Weekly Journal #02

Week Overview (Thursday 21 - Sunday 24)

Planned Tasks:

- **Resolve the problem of the undetected camera**
- **Enable the tactile buttons to take pictures, and videos and power on/off the Raspberry Pi**
- **Starting the web server**

Thursday, November 21

Due to limited progress earlier, as we lacked a screen to connect the Raspberry Pi, we decided to investigate potential causes for our issue with the camera module not being recognized.

Actions Taken:

1. **Switched SD Cards**: Initially, we used the SD card from a Raspberry Pi keyboard. Later, we switched to a larger capacity card. Since then, the camera stopped being recognized. We researched whether using a new SD card could cause this issue. Although we followed the same setup steps as before, this seemed to be the closest explanation.
2. **SD Card Damage Test**: We tested the new SD card for damage, but it was functional.
3. **Testing with `libcamera-hello`**: This command provides a 5-second camera preview on the screen but requires enabling the camera first, which was part of the issue.

Findings:

Despite our searches, we couldn’t find the exact issue:

- **Misaligned Ribbon Cable**: Some users resolved similar problems by correctly orienting the ribbon cable.
- **Configuration Errors**: Others mentioned incorrect settings or missing files but used different setups, making the solutions less applicable.

Conclusion:

Since switching SD cards didn't resolve the problem, and troubleshooting methods provided limited insights, we decided to revert to the original SD card.

Friday, November 22

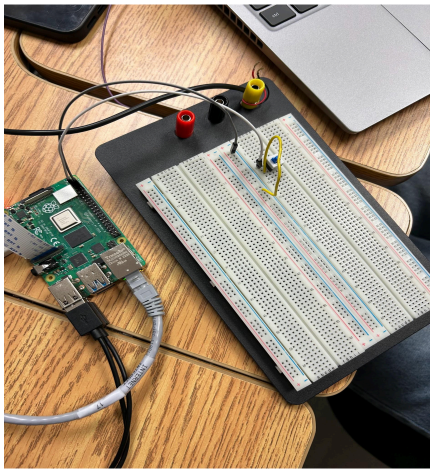
Enabling the Push Button to Capture Images

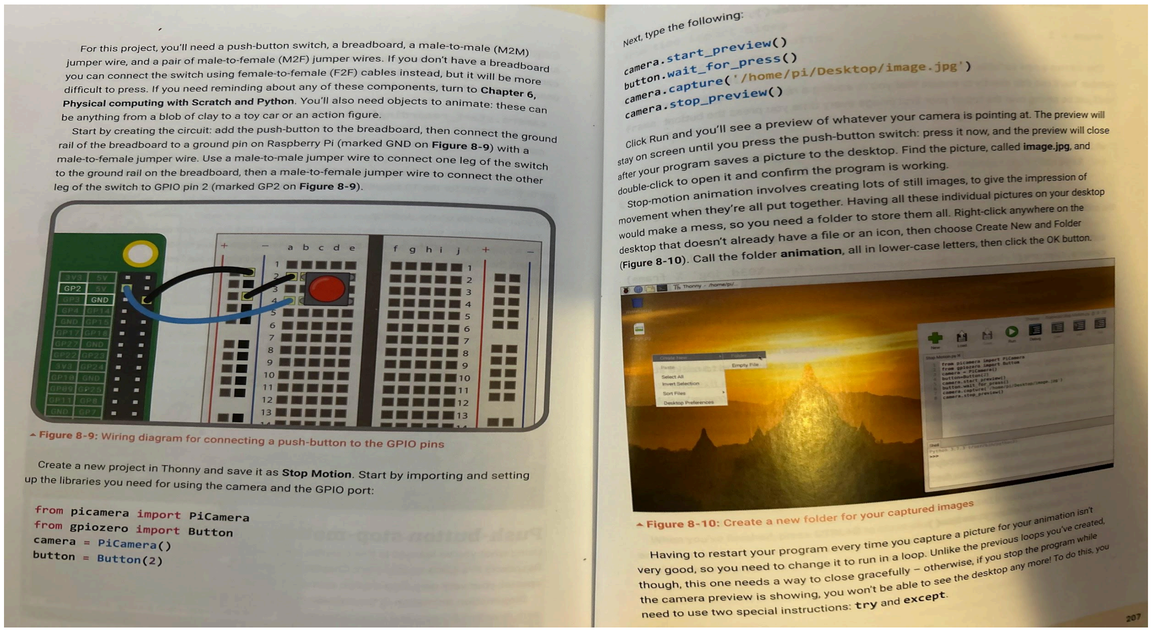
On this day, we focused on enabling the push button to capture images and save them using the camera module and the Raspberry Pi. The process began by reworking the circuit between the Raspberry Pi GPIO pins and the breadboard. We used two female-to-male wires, a male-to-male wire, and a push button to build the circuit, following the Beginner's Guide. The goal was to construct a functional setup and execute a Python script in Thonny (a Phyton IDE) to capture an image when the button was pressed.

The Circuit Setup

- Add the push-button to the breadboard.
- Connect the ground rail of the breadboard to a ground pin on Raspberry Pi with a male-to-female jumper wire.
- Use a male-to-male jumper wire to connect one leg of the switch to the ground rail on the breadboard, then a male-to-female jumper wire to connect the other leg of the switch to GPIO pin 2.

The Image of The Circuit and the steps of the guide:





For this project, you'll need a push-button switch, a breadboard, a male-to-male (M2M) jumper wire, and a pair of male-to-female (M2F) jumper wires. If you don't have a breadboard you can connect the switch using female-to-female (F2F) cables instead, but it will be more difficult to press. If you need reminding about any of these components, turn to **Chapter 6, Physical computing with Scratch and Python**. You'll also need objects to animate: these can be anything from a blob of clay to a toy car or an action figure.

Start by creating the circuit: add the push-button to the breadboard, then connect the ground rail of the breadboard to a ground pin on Raspberry Pi (marked GND on **Figure 8-9**) with a male-to-female jumper wire. Use a male-to-male jumper wire to connect one leg of the switch to the ground rail on the breadboard, then a male-to-female jumper wire to connect the other leg of the switch to GPIO pin 2 (marked GP2 on **Figure 8-9**).

Figure 8-9: Wiring diagram for connecting a push-button to the GPIO pins

Create a new project in Thonny and save it as **Stop Motion**. Start by importing and setting up the libraries you need for using the camera and the GPIO port:

```
from picamera import PiCamera
from gpiozero import Button
camera = PiCamera()
button = Button(2)
```

Next, type the following:

```
camera.start_preview()
button.wait_for_press()
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Click Run and you'll see a preview of whatever your camera is pointing at. The preview will stay on screen until you press the push-button switch: press it now, and the preview will close after your program saves a picture to the desktop. Find the picture, called **image.jpg**, and double-click to open it and confirm the program is working.

Stop-motion animation involves creating lots of still images, to give the impression of movement when they're all put together. Having all these individual pictures on your desktop would make a mess, so you need a folder to store them all. Right-click anywhere on the desktop that doesn't already have a file or an icon, then choose Create New and Folder (**Figure 8-10**). Call the folder **animation**, all in lower-case letters, then click the OK button.

Figure 8-10: Create a new folder for your captured images

Having to restart your program every time you capture a picture for your animation isn't very good, so you need to change it to run in a loop. Unlike the previous loops you've created, though, this one needs a way to close gracefully – otherwise, if you stop the program while the camera preview is showing, you won't be able to see the desktop any more! To do this, you need to use two special instructions: `try` and `except`.

The Python script:

from picamera import PiCamera

from gpiozero import Button

camera = PiCamera()

button = Button(2)

camera.start_preview()

button.wait_for_press()

camera.capture('/home/pi/Desktop/image.jpg')

camera.stop_preview()

Explanation of the script:

from picamera import PiCamera:

- Imports the PiCamera class, which is used to interact with the Raspberry Pi Camera Module.

from gpiozero import Button:

- Imports the Button class from the gpiozero library. This class simplifies working with physical buttons connected to the Raspberry Pi GPIO pins.

camera = PiCamera():

- Creates an instance of the PiCamera class to control the Raspberry Pi Camera Module.

button = Button(2):

- Creates an instance of the Button class, with the button connected to GPIO pin 2. The number 2 refers to the GPIO pin number (using the BCM numbering system).

camera.start_preview():

- Starts the camera preview, which displays the camera feed on the screen. This helps you see what the camera is capturing in real-time.

button.wait_for_press():

- Pauses the script and waits until the button connected to GPIO pin 2 is pressed.
- The program resumes execution only after the button is pressed.

camera.capture('/home/pi/Desktop/image.jpg'):

- Captures an image using the camera and saves it to the specified path:/home/pi/Desktop/image.jpg.
- You can replace the path and file name with your desired location and name.

camera.stop_preview():

- Stops the camera preview, ending the real-time display of the camera feed.

The Python Script and Initial Test

After completing the circuit, we ran our Phyton script. Initially, the camera preview started successfully, indicating that the camera module was properly initialized. However, pressing the push button did not trigger any actions and the preview screen remained displayed.

We suspected two potential issues:

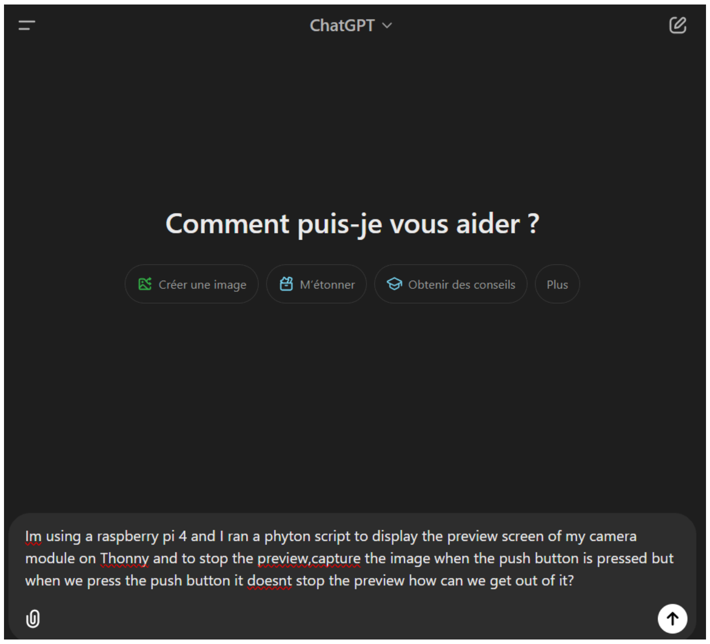
1. A coding error, such as a typo or incorrect logic.
2. A problem in the circuit itself.

So, we double-checked all connections against the guide. Everything appeared correctly configured, which shifted our focus to the script.

Encountering a New Problem

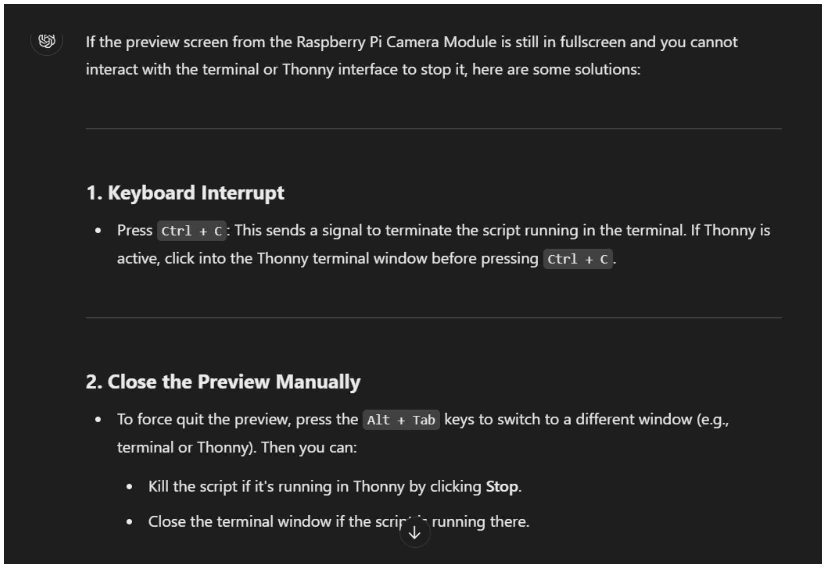
While troubleshooting, another issue arose: the camera preview could not be closed. We were unable to exit the preview screen or stop it, either through the terminal or using shortcut keys. Despite trying multiple solutions suggested by ChatGPT, none worked because the terminal remained inaccessible while the preview screen was active and the keyboard shortcuts were just not working.

The input:



Im using a raspberry pi 4 and I ran a phyton script to display the preview screen of my camera module on Thonny and to stop the preview,capture the image when the push button is pressed but when we press the push button it doesnt stop the preview how can we get out of it?

The output:

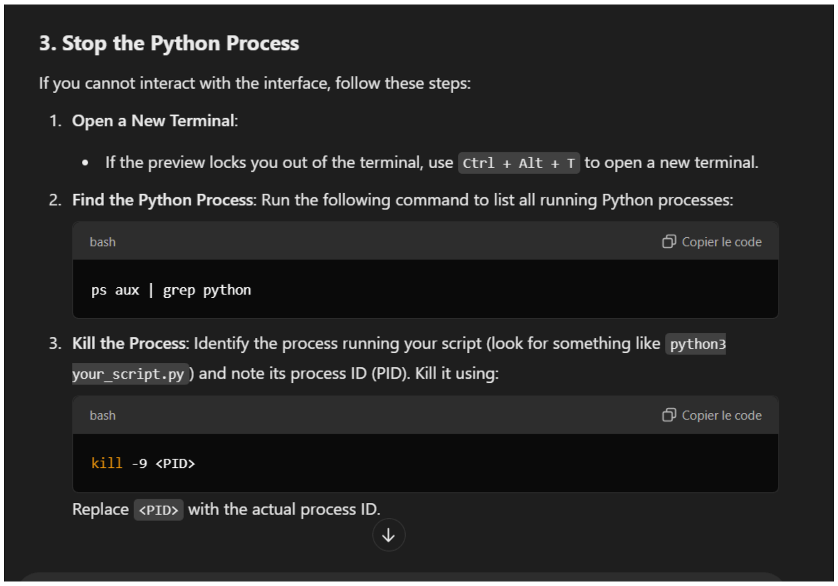

If the preview screen from the Raspberry Pi Camera Module is still in fullscreen and you cannot interact with the terminal or Thonny interface to stop it, here are some solutions:

1. Keyboard Interrupt

- Press `Ctrl + C`: This sends a signal to terminate the script running in the terminal. If Thonny is active, click into the Thonny terminal window before pressing `Ctrl + C`.

2. Close the Preview Manually

- To force quit the preview, press the `Alt + Tab` keys to switch to a different window (e.g., terminal or Thonny). Then you can:
  - Kill the script if it's running in Thonny by clicking **Stop**.
  - Close the terminal window if the scrip[illegible] running there.

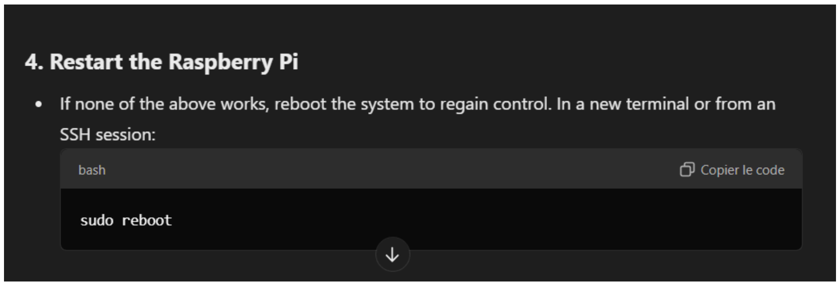

3. Stop the Python Process

If you cannot interact with the interface, follow these steps:

1. **Open a New Terminal**:
   - If the preview locks you out of the terminal, use `Ctrl + Alt + T` to open a new terminal.
2. **Find the Python Process**: Run the following command to list all running Python processes:

   ```bash
   ps aux | grep python
   ```
3. **Kill the Process**: Identify the process running your script (look for something like `python3 your_script.py`) and note its process ID (PID). Kill it using:

   ```bash
   kill -9 <PID>
   ```
   Replace `<PID>` with the actual process ID.


4. Restart the Raspberry Pi

- If none of the above works, reboot the system to regain control. In a new terminal or from an SSH session:

  ```bash
  sudo reboot
  ```

The Solution

As a last resort, we unplugged the power supply to turn off the Raspberry Pi. While this method is not recommended, it was the only option to regain control. After restarting the Raspberry Pi, we searched for alternative ways to test the push button. We decided to seek help from the computer engineering department, as they are more experienced with breadboards, wires, and electrical connections. We approached a random student who kindly referred us to his teacher. The teacher confirmed that the issue could be due to a damaged breadboard—an option we hadn't considered before—or, as we suspected, a problem with the script.

Testing the Push Button

Through YouTube research, we found a video demonstrating how to test the push button without taking a picture, simply to verify its functionality. Using the video's guidance, we implemented a basic script to test the button.

Outcome: The test was successful, confirming that the button was functioning properly.

Link of the video:https://www.youtube.com/watch?v=IHvtJvgM_eQ&t=292s

- Description:This video shows you how to control your Raspberry Pi using a button. It also show you how to turn an LED an and off using a button and the Raspberry Pi GPIO pins.

The simplified script:

```
from gpiozero import Button
button = Button(2)
def button_pressed():
     print("Button was pressed!")
button.when_pressed = button_pressed
```

Next Steps

With the circuit and push button verified, we hypothesized that the issue lay within our script. We continued researching alternative scripts to capture an image using the push button. Unfortunately, none of the scripts we tested worked during this session.

Conclusion

We ended the day with a working circuit and a functional push button, but we still needed to find the correct script to take a picture. Once this is resolved, adapting the script for video recording should be straightforward, requiring only minor modifications. Although we couldn't achieve full functionality, the progress made provided a solid foundation for further development.

Sunday, November 24

My colleague brought the Raspberry Pi home to continue troubleshooting with additional resources, including the camera module, connectors, and a Raspberry Pi guidebook. Her brother helped optimize the setup, suggesting a configuration that required fewer connectors.

Final Setup:

- Connected the Raspberry Pi to the TV.
- Verified that button presses printed messages correctly, indicating no hardware or connection issues.
- Narrowed the problem to the code and the **`python3-picamera`** library.

Steps Taken:

Installed the library:

```
sudo apt update
sudo apt upgrade
sudo apt install python3-picamera
```

- Tested the button integration and verified library functionality.

Wrote the script:

```
from picamera import PiCamera
from gpiozero import Button

button = Button(2)
camera = PiCamera()

def take_picture():
    camera.capture('/home/danat/Desktop/images/image.jpg')

button.when_pressed = take_picture
```

Error Encountered:

Received the error: **"missing 1 required positional argument: 'output'"**. After some debugging, she realized the mistake: missing parentheses when initializing `PiCamera`. Correcting this to `camera = PiCamera()` resolved the issue.

Final Outcome:

The button successfully triggered the camera to capture a photo, resolving the issue completely.

Completed Tasks:

- **Resolved the issue of the undetected camera.**
- **Enabled the tactile button to take pictures. Since we've figured out the solution, it should be straightforward to implement the same functionality for the other buttons.**

Link to Repo:https://github.com/ayakharchafi/Unix-Project24