Raspberry Pi For Beginners - Final Project

Time to start the final project of this course, and practice on everything you've seen until now! First, make sure you have watched the intro video showing the final result of the project. Then, here are some tips to help you get started with the project.

You will create 2 different Python programs.

The first Python program will:

- Monitor the PIR sensor. When some movement is detected, make sure it was detected for a minimum of 5 seconds. You can use the callback functions to know when a movement is detected, and when it's not detected anymore. You can use the "time.time()" function to get the current time. As an additional visual indicator you can also turn on an LED when the PIR sensor detects some movement.
  !!! Alternative option: if your PIR sensor is too unreliable and outputs too much noise, you can use the push button instead. In this case, take a photo if the push button has been pressed for more than 5 seconds. The code will look very similar.
- At this point, take a photo and save it to the /home/pi/photos_final_project folder. Create a function "take_photo()" to take a photo and save it. The file name starts with "img_" and then has a timestamp to make it unique. To get a timestamp you can use the "time.time()" function.
- Then, add the name of the photo (complete path + file name) into a log file named photo_logs.txt (use the append mode with "a" when opening the file). During the setup of the program, check if this file exists and remove it so it doesn't contain data from previous program runs.
- Then, send an email to yourself with the photo as an attachment. Setup email (with Yagmail) in the setup section of the program. Create a "send_photo_by_email()" function to send the photo which was just taken by email.
- Also, as an additional feature, make sure you don't take more than one photo (+ write to log file and send email) every 30 seconds. For that you can also keep a "last_time_photo_taken" variable where you store a time.

The second Python program will:

- Run a Flask application with a default route "/" just returning "Hello"
- Add another route "/check-photos"
- When the function for this URL is called, read the log file (photo_logs.txt created by the other Python program, using the read mode "r"). Count the number of lines in the file, and get the last photo path (last line). You can use a "for line in f" loop to go through all the lines in the file.
- Then, compare the line count with the previous line count (= the last time you called this URL). For that you'll need to use a global variable to keep track of the count.

- Return a string telling how many new photos have been taken since the last time the URL was called, + give the path to the photo file.
- And finally, use an <img src="photo_path"> tag inside the string you return to print the image in the browser. To make things work you'll also have to change the line "app = Flask(__name__)" to become "app = Flask(__name__, static_url_path=CAMERA_FOLDER_PATH, static_folder=CAMERA_FOLDER_PATH)", where CAMERA_FOLDER_PATH is "/home/pi/photos_final_project".

Here's now a quick overview of what I'm going to do in each step of the solution. If you are completely stuck, you can always watch the next solution step, and then work by yourself again to complete the next step. Steps 1-4 are for the 1st Python program, Steps 5-6 for the 2nd Python program:

- **Step1:** Setup the code for the PIR sensor. Detect when there is movement. If there is movement, check that the movement has been detected for 5 seconds. If yes, check that we didn't take a photo in the last 30 seconds. If yes, then print a message to say we are going to take a photo and send it by email. Also, power on one LED when the PIR sensor detects movement. (Reminder: alternatively, you could use the push button instead of the PIR sensor)
- **Step 2:** Setup the camera. Create the function to take a photo and save it into a file. Call that function from one of the PIR callbacks.
- **Step 3:** In the setup section of the program, remove the log file if it exists. Create the function to save new photo paths to the log file, and call that function just after taking a photo.
- **Step 4:** Setup email with Yagmail. Create the function to send an email with an attachment. Call that function just after updating the log file.
- **Step 5:** Create the Flask application with default route and "/check-photos" route. Read the log file and return the number of new photos since the last check.
- **Step 6:** Show the file name of the latest photo, and display the photo
- **Step 7 (Bonus)**: Start the 2 Python programs on boot with systemd, so you don't need to manually get access to your Pi and start the programs from the terminal.

All right, that's enough tips for now! Time for you to start working on this project. Again, if you need any help, feel free to:

- Re-watch lessons from the previous sections.
- Search on Google.
- Just take a break and come back later, this is often a good thing to do.
- And if completely stuck, watch the next solution step.

The solution will help you in 3 ways: first, if you are really stuck in one part, don't waste 10 hours being stuck. After some point, it's best to watch the solution and try to understand it. Second, after you finish the project, you can see how someone else is thinking about the same problem. And third, I will try to give you a solution with clean code and best practices, so you can use those tips to make your programs cleaner and more readable in the future.