iOrthosis-Control

A Convolutional Neural Network powered mobile control system for a hand orthosis/exoskeleton activated by voice commands. All running on a Raspberry Pi 4B.

This repository holds files related to a graduation assignment for the Saxion University of applied sciences. The MSWC dataset is used in combination with a simple CNN architecture.

# Folder structure

The folders have the following purposes:

**Construction Files:** Folder containing CAD files and electronic diagrams.

**Custom dataset:** Custom dataset containing several wavs of environmental sounds and conversations. Can be used to test the model after training. Additional wavs may also be added, although they need to be 16kHz .WAV files

**Dataset MSWC**: A pre-processed dataset containing the words: “red”& “blue”. The words are derived from selected words of the MSWC dataset. [Link to entire dataset](https://mlcommons.org/en/multilingual-spoken-words/)

**On-device inference**: Holds all the needed files for the inference that is used on the Raspberry Pi 4B. A trained model is also needed for inference which is not included in this folder.

**Pre-processing&Training**: Holds pre-processing scripts for the MSWC dataset(MSWC\_ds\_PreProc) and the notebook that is used for training the model(modelTraining\_ControlSystem). The opusdec.exe is also needed for the pre-processing of the dataset.

**Trained models**: Holds several pre-trained models. From V5.0 onwards the models also contain their performance data and metadata. This can be found inside of the model folder under “metadata” & “custom\_dataset\_results”. Currently V5.0 is meant to be used with the provided inference scripts. This model has one issue that it misclassifies silences as commands, due to no silences being included in the training.

# Setup RPI image

This guide will describe the process of running a pre-trained model onto a new Raspberry Pi 4B. The following things are needed:

* Raspberry Pi 4B + USB-C power supply
* Minimum 16GB of storage, so either a flashdrive or SD card(Flashdrive preferred due to read-write cycles damaging SD card)
* Mini displayport -> HDMI/Displayport cable
* IQaudio Codec Zero ADC HAT or another ADC for the microphone
* Secondary monitor + Keyboard + Mouse for RPI

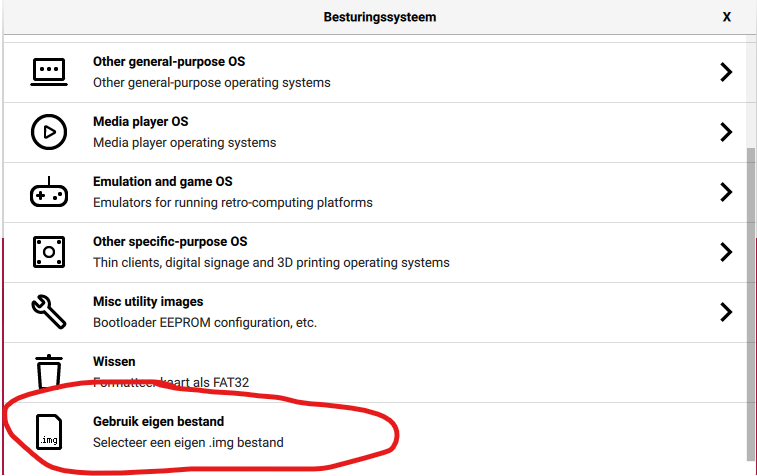
**Flashing image to Raspberry Pi** We start by flashing an image which holds an OS to the Raspberry Pi. We will use an existing image which includes the TensorFlow library, which will make installation easier.

Download the QEngineering 64-bit Raspberry Pi OS image: <https://drive.google.com/file/d/1s8ulI44O96qmVPmWyz8yw3lzamh-32gN/view?usp=sharing>

or their GitHub: <https://github.com/Qengineering/RPi-image>

Download and install the image flasher:

<https://downloads.raspberrypi.org/imager/imager_latest.exe>

Insert the storage device into your computer. Open the Raspberry Pi imager, Go to: Select OS, and scroll down and select the custom image file. [](https://user-images.githubusercontent.com/42100039/160374489-421ad84e-0802-4c09-8e80-84c354f69840.png) [](https://user-images.githubusercontent.com/42100039/160374566-718dac19-fa1b-47c0-a4e5-2a126f687d52.png)

Select the recently downloaded QEngineering Raspberry Pi OS image, and press Open.

Select the right storage device(which should be the inserted SD or flashdrive).

Before flashing also press the cog icon after you have selected the OS file and storage device.

Graphical user interface, website

Description automatically generated

It will open up another window which allows the user to configure certain settings so that they do not need to be changed after flashing.

The hostname can be set to any hostname that is required, here I used “orthosis.local”, that means it can be found in the local network using that name.

SSH is also enabled, this will allow for accessing the raspberry pi using another computer and a program called Putty. You can then access the RPI via a terminal.

Graphical user interface, text, application

Description automatically generated

The password and username is also set. The password is currently set to “hankamp”.

The WiFi network is also configured. The name of the network needs to go at SSID. The password of the network should be entered at the password section.

Graphical user interface, application

Description automatically generated

Lastly the locale settings need to be configured aswell. This ensures the right timezone and wifi settings. Here I used NL LAN county and Amsterdam timezone.

Graphical user interface, application

Description automatically generated

After the configuration press save, double check that you have the right storage device and press write.

This process might take a while.

Afterwards the SD card is flashed. Windows might ask if you want to reformat the drive. **Do NOT do this, as it delete all the progress that is made thusfar.**

# Putty guide

Instead of accessing the RPI by using a second monitor and peripherals. One can also access the RPI by using their main laptop/desktop. This can be done via SSH and a program called Putty. This will allow you to use your main computer for inputting commands to the RPI. This is not needed but may be more convenient.

You can download Putty here: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Download and install the latest putty version for your machine.

After installing open putty, it will look similar to this.

For the hostname enter the hostname that was set during the configuration, so in my case that would be “orthosis.local”. Also add the user that you would like to log on as. So in this case the username is “pi”. Therefor the hostname will be: pi@orthosis or may also be [pi@orthosis.local](mailto:pi@orthosis.local).

Additionally the IP may also be found via the RPI itself by typing “ifconfig” in a command window, or by looking at the router and the connected devices.

**Your laptop/desktop will have to be on the same network as the RPI. This network will have to be the same one that was set during the configuration during the previous steps.**

Give this configuration a name at the saved sessions. And press Save, this will allow for easier future access. After saving the configuration the connection can be established by pressing open.

Graphical user interface

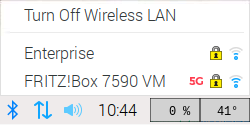
Description automatically generated

After opening the connection. Log in by using the password that was set, so “**hankamp**”. Now you can access the RPI via the command line terminal.

# Setup RPI settings

After flashing, the storage device can be inserted into the Raspberry Pi. Also connect the Raspberry to all the peripherals described in the previous chapter, such as the monitor. Also connect both HATs to the Raspberry Pi.

After connecting the peripherals connect the USB-C PSU to the Pi. Then login to the Raspberry Pi using the password: “3.14” or with “Hankamp” if the previous steps were followed.

imageAfter logging in the Raspberry Pi will start up, this might take a while.

After starting the Raspberry Pi the WiFi may need to be manually enabled if this was not done correctly during the flashing of the SD card. To do this log in to the desired WiFi by clicking the symbol in the top right corner.

Press the button, after a while it shows the available networks, log into the desired network using the credentials.

Test if the WiFi has been correctly configured by opening the internet browser and going to a website.

If the Wi-Fi is set properly you can continue to the next step of the setup, which is needed for the HAT. Open a command prompt (Via SSH and Putty, or via a window on the RPI) and type the following line: “**sudo raspi-config**”

It will open the following screen.

Text

Description automatically generated

Go to interface (3) with the arrow keys and enable SPI and I2C by pressing enter. As can be seen on the following picture.

Text

Description automatically generated

Now go back to the previous screen and go to the advance options (6). Then press enter on Expand Filesystem. This will make sure that the entire SD card is used for storage.

Text

Description automatically generated

After this step go back to the previous screen and press finish. When asked to reboot press Yes. Otherwise type “**sudo reboot now**” Into the command line.

This will reboot the RPI and enable all the settings that were just configured.

Also update the current version by: “**sudo apt-get update**”

And “**sudo apt-get upgrade**” followed by “**sudo reboot now**”

After updating the settings, some additional settings need to be added for the HAT to work. This will be done by editing certain lines manually.

Type: “**sudo nano ~/boot/config.txt**”

Place an # before the line dtparam=audio=on, this will disable HDMI audio. Also add the following line under that one: “**dtoverlay=iqaudio-codec**”.

Save by pressing CTRL+X, and press enter.

Text

Description automatically generated

# Cloning GitHub/Bitbucket repository

After configuring all the necessary settings, the code needs to be imported onto the Raspberry Pi. This will be done via Git. This step can only be completed if a working internet connection has been established on the RPI.

After the reboot that was done during the previous steps, log back into the RPI and open a new command line window.

Before being able to clone the repository you will need the link to the repository. Currently this is: <https://github.com/Gerekt/Orthosis-Control.git>

Now clone the repository by using the link.

Type: “**sudo git clone** [**https://github.com/Gerekt/Orthosis-Control.git**](https://github.com/Gerekt/Orthosis-Control.git)”

The repository has now been cloned onto the RPI.

# Downloading Python libraries

The image that we used for the RPI already has several Python libraries pre-installed such as TensorFlow. However we need some additional libraries for the code.

First sounddevice: “**python3 -m pip install sounddevice**”

We cant install sounddevice easily as it needs portAudio to work, which is a C library. The only way to do this on a RPI with existing python modules is to manually compile the library with the following commands.

“**cd ~” and “wget** [**http://files.portaudio.com/archives/pa\_stable\_v190700\_20210406.tgz**](http://files.portaudio.com/archives/pa_stable_v190700_20210406.tgz)”

“**tar – xvzf pa\_stable\_v190700\_20210406.tgz**”

“**sudo apt-get install libasound-dev**”

“**sudo apt-get install alsa-utils**” And “**sudo apt-get install mpg321**”

“**sudo apt-get install lame**”

“**cd portaudio**” And “**./configure && make**”

“**sudo make install**” And “**sudo ldconfig**”

“**sudo cp lib/.libs/libportaudio.a ~/Orthosis-Control/On-device inference/”**

“**sudo cp ~/portaudio/include/portaudio.h ~/Orthosis-Control/On-device\ inference/**”

Now we have sounddevice and the C library. Next up Scipy. But first all the dependencies of scipy.

“**sudo apt-get upgrade**” and “**sudo apt-get update**”

“**python3 -m pip install imutils**” And “**python3 -m pip install matplotlib**”

“**python3 -m pip install ipython**” And “**python3 -m pip install jupyter**”

“**python3 -m pip install sympy**” and “**python3 -m pip install nose**”

Jupyter and Sympy may give errors when installing, during my installation this happened but did not matter for the installation of scipy.

“**sudo apt install python3-gpiozero**”

The last modules will take quite a long time, so be patient. (+20 min)

“**python3 -m pip -vv install pandas**” and “**python3 -m pip -vv install numpy**”

“**python3 -m pip -vv install scipy**”

# Testing configuration

Now we should have all the required libraries and files. We will test if everything is working by running the main control script. Go to it by: “**cd ~/Orthosis-Control/On-device inference/**”

And run the script with: “**python3 ControlSystem.py**”

After loading all the libraries it should start and give no library errors. It may give some errors related to the input device. This can be solved by running another script in the same folder. This script lists all the available input devices, the right one should be selected and copied into the main control system script.

Run the script with: “**python3 audioListDevices.py**”. The IQaudio microphone input should be the default.

Additional troubleshooting may be done with “**arecord -l**” and “**arecord -L**”, which lists all available devices.

And “**sudo alsamixer**”, which allows for changing certain parameters of the audio cards. Select the right card by pressing F6. Check if the microphones are enabled(Green box), if not enable or increase the gain by pressing “M” and the arrow keys respectively.

# Problems/Improvements

This chapter will summarize potential improvements that could be made and also the problems that the system currently has. If possible, solutions will also be provided that may solve the problem.

## Software

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | Type | Description | Solution |
| Models not trained with silences | Problem | The models are not trained with samples of silences and therefore misclassify silent audio as commands. | Add training silent training samples to unknown class or add new class containing silences and light noise. |
| Electronic peripherals not used | Improvement | The peripherals (LED’s, IO expander, and power button) are not used in the software | Add functionality into code. |
| Control script stops after 20sec | Improvement | The main script ControlSystem.py stops after 20sec. It stops as this made troubleshooting easier | Easier solution is to remove the 20 second variable in the loop to make it run indefinitely.  May also add start stop capability to script, meaning that the classes get deleted and loaded again when the microphone activation button is pressed. |
| Device ID of IQAudio ADC can change | Improvement | The Device ID of the input devices of the IQAudio HAT can change after reboots. Which will result in no captured audio in the main control script. | Change HAT with USB ADC as the HAT is difficult to setup and troubleshoot. An USB ADC that is supported by the RPI may give better results. |
| TensorFlowLite capability | Improvement | The script currently uses the TensorFlow library and a TenserFlowLite model. This works fine but fully using TensorFlowLite as the library will give better performance. | Change TensorFlow library with TensorFlowLite. Will require slight reformatting of the code in the main script and wavToSpectrogram.py.  Should be possible as no TensorFlow function is needed anymore. |

## Mechanical

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | Type | Description | Solution |
| Sluggish case design | Improvement | The cases for the control system and user interface are big, bulky and heavy. They can be shrunk in size drastically. | Remodel both cases and print using less layers/walls. The models currently have a lot of empty space. |
| No aux passthrough | Improvement | The AUX cable from the microphone needs to go into the case, currently this results in a hole in the wall of the case. | Add an AUX passthrough where the user can plugin the microphone. The passthrough is fixed into the walls of the case. will then be connected to the ADC. |
| No analog inputs | Problem | The RPI does not have any analog inputs, but one is needed for the motor position | An I2C ADC can be added in the user interface, like for example the AD7992.  Alternatively the motor position may be discarded, I.E. not be used. |
| Heat/cooling | Problem | The RPI gets very hot after a while. The HAT’s restrict airflow. | Add a special cooling case made of metal. A cooling fan may also be added.  A sleeker HAT will also improve cooling. |

## Electrical

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | Type | Description | Solution |
| System enabler | Improvement | System enabler is not implemented. System can therefor not be automatically shutdown | Select the right MOSFETS and rebuild the enabler board. |
| Cables | Improvement | Cables inbetween cases currently uses individual cables and are not long enough. | Rewire with cable with multiple strands like an ethernet cable. |