Journal #03

Monday 25th: Installing the Webserver

1st try: As mentioned before, our plan was to install nginx as a reverse proxy and then PiGallery2, to act as a middleman to handle requests. The first option that I tried was a direct install, meaning that I would have to download Node.js to run PiGallery2 natively. Everything was working well:

- Install Node.js
- Install PiGallery2 from source: from GitHub
- Edit the configuration

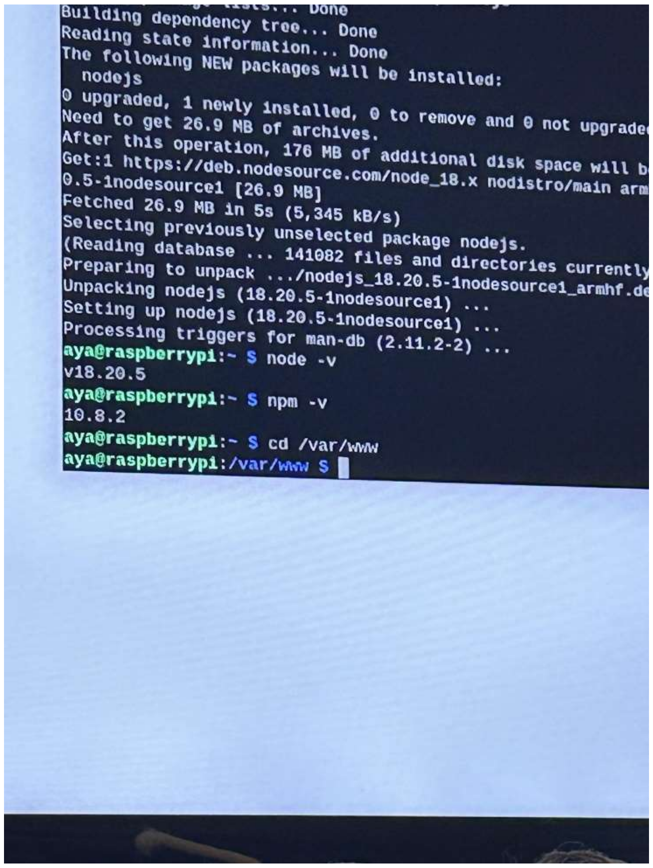
In this process, I got confused so I used my best friend, chat GPT. It mentioned that I needed to retrieve a file named index.ts and I had it. But when I tried to continue, it was telling me that the index was not found, so I believe that I did not install the dependencies properly, which made me have this error. I could not go further from here even after researching about it, so I gave up this method.

2nd try: The second option, which is the recommended one since it's way easier to install, was to install PiGallery and run it in a docker. These are the steps that you can also find in the GitHub repository:

- curl -sSL https://get.docker.com | sh
- sudo apt-get install -y python3-pip sudo pip3 install docker-compose
- sudo apt-get remove python-configparser
- sudo pip install docker-compose
- mkdir ~/pigallery2
- cd ~/pigallery2
- Create a dicker-compose.yml file INSIDE the pigallery2 directory
- docker-compose up -d
- Configure nginx as reverse proxy: sudo nano /etc/nginx/sites-available/pigallery2

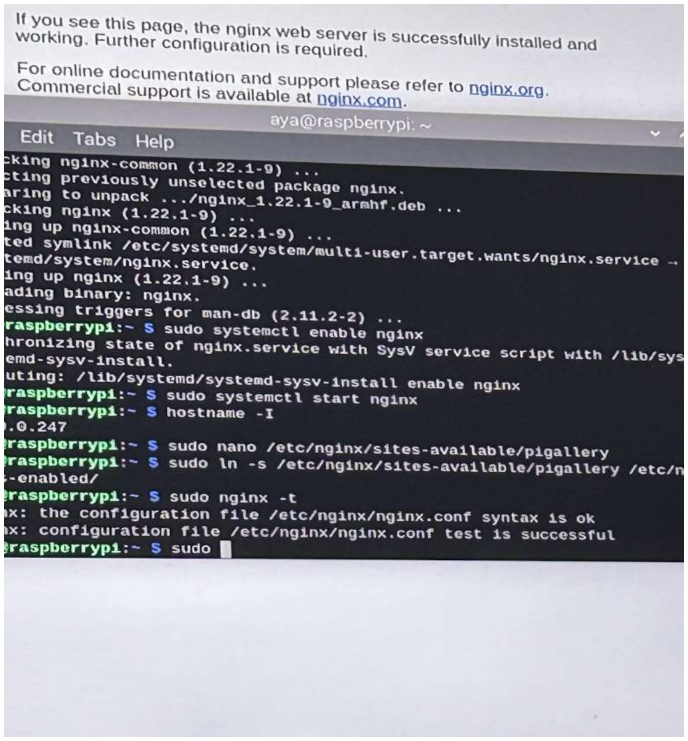
- Create a symbolic link: enable the configuration: sudo ln -s /etc/nginx/sites-available/pigallery2 /etc/nginx/sites-enabled/
- sudo nginx -t
- sudo systemctl restart nginx
- Access the web server on the web: http://10.0.0.247 (the IP of our Raspberry Pi)

This worked wonderfully and it was so fast to install. You also need to install nginx before making it into a reverse proxy, but we had it already installed so I skipped that step. The downside of using this option is that our server depends on the person who created the docker and the PiGallery repo. If later on in the future they decide to remove it, our server will also be removed. This dependence does not occur with the first option since it is installed natively.



```
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 0 not upgrade
Need to get 26.9 MB of archives.
After this operation, 176 MB of additional disk space will b
Get:1 https://deb.nodesource.com/node_18.x nodistro/main arm
0.5-1nodesource1 [26.9 MB]
Fetched 26.9 MB in 5s (5,345 kB/s)
Selecting previously unselected package nodejs.
(Reading database ... 141082 files and directories currently
Preparing to unpack .../nodejs_18.20.5-1nodesource1_armhf.de
Unpacking nodejs (18.20.5-1nodesource1) ...
Setting up nodejs (18.20.5-1nodesource1) ...
Processing triggers for man-db (2.11.2-2) ...
aya@raspberrypi:~ $ node -v
v18.20.5
aya@raspberrypi:~ $ npm -v
10.8.2
aya@raspberrypi:~ $ cd /var/www
aya@raspberrypi:/var/www $
```

Trying the first option

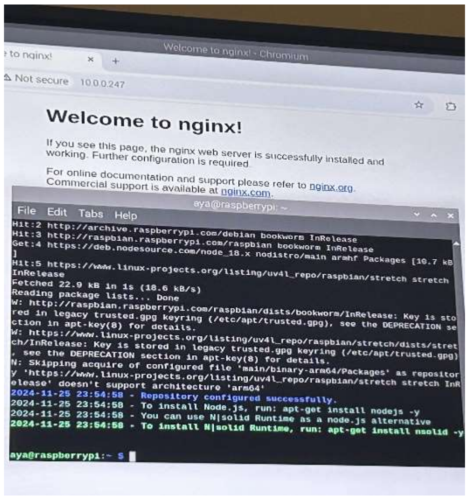


If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

```
cking nginx-common (1.22.1-9) ...
cting previously unselected package nginx.
aring to unpack .../nginx_1.22.1-9_armhf.deb ...
cking nginx (1.22.1-9) ...
ing up nginx-common (1.22.1-9) ...
ted symlink /etc/systemd/system/multi-user.target.wants/nginx.service →
temd/system/nginx.service.
ing up nginx (1.22.1-9) ...
ading binary: nginx.
essing triggers for man-db (2.11.2-2) ...
raspberrypi:~ $ sudo systemctl enable nginx
hronizing state of nginx.service with SysV service script with /lib/sys
emd-sysv-install.
uting: /lib/systemd/systemd-sysv-install enable nginx
raspberrypi:~ $ sudo systemctl start nginx
raspberrypi:~ $ hostname -I
.0.247
raspberrypi:~ $ sudo nano /etc/nginx/sites-available/pigallery
raspberrypi:~ $ sudo ln -s /etc/nginx/sites-available/pigallery /etc/n
-enabled/
raspberrypi:~ $ sudo nginx -t
x: the configuration file /etc/nginx/nginx.conf syntax is ok
x: configuration file /etc/nginx/nginx.conf test is successful
raspberrypi:~ $ sudo
```

Testing if nginx



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

```
Hit:2 http://archive.raspberrypi.com/debian bookworm InRelease
Hit:3 http://raspbian.raspberrypi.com/raspbian bookworm InRelease
Get:4 https://deb.nodesource.com/node_18.x nodistro/main armhf Packages [10.7 kB
]
Hit:5 https://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch
InRelease
Fetched 22.9 kB in 1s (18.6 kB/s)
Reading package lists... Done
W: http://raspbian.raspberrypi.com/raspbian/dists/bookworm/InRelease: Key is sto
red in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION se
ction in apt-key(8) for details.
W: https://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch/dists/stret
ch/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg)
, see the DEPRECATION section in apt-key(8) for details.
N: Skipping acquire of configured file 'main/binary-arm64/Packages' as repositor
y 'https://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch InR
elease' doesn't support architecture 'arm64'
2024-11-25 23:54:58 - Repository configured successfully.
2024-11-25 23:54:58 - To install Node.js, run: apt-get install nodejs -y
2024-11-25 23:54:58 - You can use N|solid Runtime as a node.js alternative
2024-11-25 23:54:58 - To install N|solid Runtime, run: apt-get install nsolid -y

aya@raspberrypi:~ $
```

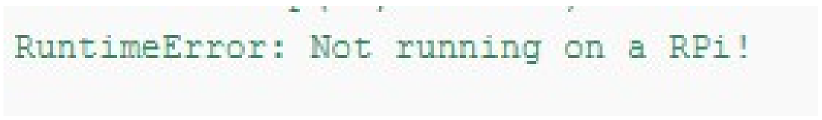
Repository configured successfully

Adding more buttons:

Previously, we installed 1 button which lets us click the button to take pictures. The second button will need to record until the user wants to stop the recording. We thought of adding a third button that would turn off and on the Raspberry Pi when we clicked on it, but we realized that the Raspberry can't turn on once it shuts down. We need to manually plug it back on to turn it on. To make it happen we would need another component: a power button switch. We decided to give up on this idea since we can't afford to wait for the component to arrive (order it). Another solution was to put the Raspberry Pi into sleeping/hibernating mode, but when I searched it up, it mentioned that the Raspberry Pi does not support this feature so there I came across one last solution (doesn't work), which was to shut it down softly (sudo shutdown -h now). I learned that this still shuts down the Raspberry Pi completely, but with more caution than by doing: sudo shutdown now.

Encountering the biggest problem:

When I was experiencing the 6pin switch button, I used a resistor and ran the Python code, which contained the code to record when the user clicked it and stopped recording when they clicked it back. It wasn't working so I switched back to the 4-pin button, which I've gotten familiar with. When I ran another code using this button, it gave me an error that did not make any sense:



```
RuntimeError: Not running on a RPi!
```

I searched for the cause of the error and it was either the Raspberry itself that wasn't working or it was the read-write permissions for accessing the GPIO. I found this article that explained it really well and it even gave a step-by-step guide to fix this issue but mine didn't work. I'm guessing it has something to do with me not enabling the SSH but during this time I had no clue why it was causing this problem:

- https://zoveloper.medium.com/runtimeerror-not-running-on-a-rpi-c6a4930c9c84

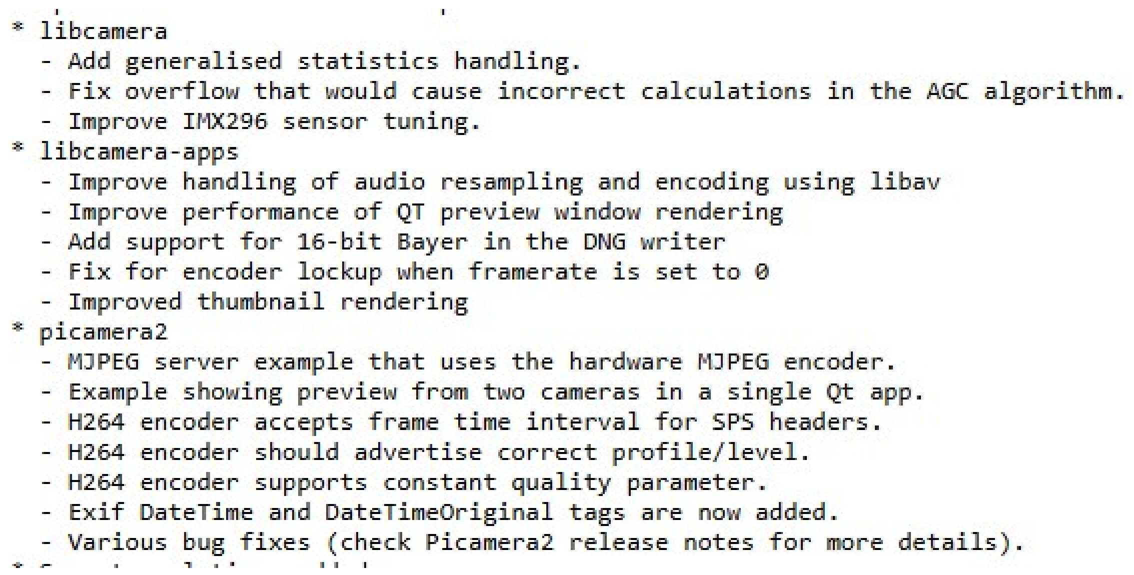
Conclusion:

I had an SD card that could recognize the camera module, but now, it couldn't recognize the Raspberry P, so it has become useless AND I have another SD card that can't recognize the camera but can recognize the Raspberry Pi/GPIO. So this situation became a serious problem for our project

Friday, November 29: Fixing the Issue

Re-Flashing the Raspberry Pi:

On Tuesday, we went to the STEM center in Vanier College, to see if anyone from the Computer Engineering department or robotics club knew anything about the problem. A student told us that Angelos (I hope I'm not butchering his name) has a good basis for the Raspberry Pi. He told us that we needed to reflash the RPI to fix the issue. So, instead of re-flashing the SD from our colleague, we re-flashed the SD of Aya. When picking what version to install, we saw that she had installed the 32-bit OS, but there was another one with the name, Legacy 32-bit. The issue on the SD that could not recognize the camera was because it did not contain the picamera2 library, thus it did not detect the module. So we finally got the SD to work and we finished the code for the 2 buttons: 1- Take picture, 2- Record.



```
* libcamera
  - Add generalised statistics handling.
  - Fix overflow that would cause incorrect calculations in the AGC algorithm.
  - Improve IMX296 sensor tuning.
* libcamera-apps
  - Improve handling of audio resampling and encoding using libav
  - Improve performance of QT preview window rendering
  - Add support for 16-bit Bayer in the DNG writer
  - Fix for encoder lockup when framerate is set to 0
  - Improved thumbnail rendering
* picamera2
  - MJPEG server example that uses the hardware MJPEG encoder.
  - Example showing preview from two cameras in a single Qt app.
  - H264 encoder accepts frame time interval for SPS headers.
  - H264 encoder should advertise correct profile/level.
  - H264 encoder supports constant quality parameter.
  - Exif DateTime and DateTimeOriginal tags are now added.
  - Various bug fixes (check Picamera2 release notes for more details).
```

legacy 32-bit Raspberry Pi

https://www.raspberrypi.com/software/operating-systems/

Steps to flash Raspberry Pi:

1. Flash: Go to the raspberry Pi Official website
2. Go to the Raspberry Pi Imager page
3. Pick your computer OS (Windows, MacOS, Ubuntu and even Terminal)
4. Download
5. Open the Downloaded file and accept all the terms

6. Raspberry pi Imager should now be downloaded and available to use
7. Click on it, and accept that it should make modifications on your computer
8. Insert an SD card using an adapter or by directly inserting it to your computer
9. First: Pick your raspberry pi device
10. 10- Second: Pick your OS system (64-bit, 32-bit, legacy 32-bit, etc…). We took the Legacy 32-bit since it comes with the picamera Library
11. Thirdly: pick your SD card storage
12. Flash

Final Code:

```
from gpiozero import Button

from picamera import PiCamera

from signal import pause

from datetime import datetime

import time


button = Button(2)

deux = Button(3)

def get_timestamp():

   return datetime.now().strftime("%Y-%m-%d_%H-%M-%S")


#Picture

def take_picture():

    timestamp = get_timestamp()

   file_path = f"/home/aya/Desktop/img_{timestamp}.jpg"

   camera.capture(file_path)

   print("took picture")
```

```
deux.when_pressed = take_picture


#Record
recording = False
clicks = 0
last_click_time = 0


def toggle_recording():
   global recording
   recording = not recording
   if recording:
      timestamp = get_timestamp()
      video_file = f"/home/aya/Desktop/video_{timestamp}.h264"
      print(f"Recording: {video_file}")
      camera.start_recording(video_file)
   else:
      print("Not recording")
      camera.stop_recording()


def button_clicked():
   global clicks, last_click_time
   current_time = time.time()


   if current_time - last_click_time <= 0.5:
      clicks += 1
```

```
    else:
        clicks = 1

    last_click_time = current_time

    if clicks == 2:
        toggle_recording()
        clicks = 0

button.when_pressed = button_clicked
print("Ready for double click")
pause()
```

****Take pictures using 1 click, Record by clicking 2 times (We have 2 buttons). Clicks are counted to check when it should record and when it should stop recording. Adding a time stamp to the file name so there are no problems that will resurface. Adding printing statements for debugging****

Webserver:

I also realised that we need to start our docker when we turn back on the Raspberry Pi, so we can create a systemd which can start the docker (include the time stamp, so it can always have a different name) when the RPi boots. As I mentioned above, having a running the PiGallery in a docker can have it's downsides since it depends on the person that provides it. If they shut it down, our server will also shut down, so our gallery heavily relies on them.

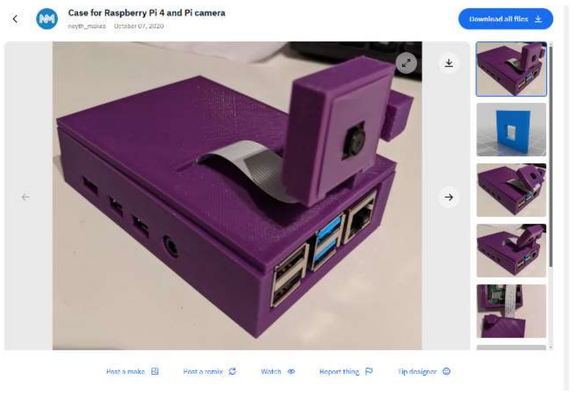
Choices:

- Keep the docker and create a systemd to start it at boot time
- Restart the webserver in the new SD card and install PiGallery2 natively

Raspberry Pi Case:

Aya will be printing the case for the Raspberry Pi and the camera module using the model that we found online. The description says that it is compatible with the Raspberry Pi 4 B model and the camera module, which is what we have:

https://www.thingiverse.com/thing:4617605



Conclusion:

What’s left:

-Finish the Webserver

-Publish pictures to the webserver

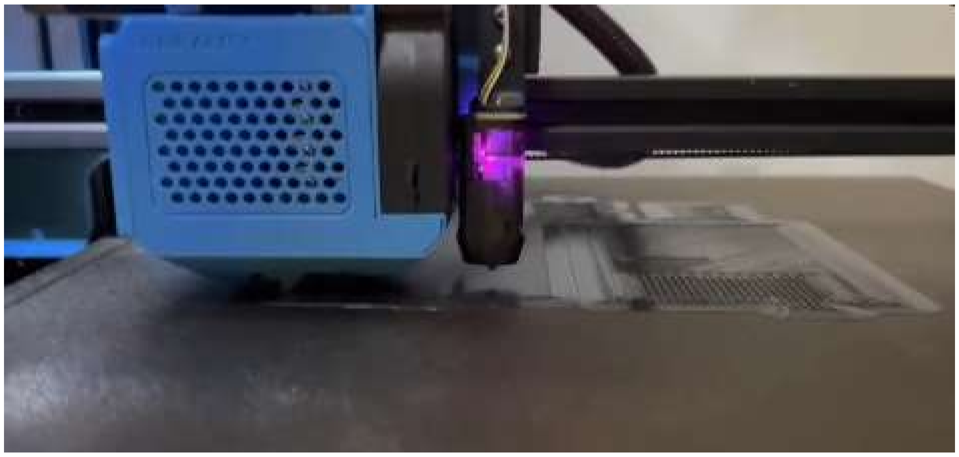
-install the case

-Create a firewall

-Create SSH keys

-Create systemd processes

Saturday, November 30: Final results of the case



Beginning of the case - Took 24hrs!

Finished product

Sunday, December 1st and Monday, December 2nd : Finishing the project

Reality:

Since we have a project to give for tomorrow (Application Development), I did not start the last touch-ups of this project and I will most likely finish at 2 am so I won't be able to write what I have done (Journal due before midnight))but these are the plans of what I should do and finish if everything goes well:

1. Install Nginx as reverse proxy and PiGallery2 natively
2. Verify if PiGallery publishes the pictures
3. Finish the scripts
4. Install a firewall
5. Create ssh keys

6. Finish creating systemd processes

If everything goes well, we will finish today/tomorrow and we will only need to finish the presentation and the PowerPoint slides.

GitHub repo: https://github.com/ayakharchafi/Unix-Project24.git

Difference from the project proposal:

We were very ambitious when starting this project and I've realized that we could have finished everything in advance if the Raspberry had detected the camera. We did not calculate these inconveniences in our schedule which we should had from the start. To start, we decided to not put the camera in a harness since it became really bulky: breadboard, connectors, camera module and buttons. We also decided to not make it portable because of the time limit that we had and the hardware requirements. This project cost us way more than expected. We removed the idea of having a cooling system: fans and heatsink. This would make the raspberry heavier and would cost more. We also read that for regular use and for low settings, the Raspberry 4 B should be fine without a cooling system. It would even be too extra if we added one. We also planned on having 3 buttons but we found out that we can't turn on a Raspberry Pi once it has shut down. We considered adding a third button to only shut down but that wouldn't be something that we really wanted. Another idea that we had and we were almost going to buy was a screen for the Raspberry Pi. So instead of using a screen, a keyboard and a mouse to navigate through the controls, we could have a touch screen connected to the RPI. We faced so many problems that we had to drop a lot of details that we wanted to implement, to make the camera more realistic. However, this is just a prototype so it can't be perfect. The aspects that we kept were the reverse proxy using nginx and PiGallery2 as the gallery organizer/ user interface. We have 2 buttons, one that is to take pictures and the other is to record. Thanks to my colleague's connections, we obtained a lot of hardware that we did not need to pay for and that we could use for the project, so I'm really thankful for that and of course, her hard work and ambition. In conclusion, the biggest changes that we made for our project were the camera portability and the use of straps, to be able to wear it, like a go-pro.