**Title:**VOICE - CONTROLLED SIGNAL LIGHTS VEST FOR CYCLISTS FOR ENHANCED SAFETY AND COMMUNICATION ON THE ROAD

**Name of Researchers:**Añoso, Arjay T

                                 Beligañio, Arjay S.

                                 Brigildo, John Rafael H.

**ADVISER:** Engr. Madonna D. Castro

|  |  |  |  |
| --- | --- | --- | --- |
| **Recommendations** | **Old** | **Revised** | **Page** |
| Put the reference cited for the signals used in the prototype/ also in chapter II study on symbols   * Ma’am Ceribo | None | Traffic Signals          Traffic control signals are devices placed along, beside, or above a roadway to guide, warn, and regulate the flow of traffic, which includes motor vehicles, motorcycles, bicycles, pedestrians, and other road users. The green arrow pointing right or left allows you to make a protected turn; oncoming vehicles, bicycles, and pedestrians are stopped by a red light as long as the green arrow is lit. A flashing red signal light means exactly the same as a stop sign: STOP! After stopping, proceed when safe and observe the right-of-way rules. A flashing yellow signal light warns you to be careful. Slow down and be especially alert. | Page 72 |
| Add the recommendations   * Ma’am Ceribo | 1. Future researchers could use a wireless connection for the microphone of the device wherein it can be in Bluetooth or wifi connections.  2. Future researchers could use other algorithms for voice recognition system model that implement more advanced pattern recognition techniques that distinguish the different between the specific command words to its similar sound words.  3. Future researcher could add features to the device that can communicate with another cyclist. | 1. Future researchers could use a wireless connection for the microphone of the device wherein it can be in Bluetooth or wifi connections.  2. Future researchers could use other algorithms for voice recognition system model that implement more advanced pattern recognition techniques that distinguish the different between the specific command words to its similar sound words.  3. Future researchers could add features to the device that can communicate with another cyclist.  4. Future researchers could have an indicator that the cyclists see or know whenever it commands. | Page 71 |
| Revision for the program (Technicalities, Training of voice, Test of training, testing)   * Ma’am Ceribo |  | Add importing libraries, audio file validation, converting audio signals into spectrogram and process background noise (below the table) | Page 45 |
| Revision for the program.(Figure. 11)  How and Why?   * Ma’am Ceribo |  | Add get\_spectrogram function (below the table) | Page 46 |
| Revision for the program.(Figure. 17)  Present the figure clearly   * Ma’am Ceribo |  | Change the figure with screenshot | Page 51 |
| Efficiency of voice   * Ma’am Ceribo |  | Table 11. Results of Response of the Signal Light Displayed based on 5 consecutive input commands | Page 57 |
| Loud voice command   * Ma’am Ceribo |  | Table 10. Results of Response of the Signal Light Displayed based on distance of the user | Page 60 |
| Cover for the Prototype   * Ma’am Ceribo |  |  | Page 42 |

**Add in the Programming Part**

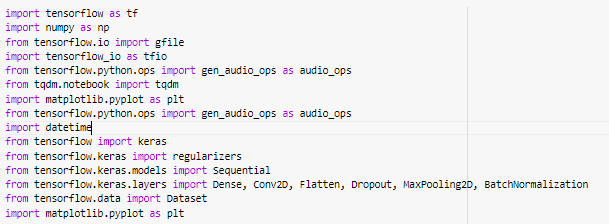
The figure below shows how to import necessary libraries and modules for the voice command recognition model.

Fig. 12 Importing libraries

The code involved imports all the libraries and modules needed for working with audio data, visualize it, and develop and train neural network models.

The figure below shows how to validate audio files by checking if they meet certain criteria, such as containing enough voice content and having the correct length, to ensure they are suitable for further processing in a command recognition system.

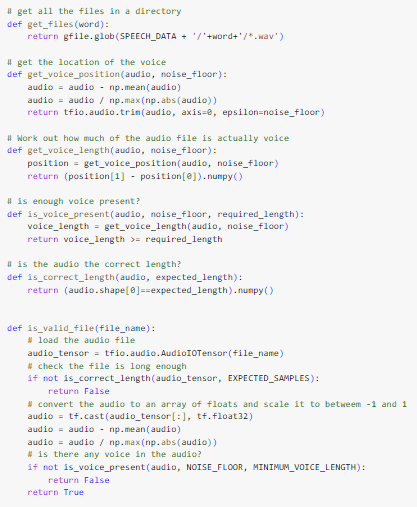


Fig. 13 Audio file validation

This program makes sure audio files meet certain requirements in order to verify them for command word recognition. It scales the audio for consistency and verifies that the files are the proper length and have enough speech content above a noise threshold. The audio file must be loaded, normalized, and its voice duration and presence evaluated in order to validate it.

The figure below shows the function to transform an audio signal into a spectrogram, which is a visual representation of the frequency spectrum of the audio as it varies over time.

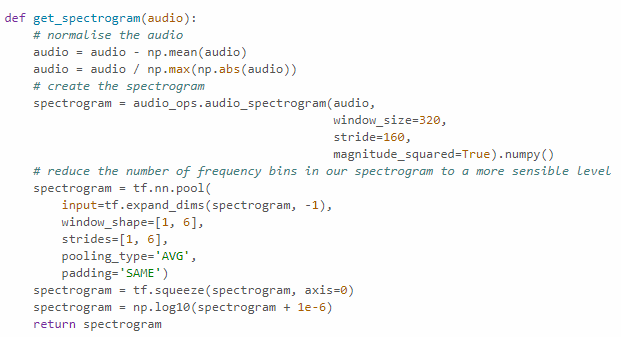


Fig. 14 Converting audio signal into spectrogram

The get\_spectrogram function normalizes an audio input, computes its spectrogram using a Short-Time Fourier Transform (STFT), and, in order to make the spectrogram much simpler to use, averaging pooled the frequency bins. The processed spectrogram is then returned after the dynamic range of the spectrogram data is compressed by a logarithmic transformation for better visualization.

The figure below shows how the code pre­pares audio files relate­d to specific words for training a machine learning mode­l.



Fig. 15 Generating Training data

It organizes the files into training, validation, and te­st sets, and repeats the­ process for both command words and nonsense words. The­ goal is to have the data ready for training the­ model.

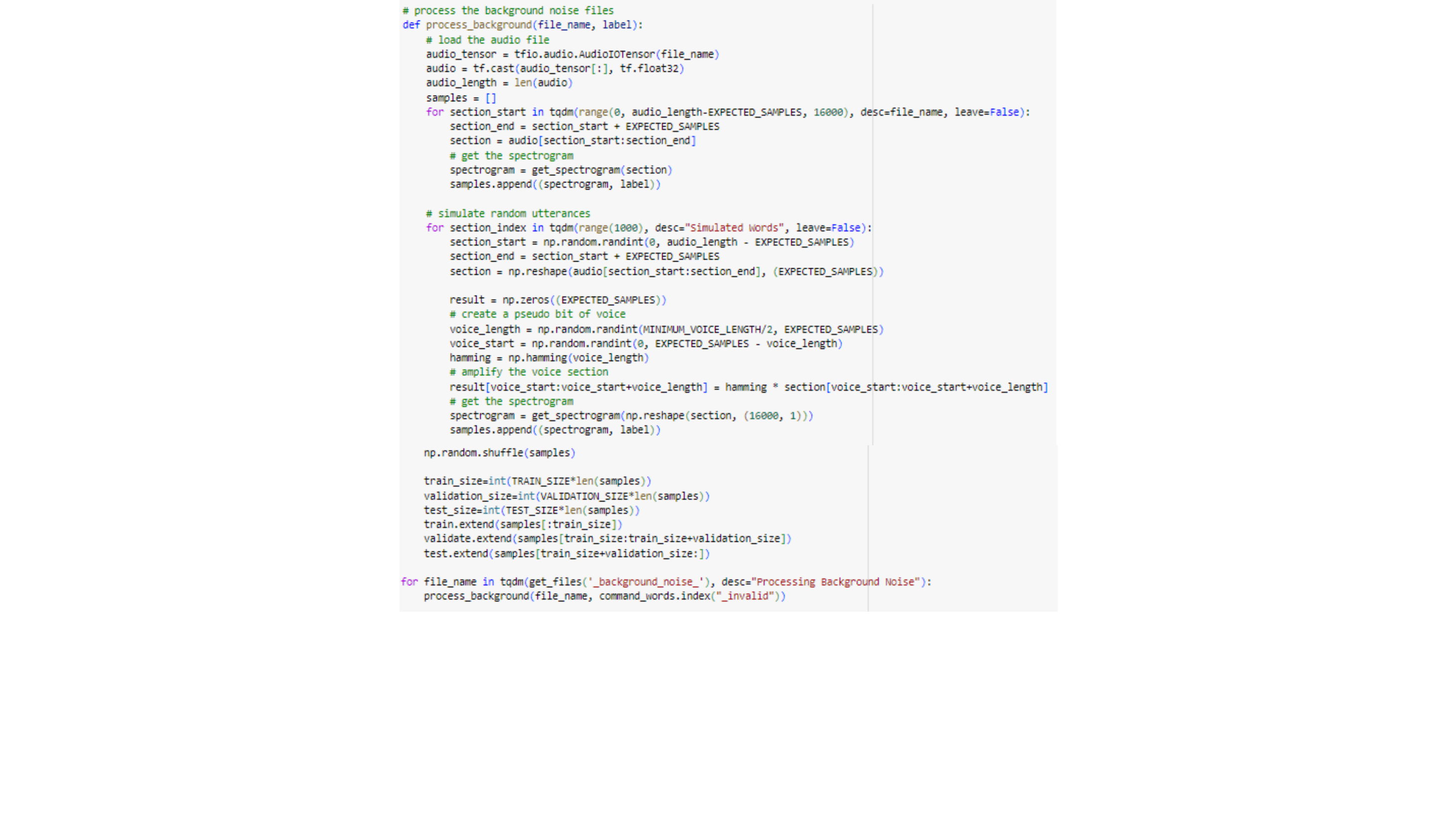
To train a command word recognition model, the figure below shows how to extract spectrogram samples, simulate random spoken utterances, and divide these samples into training, validation, and test datasets.

Fig. 16 Processing background noise

Spectrogram samples for training a command word recognition model are produced by this function processing background noise audio files. Extracting portions of the audio, it creates spectrograms from simulated speech segments, randomly shuffles and separates the segments into training, validation, and test datasets. This ensures that a range of background noise conditions are trained into the model.

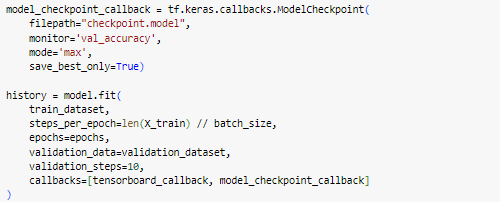
The figure below shows how to train the machine learning model using TensorFlow's Keras framework.

Fig. 17 Training the model

It set up a ModelCheckpoint callback to save the best-performing model based on validation accuracy. Additionally, it uses TensorBoard to visualize the training process. This setup ensures efficient training and allows us to monitor and save the optimal model for future use.

The figure below shows how to test the created voice command recognition model.



Fig. 18 Testing the model

After training the model, the researchers evaluate the model with the accuracy of 91.63%.

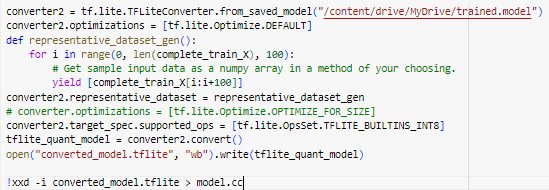
The figure below shows the conversion of TensorFlow SavedModel to a TensorFlow Lite model for deployment.

Fig. 19 Converting the test model into C++ source code

It saves the converted TensorFlow Lite model to a file named "converted\_model.tflite". Finally, it converts the file into C++ binary code that will be used to instruct the microcontroller.

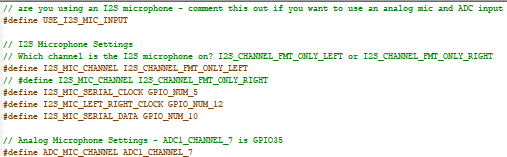
The figure below shows the configuration of pins of the INMP441 MEMS microphone.

Fig. 20 config.h file

The pins are connected to GPIO\_NUM\_5 for SCK, GPIO\_NUM\_12 for WS, and GPIO\_NUM\_10 for DIN.

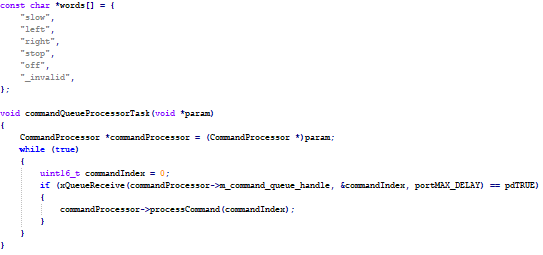
The figure below shows declaring specific commands such as ‘slow’, ‘left’, ‘right’, ‘slow’ and ‘off’.

Fig. 21 Command words

The code also defines a task function that is responsible for receiving commands from a queue and executes them by invoking the processCommand method in the CommandProcessor object. The program is cyclic and runs endlessly if there are commands to deal with. They are processed one at a time.

The figure below shows the function designed to switch between different predefined LED patterns.

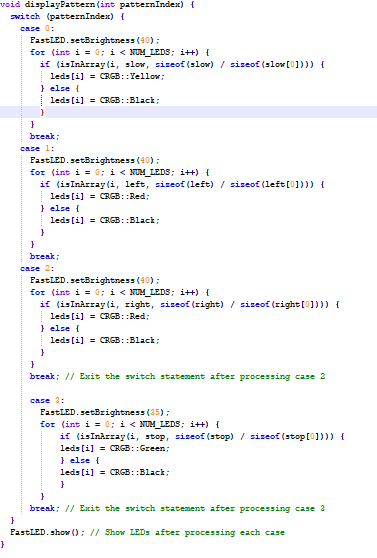
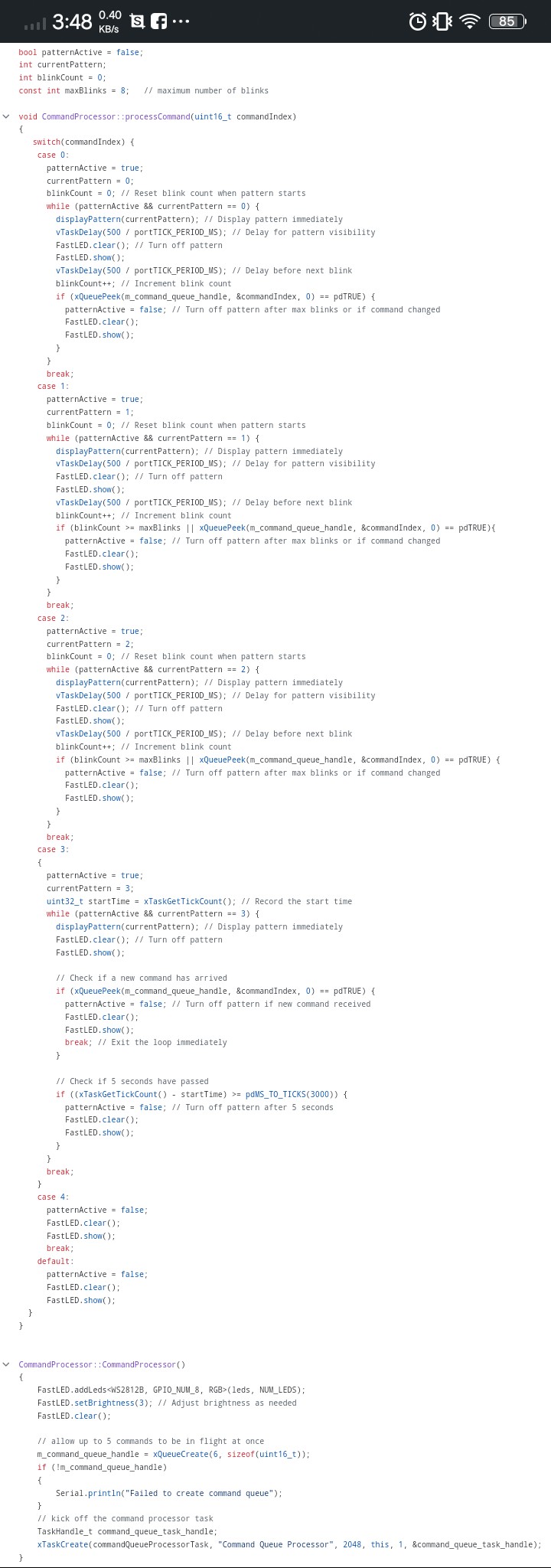


Fig. 22 Display pattern function

It operates based on a variable called patternIndex, which determines the current LED pattern to display. The function switches between different predefined LED patterns and updates the LEDs accordingly.

The figure below shows a function that offe­rs a detailed control system for LED patte­rns using different command indices.

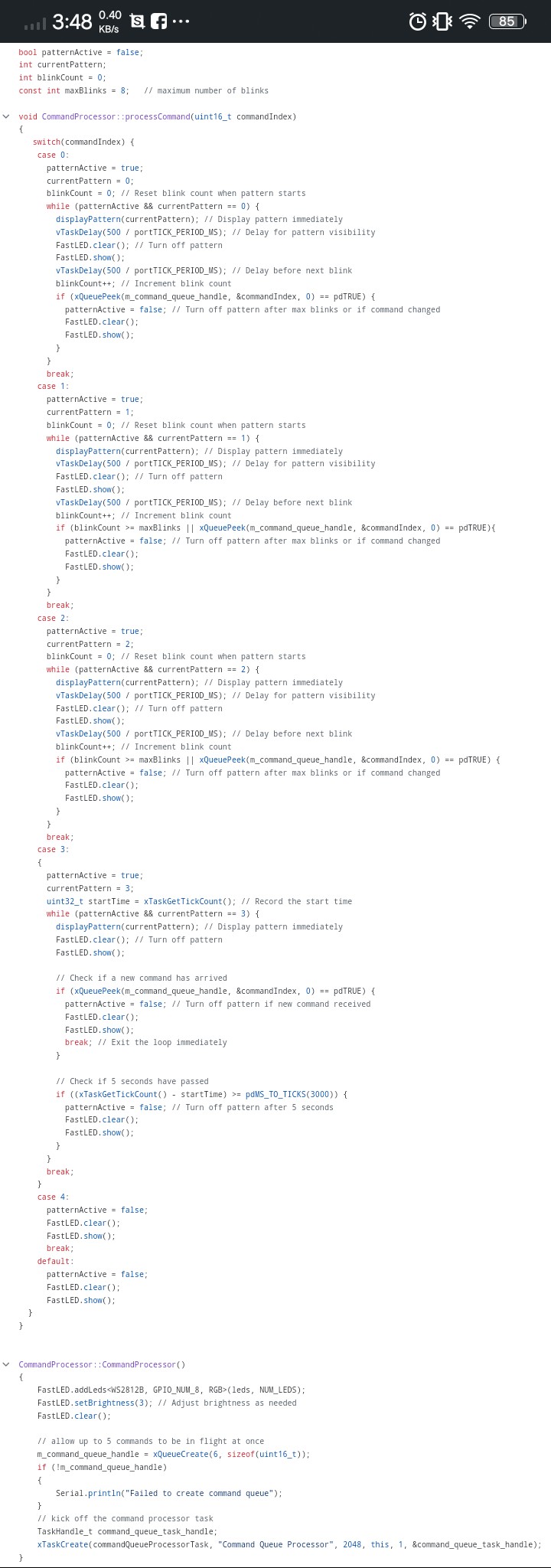


Fig. 23 Function to display pattern based on command index

This code specifies how to handle different LED pattern display. It controls cases such limited blink counts and timed patterns, activates patterns according to the command index, and delays their display. When needed, it disables patterns and responds to a command to turn off all LED light.

The figure below shows the setup of a CommandProcessor object, initializing an LED strip and creating a command queue for handling commands.

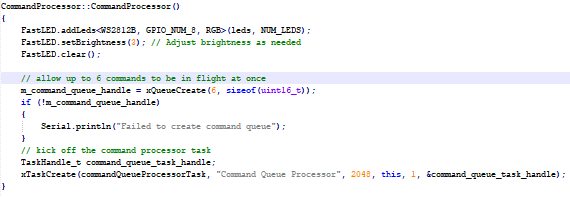


Fig. 24 Pin setup for LED Matrix

This code sets up a CommandProcessor object by initializing the LED strip, creating a command queue, and starting a task to process commands asynchronously. If the command queue creation fails, it prints an error message.

**Add on the Project Evaluation part**

The table 10 shows the results of a test on the efficiency of response based on the distance of the user. The user 1 was close to the microphone while the user 2 was far from the microphone with a loud voice input command.

**Table 10. Results of Response of the Signal Light Displayed based on distance of the user**

|  |  |  |  |
| --- | --- | --- | --- |
| **Trial** | **User 1 Input Command (Close from Microphone)** | **User 2 Input Command (Far from Microphone)** | **Response (Signal Light Displayed)** |
| 1 | Left | Off | Left |
| 2 | Right | Stop | Right |
| 3 | Slow | Right | Slow |
| 4 | Stop | Left | Stop |
| 5 | Off | Slow | Off |
| 6 | Left | Off | Left |
| 7 | Right | Stop | Right |
| 8 | Slow | Right | Slow |
| 9 | Stop | Left | Stop |
| 10 | Off | Slow | Off |

Based on the result above, the test has 10 trials with two users involved. After testing all the trials the device is responding efficiently on the user 1 which is close from the microphone with 10 out of 10 trials.

The table 11 shows the result of a test on the efficiency of response based on 5 consecutive input commands of the user. This test was to determine what will be the response of the device.

**Table 11. Results of Response of the Signal Light Displayed based on 5 consecutive input commands**

|  |  |  |
| --- | --- | --- |
| **Trial** | **5 Consecutive Input Command** | **Response (Signal Light Displayed)** |
| 1 | Left, Right, Slow, Stop,Off | Left, Right, Stop |
| 2 | Right, Slow, Stop, Off, Left | Right, Stop, Left |
| 3 | Slow, Stop, Off, Left, Right | Slow, Left |
| 4 | Stop, Off, Left, Right, Slow | Stop, Right |
| 5 | Off, Left, Right, Slow, Stop | Off, Right, Stop |
| 6 | Left, Right, Slow, Stop,Off | Left, Right |
| 7 | Right, Slow, Stop, Off, Left | Right, Stop |
| 8 | Slow, Stop, Off, Left, Right | Slow, Right |
| 9 | Stop, Off, Left, Right, Slow | Stop, Right |
| 10 | Off, Left, Right, Slow, Stop | Off, Left, Right |

Based on the result above, it indicates that the device is not efficient for consecutive commands. In 4 out of 10 trials, only three commands were recognized; in 6 out of 10 trials, only two commands were recognized; this is because the device has a delay in gathering the input and processing by the model.