How to “Reinstall” the Airflow Demo Machine from Scratch

By: Adam Nguyen

Preface

If you are reading this, then I can assume that something bad has happened. Not to worry, this guide will take you through setting the machine right back up.This is a bit of a lengthy process, but it will be mostly for the “downloading” and “recompiling” or “rebuilding”. I have streamlined the process as much as possible. And perhaps, you will not need to start from the very beginning. If you are new to the Linux operating system, then this guide will also serve as an excellent overview of why it is superior to Windows. Just kidding. (not really).

Step 1: Reflash the Jetson Nano

Required Items/Equipment:

- Linux computer with the Nvidia SDK Manager installed
  - https://docs.nvidia.com/sdk-manager/install-with-sdkm-jetson/index.html
- A USB-A to micro-USB cable

Online Guide from Seeedstudio

This step is relatively straightforward and an abundance of documentation is available. A very helpful step by step guide can be found here:

https://wiki.seeedstudio.com/install_NVIDIA_software_to_Jetson-10-1-A0/#flashing-to-emmc-with-sdk-manager

We will want to flash the software/OS to the EMMC.

Advice:

- For the SDK manager to recognize the Jetson device, it must be in *recovery mode*. To do this, use a jumper between the FORCED RECOVERY pin and the GND pin on the carrier board. There are labels/markings indicating the pins on the carrier board.
- After flashing the device, the jumper must be removed for it to boot normally.
- A LAN connection can make this process quicker

Step 2 (Optional, but recommended): Clean-up space on the Jetson

Congratulations for making it this far.

We will want the device to boot and run from the onboard EMMC because it will be much quicker than running from a USB connected drive. However, this will pose an issue with limited space. After flashing the Jetson, we will only have approximately 2GB of free space or so. In the next steps, we will use another trick to move some things that are normally on the root file system (the EMMC) to a USB drive, but cleaning up space on the EMMC of things we won't need will give us some "breathing space".

The following guide is helpful to cleaning up some space, however do not use all the steps. We will only want to follow steps 1 to 7 from this guide:

https://collabnix.com/easy-way-to-free-up-jetson-nano-sd-card-disk-space-by-40%EF%BF%BC%EF%BF%BC/

Again, do NOT do the last step.

Step 3: Mount the USB drive as a permanent mount and Reconfigure Docker

Preface

A portion of this application uses a Docker container and the Docker Image is approximately 4GB. This is more than the available space after flashing the Jetson device. Therefore, we will need to do a couple of tricks. Docker images are typically stored in the root file system, so we will also need to reconfigure and move Docker related data items from the root file system (EMMC) to the new USB drive.

Please pay careful attention to the next steps.

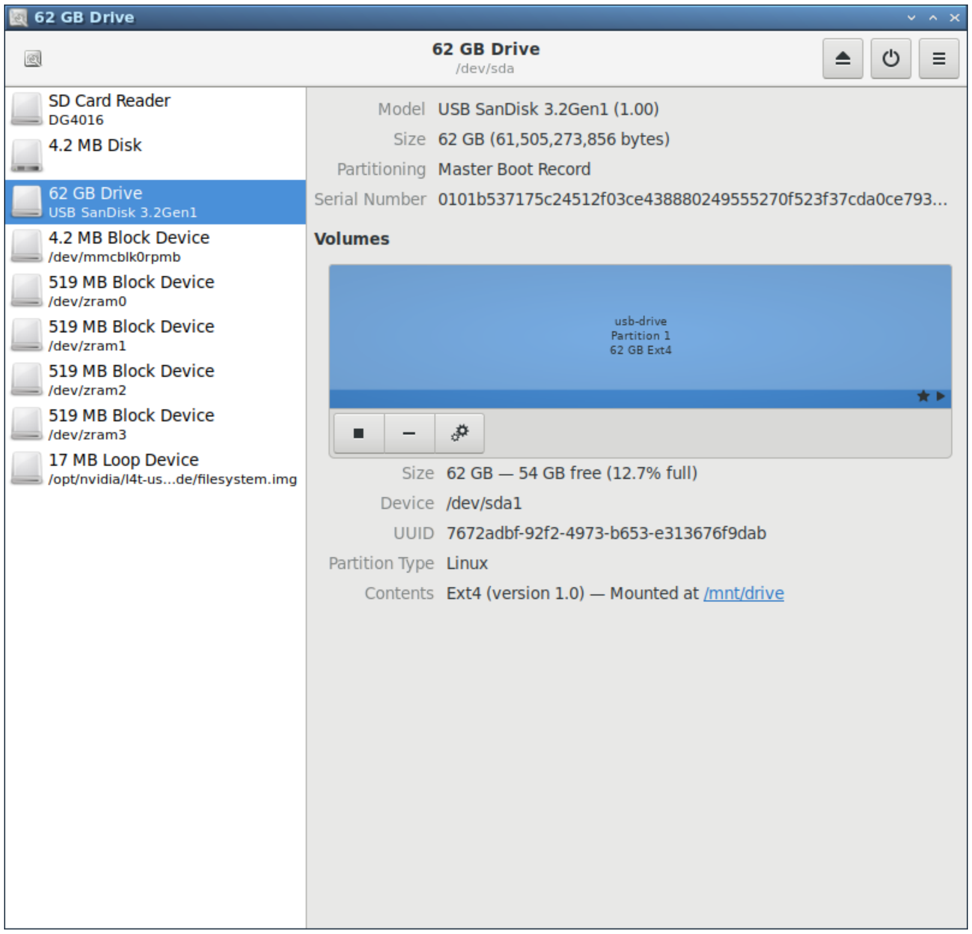
Step 3a: Reformat the USB-Drive

Plug the USB-drive to the Jetson device. It is highly advised that this USB-drive is of a 3.0 generation type, or speed issues may occur.

We will now reformat the USB drive to the correct type, the EXT4 format. There are many tools to do this, but we can do this with the native *gnome-disks* tool.

On the terminal, issue the command:

```
gnome-disks
```



Model	USB SanDisk 3.2Gen1 (1.00)
Size	62 GB (61,505,273,856 bytes)
Partitioning	Master Boot Record
Serial Number	0101b537175c24512f03ce438880249555270f523f37cda0ce793...

Size	62 GB — 54 GB free (12.7% full)
Device	/dev/sda1
UUID	7672adbf-92f2-4973-b653-e313676f9dab
Partition Type	Linux
Contents	Ext4 (version 1.0) — Mounted at /mnt/drive

Then navigate to the USB-drive of interest and reformat the drive to Ext4 format.

As well, we will need the UUID and the path of the device for later purposes. However, this information can be easily accessed again.

Step 3b:
Mount the USB-drive and add it to the fstab (file system table)

We will first need to create a directory for where the USB-drive will mount to. In the terminal, issue the command:

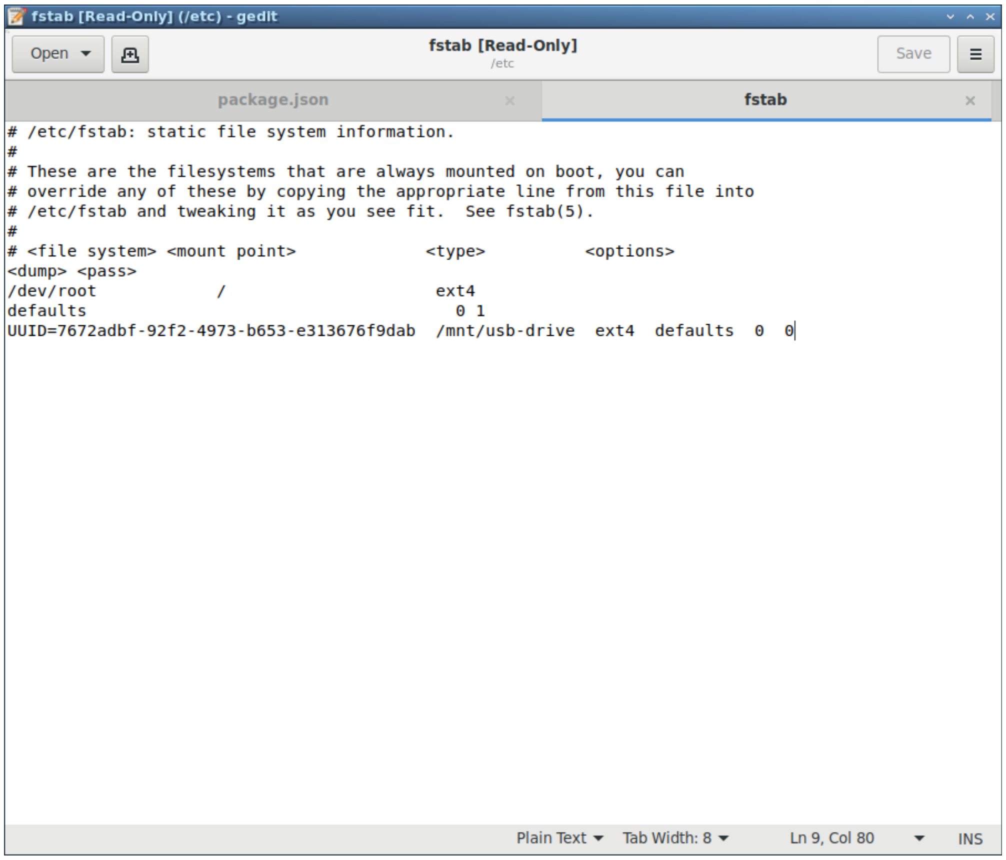
```
sudo mkdir -p /mnt/usb-drive
```

Now, we will add this drive to the file system table (fstab). We will require the UUID and device path from the previous step.

To open the fstab file with a text editor (e.g. with gedit), issue the terminal command:

```
sudo gedit /etc/fstab
```

Then add the USB-drive to the fstab as such:



```
# /etc/fstab: static file system information.
#
# These are the filesystems that are always mounted on boot, you can
# override any of these by copying the appropriate line from this file into
# /etc/fstab and tweaking it as you see fit.  See fstab(5).
#
# <file system> <mount point>             <type>          <options>
<dump> <pass>
/dev/root            /                      ext4
defaults                                      0 1
UUID=7672adbf-92f2-4973-b653-e313676f9dab  /mnt/usb-drive  ext4  defaults  0  0
```

Now reboot the Jetson device, and we should see that the USB drive will have been automatically mounted.

Step 3c: Reconfigure Docker

Fixing Permissions

It is possible that Docker has yet to be added to the sudo group. We will first correct this issue.

Run the command:

```
sudo groupadd docker
```

Then:

```
sudo usermod -aG docker $USER
```

Now logout and log back into Ubuntu. This will give the user the ability to run Docker without the use of “sudo”.

Moving Docker Data Directory

The next steps are based from this guide:

https://linuxconfig.org/how-to-move-docker-s-default-var-lib-docker-to-another-directory-on-ubuntu-debian-linux

First, we stop the Docker service:

```
sudo systemctl stop docker.service
sudo systemctl stop docker.socket
```

Next, we create the new location for the Docker data directory:

```
mkdir -p /mnt/usb-drive/docker
```

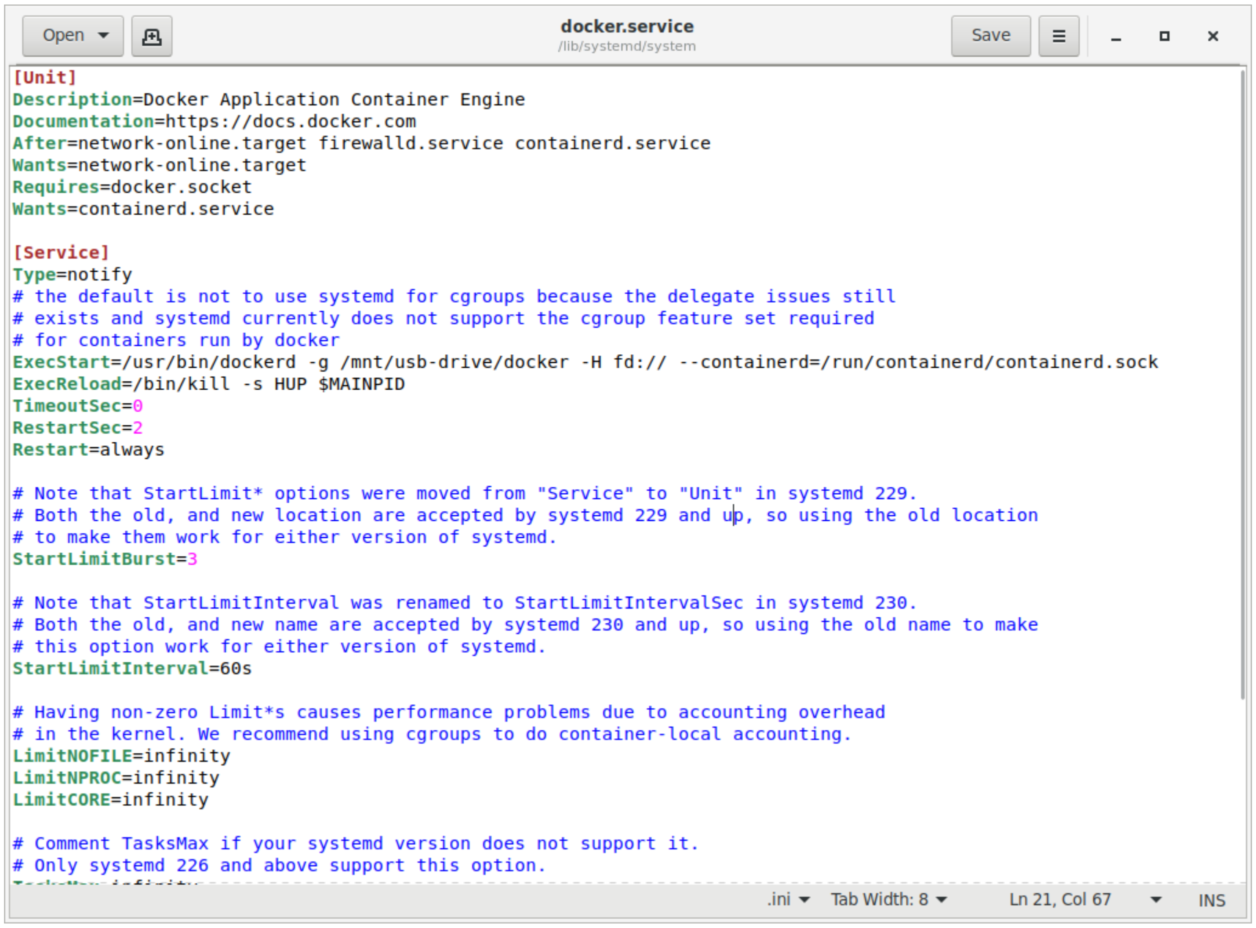
Next, we reconfigure the Docker systemd service. Using a text editor, open the service file:

```
sudo gedit /lib/systemd/system/docker.service
```

Edit line 14 so it looks like:

```
ExecStart=/usr/bin/dockerd -g /mnt/usb-drive/docker -H fd://
--containerd=/run/containerd
```

The text file should look something similar to:



```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket
Wants=containerd.service

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -g /mnt/usb-drive/docker -H fd:// --containerd=/run/containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not support it.
# Only systemd 226 and above support this option.
```

Now we copy the old location to the new location:

```
sudo rsync -aqxP /var/lib/docker/ /mnt/usb-drive/docker
```

Finally, we restart the systemctl daemon:

```
sudo systemctl daemon-reload
sudo systemctl start docker
```

And now we are done with this step.

Step 4: Clone the Repository

First, we will make a new directory to hold the repository:

```
mkdir /mnt/usb-drive/workspace
```

Next, change directory to the newly created directory:

```
cd /mnt/usb-drive/workspace
```

Then clone the application repository:

```
git clone https://github.com/adamsnguyen/airflow-control.git
```

Step 5: Build the Docker image for the ROS application.

We will change directories to where the Dockerfile is located in the repository:

```
cd /mnt/usb-drive/workspace/airflow-control/docker/ros/
```

Then, we run a script that builds the docker image:

```
./build.sh
```

This may take awhile…

Step 6: Install the MQTT broker/daemon

The application uses a MQTT broker that runs as a background process on the Jetson device. To install this broker, simply issue the commands:

```
sudo apt update && sudo apt install -y mosquitto
```

Step 7: Install Node

https://github.com/nodesource/distributions/blob/master/README.md

We will install the 16.xx LTS version:

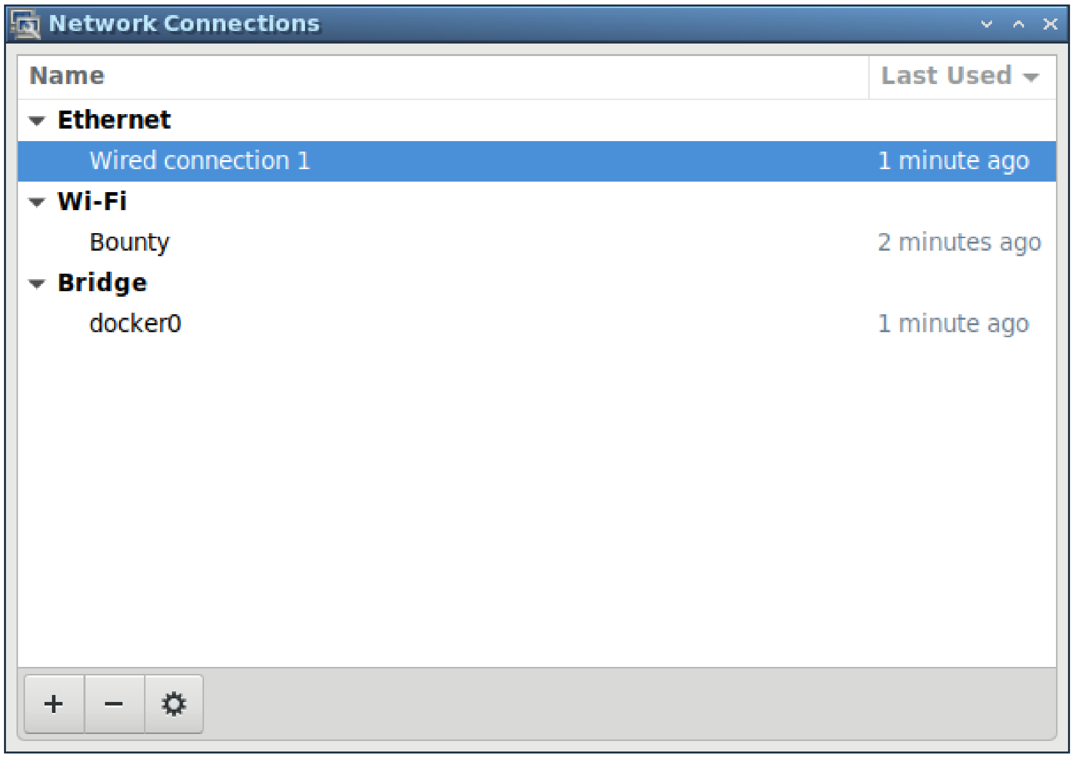
Issue the command:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash - &&\
sudo apt-get install -y nodejs
```

Step 8: Set a Static IP Address

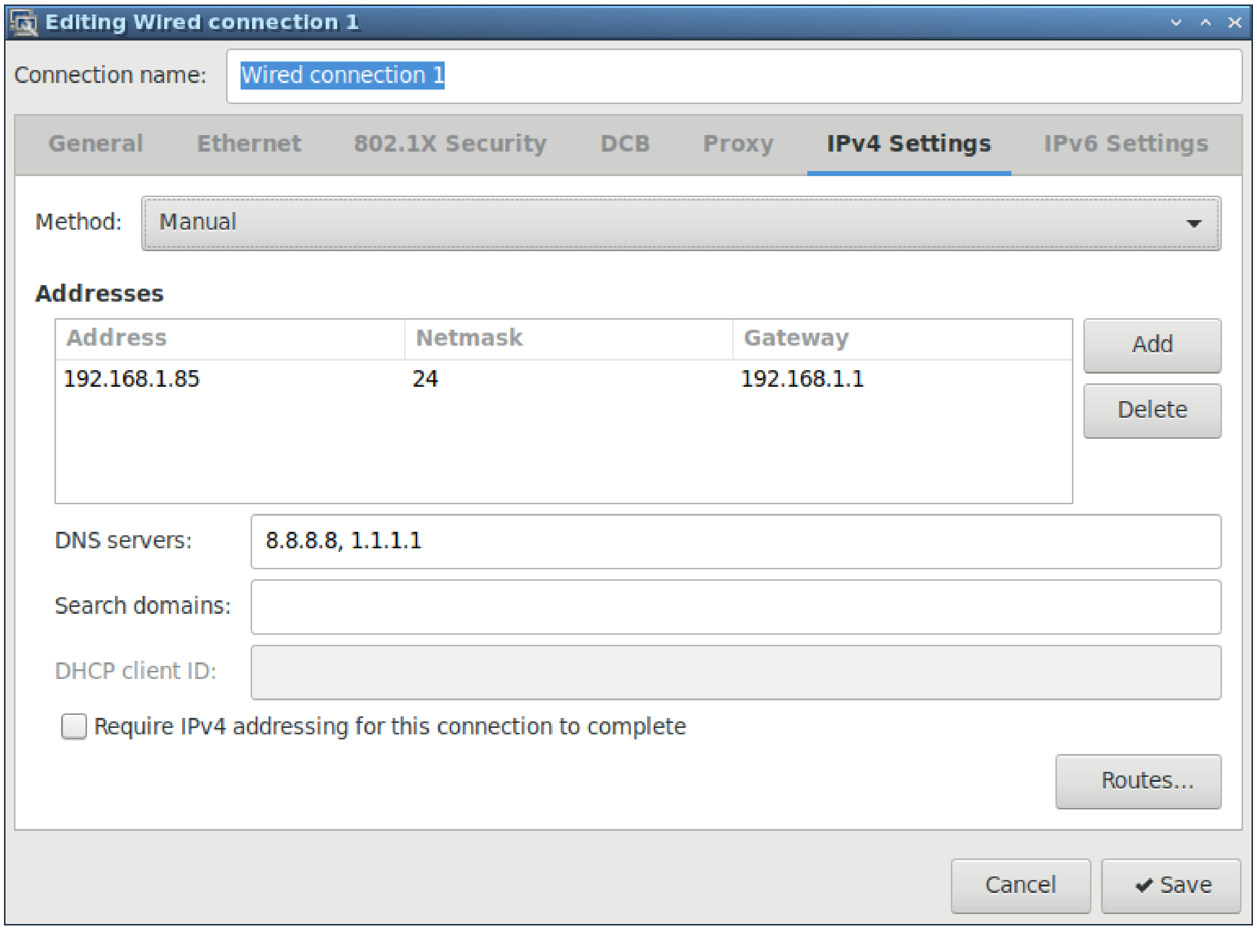
This step assumes that we are using the provided ASUS WL-520GU router. If a different router is used, the steps will be similar but some parameters may be different. Connect the Jetson device to the router with an ethernet cable.

Open Network Connections:



Name	Last Used
Ethernet	
Wired connection 1	1 minute ago
Wi-Fi	
Bounty	2 minutes ago
Bridge	
docker0	1 minute ago

Click onto the gear button on “Wired connection 1” and add the following information into IPv4 Settings. Click on Save:



Connection name: Wired connection 1

Method: Manual

Addresses

Address	Netmask	Gateway
192.168.1.85	24	192.168.1.1

DNS servers: 8.8.8.8, 1.1.1.1

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

Step 9: Create static paths for USB device with symlinks

A udev rule has already been created and is included in the repository. We will just need to copy this file to the appropriate location with the following command:

```
sudo cp /mnt/usb-drive/workspace/airflow-control/99-usb-serial.rules
/etc/udev/rules.d/99-usb-serial.rules
```

Step 10 (If necessary): Re-upload the Arduino firmware

First, follow the guide here to install the binary for Arduino IDE that is compatible with the Jetson Nano

https://jetsonhacks.com/2019/10/04/install-arduino-ide-on-jetson-dev-kit/

After having the Arduino IDE installed, open the application with the terminal using:

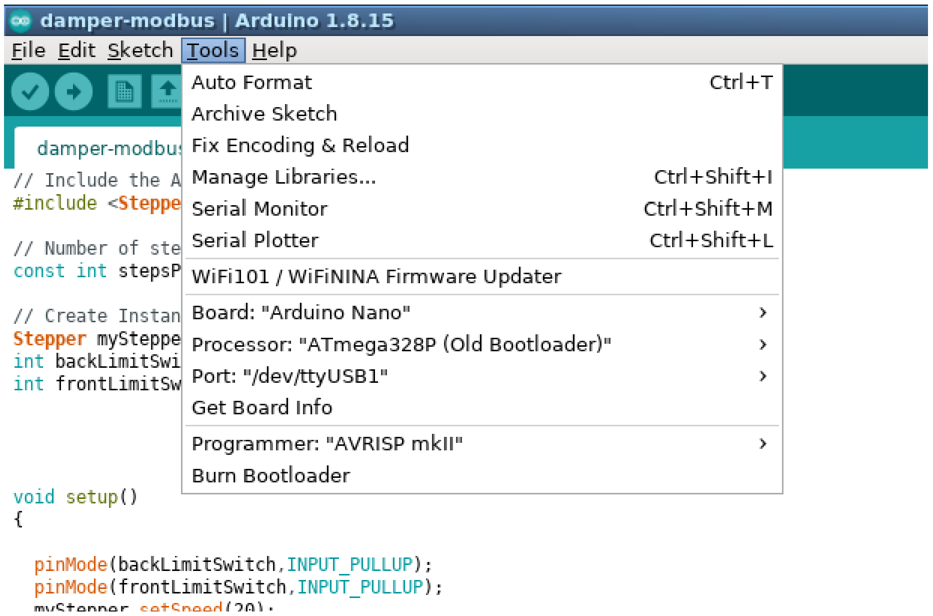
```
arduino
```

Then, open the damper module sketch within the application.

This is located at:

```
/mnt/usb-drive/workspace/airflow-control/mcu-firmware-damper-control/damper-modbus/damper-modbus.ino
```

Set the configurations as such:


damper-modbus | Arduino 1.8.15
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Burn Bootloader
damper-modbu

Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port: "/dev/ttyUSB1"
Programmer: "AVRISP mkII"

```
// Include the A
#include <Steppe

// Number of ste
const int stepsP

// Create Instan
Stepper mySteppe
int backLimitSwi
int frontLimitSw



void setup()
{

  pinMode(backLimitSwitch,INPUT_PULLUP);
  pinMode(frontLimitSwitch,INPUT_PULLUP);
```

Then upload the firmware to the arduino.

Step 11: Install and Configure PM2

PM2 is a very handy production manager for node applications. This service will make sure the web servers are started on boot and that they stay running.

Further information/documentation can be found here:

https://pm2.keymetrics.io/

First, install PM2:

```
sudo npm install -g pm2
```

In order to allow PM2 to start our web applications on boot, we will need to provide appropriate permissions.

Run the following command:

```
pm2 startup
```

This command will output a script that you will issue in the terminal. It will look similar (but not) to this:

```
sudo su -c "env PATH=$PATH:/home/unitech/.nvm/versions/node/v14.3/bin pm2
startup <distribution> -u <user> --hp <home-path>
```

Issue that command from the output to configure PM2.

Step 12: Install and Build the Web Applications

Build and Run the Front-End Server

In the terminal, go to the root folder of the front-end application:

```
cd /mnt/usb-drive/workspace/airflow-control/frontend-server/
```

Install the application:

```
npm install
```

Build the production version:

```
npm run build
```

Add to the PM2 startup/management list:

```
pm2 start "npm run prod" --name front-end
```

Now, the front-end is now available to access!

In the browser (of any computer connected to the local network, or on the Jetson device itself), the application should now be able to be accessed at:

`http://192.168.1.85:3001/`

Save this to the PM2 startup list:

```
pm2 save
```

Note: Make sure the connection to the wired local network is turned on for you to access the application.

Tip: We can troubleshoot and see console output from the web servers with the command

```
pm2 log
```

Build and Run the MQTT “Back-end” Server

In the terminal, go to the root folder of the front-end application:

```
cd /mnt/usb-drive/workspace/airflow-control/mqtt-websocket-interface/
```

Install the application:

```
npm install
```

Add to the PM2 startup/management list:

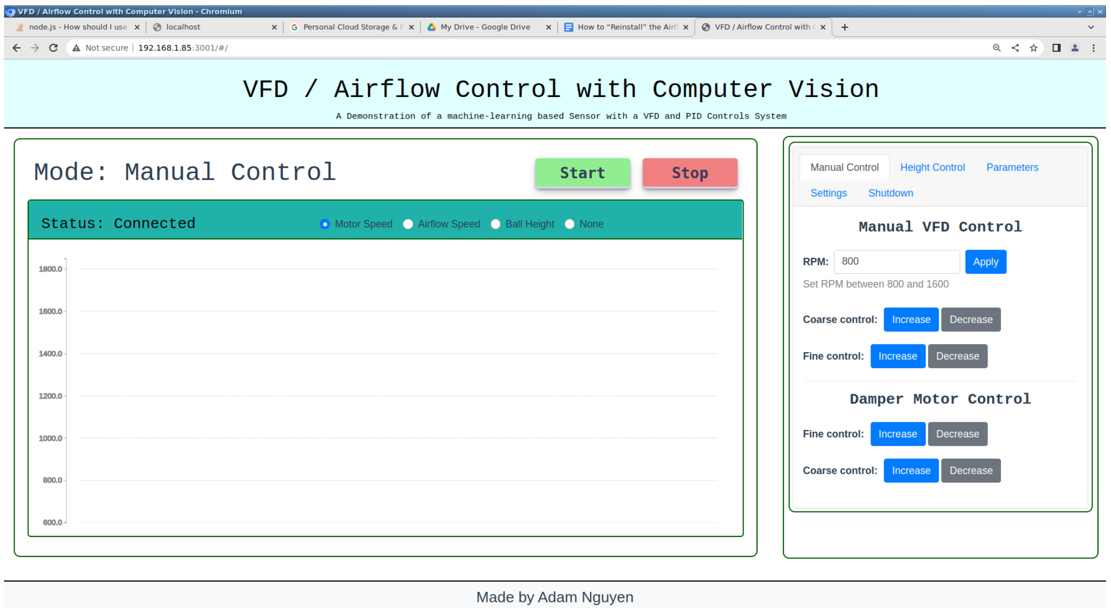
```
pm2 start "npm start" --name back-end
```

Save this to the PM2 startup list:

```
pm2 save
```

Now when you refresh the application at http://192.168.1.85:3001/

The application should now say it is connected. You can try restarting the device to see if the web application starts on boot.


VFD / Airflow Control with Computer Vision - Chromium
node.js - How should I use
localhost
Personal Cloud Storage & F
My Drive - Google Drive
How to “Reinstall” the Airfl
VFD / Airflow Control with C
Not secure
192.168.1.85:3001/#/

VFD / Airflow Control with Computer Vision

A Demonstration of a machine-learning based Sensor with a VFD and PID Controls System

Mode: Manual Control

Status: Connected

● Motor Speed ○ Airflow Speed ○ Ball Height ○ None

Manual VFD Control

RPM: 800 Apply

Set RPM between 800 and 1600

Coarse control: Increase Decrease

Fine control: Increase Decrease

Damper Motor Control

Fine control: Increase Decrease

Coarse control: Increase Decrease

Made by Adam Nguyen

Step 13: Configure Shutdown/Reboot Permissions

The application includes the “shutdown computer” and “reboot” button which typically requires root privileges and thus a password. To make this button functional under the PM2 manager, we will need to remove the password requirement for shutting down the computer.

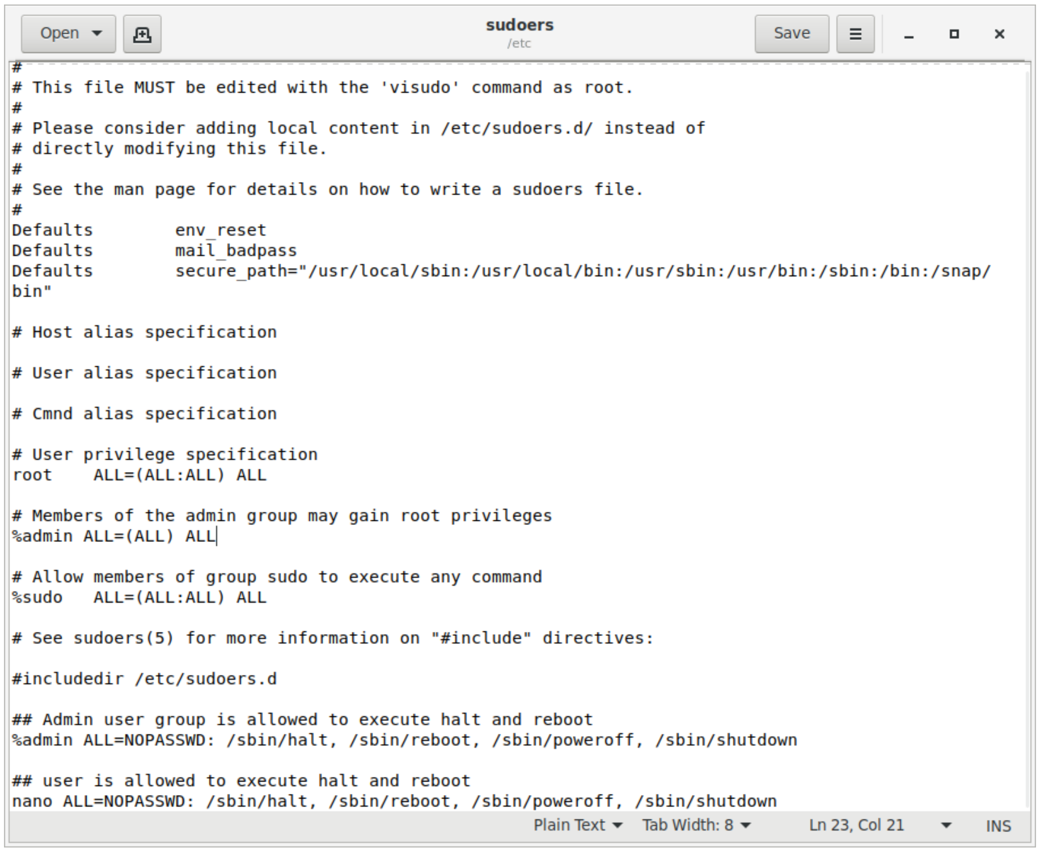
Open up the sudoers file and add the following lines at the bottom:

```
sudo gedit /etc/sudoers
```

```
## Admin user group is allowed to execute halt and reboot
%admin ALL=NOPASSWD: /sbin/halt, /sbin/reboot, /sbin/poweroff,
/sbin/shutdown

## user is allowed to execute halt and reboot
nano ALL=NOPASSWD: /sbin/halt, /sbin/reboot, /sbin/poweroff, /sbin/shutdown
```

Note that ‘nano’ should be the default user (if not nano). The file should look similar to this:



```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/
bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

## Admin user group is allowed to execute halt and reboot
%admin ALL=NOPASSWD: /sbin/halt, /sbin/reboot, /sbin/poweroff, /sbin/shutdown

## user is allowed to execute halt and reboot
nano ALL=NOPASSWD: /sbin/halt, /sbin/reboot, /sbin/poweroff, /sbin/shutdown
```

Step 14: Configure to run ROS application on boot

We will use the Cron job scheduler to manage the ROS application at boot.

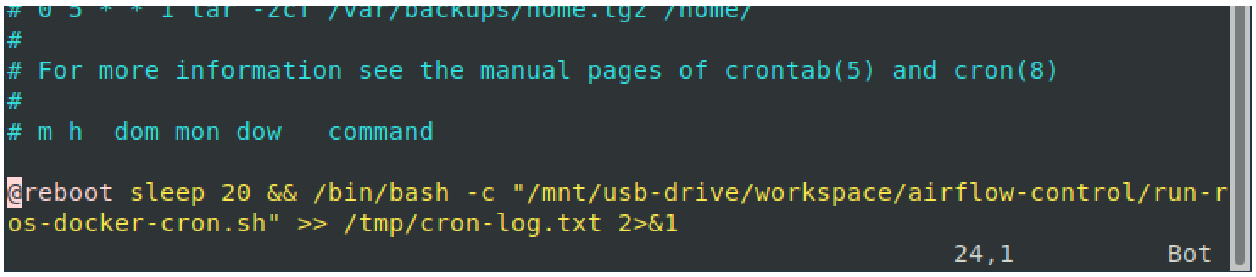
Open up the Cron table to add an entry:

```
sudo crontab -e
```

Add this line at the bottom (we are going to run a script that starts the Docker image with the appropriate parameters):

```
@reboot sleep 20 && /bin/bash -c
"/mnt/usb-drive/workspace/airflow-control/run-ros-docker-cron.sh" >>
/tmp/cron-log.txt 2>&1
```

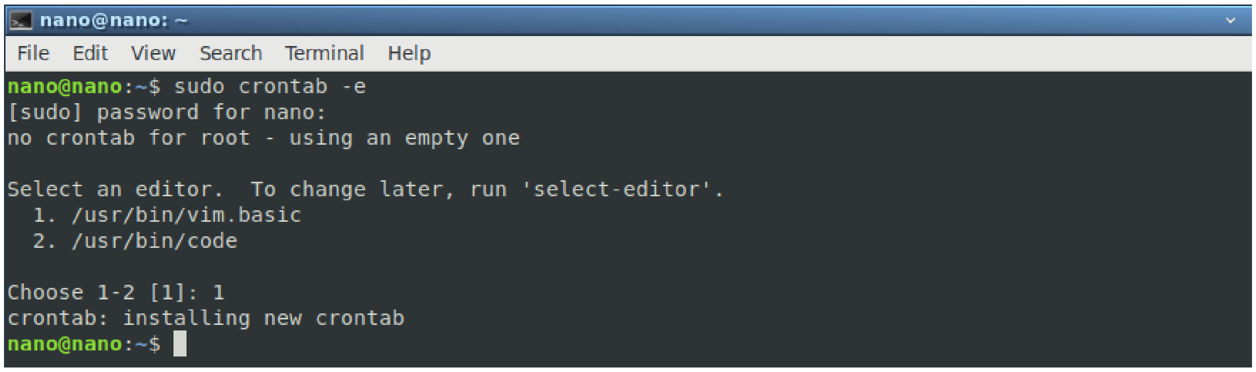
The file should look something similar to:



```
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

@reboot sleep 20 && /bin/bash -c "/mnt/usb-drive/workspace/airflow-control/run-r
os-docker-cron.sh" >> /tmp/cron-log.txt 2>&1
                                                              24,1          Bot
```

Exit and ensure that this file is saved.



```
nano@nano:~$ sudo crontab -e
[sudo] password for nano:
no crontab for root - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /usr/bin/vim.basic
  2. /usr/bin/code

Choose 1-2 [1]: 1
crontab: installing new crontab
nano@nano:~$
```

Now, we can reboot to see if it takes effect. We can troubleshoot with the command:

```
sudo systemctl status cron
```

Or by checking the log that we had set at /tmp/cron-log.txt

Make sure that the script passes the correct mounting location for Docker to use:

That is, make sure this line is correct in the script that the cron scheduler is running:

```
 --mount
type=bind,source=/mnt/usb-drive/workspace/airflow-control/ros-workspace,tar
get=/workspace \
```

In particular, the *source.*

Finish!

Congratulations if you've made it this far and if the system is back and running. You've done good work and you deserve a beer (if you are of age) or a slice of cake or some self-affirmations.

Good luck.

-Adam Nguyen

PS: If something isn't working and Google / Stack isn't providing enough hints, feel free to send me an email at avnguyen213@gmail.com