
`#!/bin/bash`

```
gst-launch-1.0 -e rtspsrc
location=rtsp://localhost:8554/payload ! rtph264depay !
h264parse ! splitmuxsink
location=/home/pi/Videos/Payload%02d%05d.mp4 max-
size-time=300000000000 max-files=96
```

- i. gst-launch-1.0 is gstreamer software installed
- ii. -e means exit gstreamer without corrupting the file created.
- iii. rtspsrc is the camera source.
- iv. rtsp://localhost:8554/payload is the link for viewing the camera
- v. rtph264depay extracts the h264 video from the rtp packets from the rtsp stream.
- vi. h264parse is for parsing the h264 video into a different file type. The following will indicate what the type is
- vii. splitmuxsink is for splitting the h264 video into different files and then converted into a different file type
- viii. For more information: Check <https://gstreamer.freedesktop.org/documentation/>



```
GNU nano 3.2 save_stream.sh
1 #!/bin/bash
2 gst-launch-1.0 -e rtspsrc location=rtsp://localhost:8554/payload ! rtph264depay ! h264parse ! splitmuxsink
3 location=/home/pi/Videos/Payload%02d%05d.mp4 max-size-time=300000000000 max-files=96
```

Figure 2 save_stream.sh file (Ctrl + Shift + 4 for word wrapping in nano)

4. Save files: Press Ctrl + X, press y and then press Enter
5. Make script executable: `chmod +x save_stream.sh`
 - a. chmod is the command for modifying the file permissions
 - b. + means adding a permission. x means execute permission
6. No need to run the script right now, something else will run it. If you like to run scripts in the terminal, type in `./<script_name>.sh`

Creating Service File

Don't want to run the scripts yourself? Let the computer do it for you.

For more information about Linux Service files, go to

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/assembly_working-with-systemd-unit-files_configuring-basic-system-settings . All highlighted text is entered in the terminal. If you like to

cancel the operation at anytime use Ctrl + C.

Let's make a service file for starting camera stream.

1. Return to Streaming folder: `cd ~/BOREALIS_Cam`
2. Create and edit service file for starting the stream: `nano BOREALIS_CAM.service`
3. Type in:

```
[Unit]
Description=BOREALIS_Payload_Cam_Livestream_service
After=network.target
Type=simple

[Service]
ExecStart=/home/pi/BOREALIS_Cam/test-launch "rpicamsrc
bitrate=8000000 preview=false ! video/x-h264, width=640,
height=480, framerate=30/1 ! h264parse ! rtph264pay
name=pay0 pt=96"
Restart=always

[Install]
WantedBy=multi-user.target
```

```
GNU nano 3.2 BOREALIS_CAM.service
1 [Unit]
2 Description=BOREALIS_Payload_Cam_Livestream_service
3 After=network.target
4 Type=simple
5
6 [Service]
7 ExecStart=/home/pi/BOREALIS_Cam/test-launch "rpicamsrc bitrate=8000000 preview=false
! video/x-h264, width=640, height=480, framerate=30/1 ! h264parse ! rtpH264pay name
=pay0 pt=96"
8 Restart=always
9
10 [Install]
11 WantedBy=multi-user.target
..
```

Figure 3 BOREALIS_CAM.service file

4. Save files: Press Ctrl + X, press y and then press Enter
5. Install the service file: `sudo cp BOREALIS_CAM.service /etc/systemd/system/ && sudo systemctl daemon-reload`
 - a. First command copies the service file to where the system will look for new service files
 - b. Second command checks for new service files
6. Enable the service file: `sudo systemctl enable BOREALIS_CAM.service`
7. To test if service file can run: `sudo systemctl start BOREALIS_CAM.service`

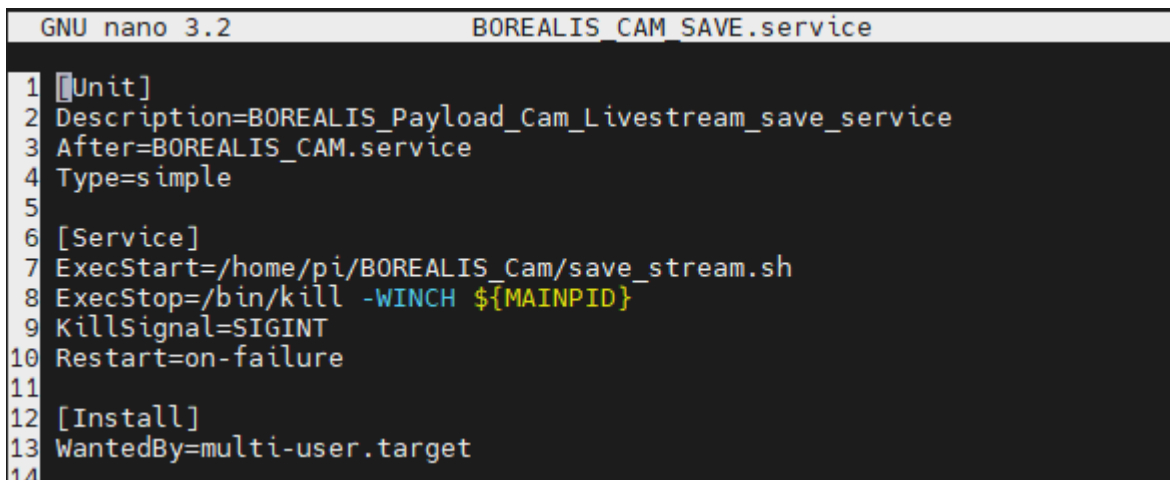
Do the similar thing to service file for saving the stream:

1. Return to Streaming folder: `cd ~/BOREALIS_Cam`
2. Create and edit service file for starting the stream: `nano BOREALIS_CAM_SAVE.service`
3. Type in:

```
[Unit]
Description=BOREALIS_Payload_Cam_Livestream_save_service
After=BOREALIS_CAM.service
Type=simple
```

```
[Service]
ExecStart=/home/pi/BOREALIS_Cam/save_stream.sh
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGINT
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

A screenshot of a terminal window showing the nano text editor. The title bar reads "GNU nano 3.2" and "BOREALIS_CAM_SAVE.service". The editor content is as follows:

```
1 [Unit]
2 Description=BOREALIS_Payload_Cam_Livestream_save_service
3 After=BOREALIS_CAM.service
4 Type=simple
5
6 [Service]
7 ExecStart=/home/pi/BOREALIS_Cam/save_stream.sh
8 ExecStop=/bin/kill -WINCH ${MAINPID}
9 KillSignal=SIGINT
10 Restart=on-failure
11
12 [Install]
13 WantedBy=multi-user.target
14
```

Figure 4 BOREALIS_CAM_SAVE.service file

4. Save files: Press Ctrl + X, press y and then press Enter
5. Install the service file: `sudo cp BOREALIS_CAM_SAVE.service /etc/systemd/system/ && sudo systemctl daemon-reload`
 - a. First command copies the service file to where the system will look for new service files
 - b. Second command checks for new service files
6. Enable the service file: `sudo systemctl enable BOREALIS_CAM_SAVE.service`
7. To test if service file can run: `sudo systemctl start BOREALIS_CAM_SAVE.service`

Almost there. We need to set the IP for where the raspberry pi should host. The camera system is done. All the system needs is packaging for flight.