

VoiceLink tutorial

This project utilizes the ESP32S3 as an AI processor for voice command recognition. The following tutorial will guide you through the step-by-step implementation of the system.

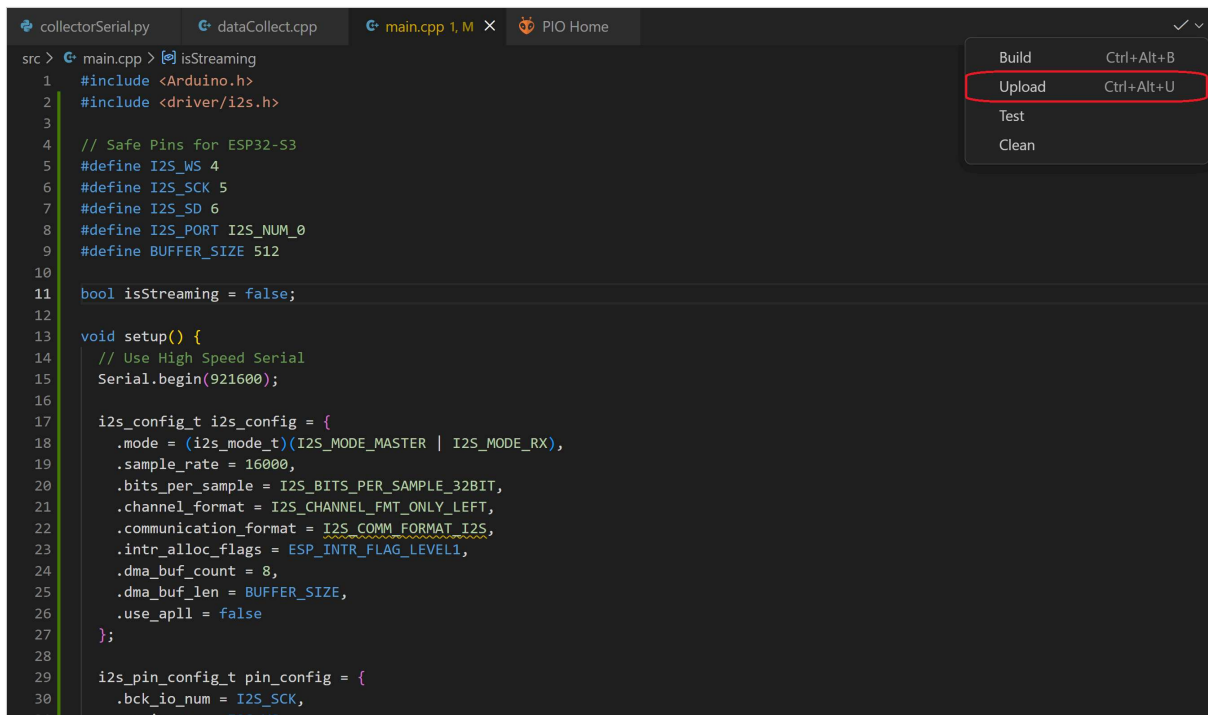
Part 1: Data Collection

In this section, we will record raw audio data from the ESP32 to build our dataset for training the AI model.

Step 1: Upload Firmware

We need to flash the ESP32 with code that reads audio from the microphone and sends it over Serial.

1. Open your project in the **PlatformIO** environment.
2. Locate the file named `dataCollection.cpp` in your file explorer.
3. Copy **all the content** from `dataCollection.cpp`.
4. Open `main.cpp` (located in the `src` folder).
5. Replace the entire content of `main.cpp` with the code you just copied.
6. **Uncomment** the code if necessary.
7. Connect your ESP32-S3 board to your computer via USB.
8. Click the **Upload** button (Right arrow icon) in PlatformIO.



Step 2: Configure Python Script

Prepare the Python script that captures audio from the USB port.

1. Check Device Manager to find your ESP32's **COM Port number** (e.g., COM3, COM9).
2. Open `collectorSerial.py`.
3. Find the line defining `COMPORT` and update it.

```
# ----- //
```

```
# // // REPLACE WITH YOUR ESP32's port
```

```
COMPORT = 'COM9' # Update to your ESP32's port
```

```
#----- //
```

Step 3: Record Your First Class

We will now record the keywords.

1. Run the script: `python collectorSerial.py`
2. Enter the label you want to record (e.g., `on`, `off`, or creative names like `haha`, `hehe`).

3. When you see `[READY] Press SPACE to record...`, press **SPACE** and speak clearly into the mic.

```
--- Voice Command Data Collector ---  
Enter label to record (e.g., 'on', 'off', 'background'): hehe  
Collecting data for: hehe  
Press 'Ctrl+C' to stop or change label.  
  
[READY] Press SPACE to record' 60 'for label: 'hehe'
```

Step 4: Repeat & Adjust

Repeat Step 3 for your second keyword and for background noise.

- To change recording duration, adjust this line in the Python script:

```
#----- //  
RECORD_SECONDS = 60 # Change this for longer dataset  
#----- //
```

Part 2: Uploading & Splitting

Now we upload the raw audio files to Edge Impulse and chop them into 1-second samples.

Step 1: Upload Data

1. Go to the **Data acquisition** tab in Edge Impulse.
2. Click **Collect new data** -> **Add data** -> **Upload data**.
3. Select your recorded file.
4. **Settings:**
 - Method: "Automatically split between training and testing".
 - Label: Enter the label matching your file (e.g., 'on').

5. Click **Upload data**.

The image displays two screenshots of the Edge Impulse Studio interface, illustrating the process of uploading data to a project.

Top Screenshot: The main view is the 'Project info' page for 'quocanmeomeo-project-1'. The 'Getting started' section offers three options: 'Add existing data', 'Collect new data' (highlighted with a red box), and 'Upload your model'. The 'Sharing' section indicates the project is private. The 'Published versions (0)' section shows no published versions.

Bottom Screenshot: The 'Dataset' page is shown, with the 'Collect data' tab selected. The 'Collect data' section has a red box around the 'Upload data' button. The 'Dataset' section shows 'Training (0)' and 'Test (16)' data points, with an 'Add data' button.

Both screenshots show a sidebar with navigation options: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse #2, Create impulse, MFCC, Classifier, Retrain model, Live classification, Model testing, Upgrade Plan, and View plans.

URL: <https://studio.edgeimpulse.com/studio/865771/acquisition/training?page=14>

Upload mode

☒ Select individual files ?

☐ Select a folder ?

Select files

on_000.wav

Upload into category

☒ Automatically split between training and testing ?

☐ Training

☐ Testing

Label

☐ Infer from filename ?

☐ Leave data unlabeled ?

☒ Enter label:

on

Step 2: Split Samples

Since we recorded one long file, we need to split it into individual keywords.

1. Find your uploaded file in the list.
2. Click the **three dots (:)** -> **Split sample**.
3. The app will auto-detect segments. **Validate** that the boxes cover the audio correctly.
4. Click **Split**.

The screenshot shows a dataset management interface. At the top, there are two status bars: 'DATA COLLECTED 2m 36s' and 'TRAIN / TEST SPLIT 78% / 22%'. Below these is a 'Dataset' section with a table of samples. The table has columns for 'SAMPLE NAME', 'LABEL', and 'ADDED'. A context menu is open over the 'background_000' sample, with 'Split sample' highlighted. To the right, there is a 'RAW DATA' section showing a waveform for 'background_000' and a 'Metadata' section with the text 'No metadata.'

SAMPLE NAME	LABEL	ADDED
off_000.s9	off	Today
off_000.s8	off	Today
off_000.s7	off	Today
off_000.s4	off	Today
off_000.s3	off	Today
off_000.s2	off	Today
background_000	noise	Today, 11:11:57 1m 0s
on_000.s48	on	Today, 11:11:57 1s
on_000.s47	on	Today, 11:11:57 1s
on_000.s44	on	Today, 11:11:57 1s
on_000.s43	on	Today, 11:11:57 1s

Step 3: Repeat & Finalize

1. Repeat splitting for your other keyword file.
2. For the **Noise/Background** file: **Do NOT split the sample**. Keep it continuous.
3. Go to the **Dashboard** tab, scroll down, and click **Perform train / test split**.

Part 3: Training the Model

We will design the processing pipeline and train the Neural Network.

Step 1: Create Impulse

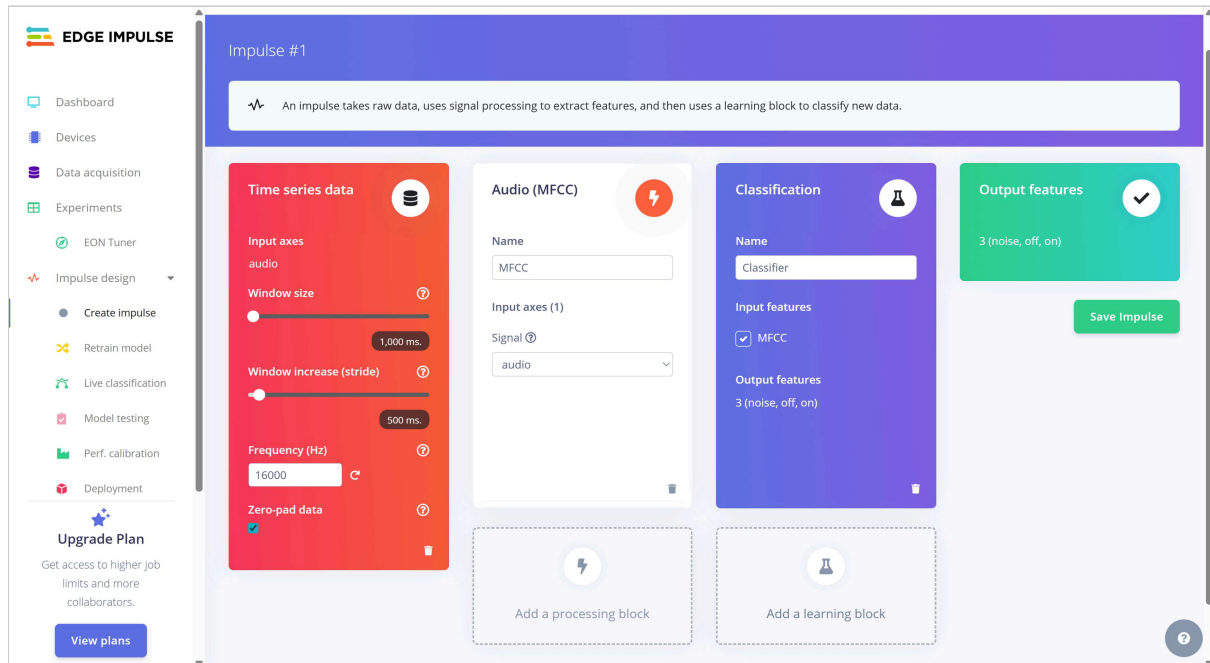
Navigate to the **Create Impulse** tab.

- **Time Series Data:**
 - Window size: **1000 ms**.
 - Window increase: **500 ms**.
 - Zero-pad data: **Unchecked**.

⚠ **CRITICAL: Frequency Setting**

This must match your hardware exactly. If your data says 6392Hz, type 6392 here. Do not use 16000Hz unless the data is actually 16000Hz.

- **Processing Block:** Add "Audio (MFCC)".
- **Learning Block:** Add "Classification (Keras)".
- Click **Save Impulse**.



Step 2: Generate Features (MFCC)

1. Go to the **MFCC** tab.
2. Click **Save parameters** (Defaults are usually fine).

Raw features

5, -3, 7, 21, 18, 35, 46, 35, 34, 18, 25, 78, 53, 97, 106, 1...

Label

off

Parameters

Autotune parameters

Mel Frequency Cepstral Coefficients

Number of coefficients

13

Frame length

0.02

Frame stride

0.02

Filter number

32

FFT length

256

Normalization window size

101

Low frequency

0

High frequency

Click to set

Pre-emphasis

Coefficient

0.98

Save parameters

DSP result

Cepstral Coefficients

Processed features

-0.8558, -0.7634, 0.4802, 0.7893, 0.6517, -1.3398, -0.4257, 0.1448, -0.2732, -1...

On-device performance

PROCESSING TIME

361 ms.

PEAK RAM USAGE

15 KB

3. Click **Generate features**.

Training set

Data in training set

2m 16s

Classes

3 (noise, off, on)

Training windows

195

Generate features

Feature generation output

(0)

4. **Visual Check:** Look at the "Feature Explorer" graph.

- o **Good:** Distinct clusters of colors.
- o **Bad:** Dots mixed together like a smoothie.

Step 3: Classifier (Training)

Go to the **Classifier** tab to train the brain.

1. **Settings:**

- Training cycles: Set to **60**.
- Learning rate: **0.005**.
- Data augmentation: **CHECKED**.

2. Click **Save & train**.

Neural Network settings

Training settings

Number of training cycles 100

Use learned optimizer ☐

Learning rate 0.005

Training processor CPU

Advanced training settings

Audio training options

Data augmentation ☐

Neural network architecture

Neural network Transfer learning Load preset...

Input layer (650 features)

Reshape layer (13 columns)

1D conv / pool layer (8 filters, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (16 filters, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (3 classes)

Save & train

Step 4: The Result

Wait for training to finish and examine the **Confusion Matrix**.

- **Goal:** >85% accuracy for Keywords, >90% for Noise.



Part 4: Deployment (The Transplant)

Goal: Move the "brain" onto the chip for offline use.


Step 1: Export the Library

- Go to the **Deployment** tab in Edge Impulse.
- Search for **Arduino Library**.
- Select **TensorFlow Lite** as Deployment engine.
- Select **Quantized (int8)**. **(Critical for ESP32 performance)**.
- Click **Build** to download the **.zip** file.

Configure your deployment


You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

Deployment target



Arduino library
 An Arduino library with examples that runs on most Arm-based Arduino development boards.

Inference engine



TensorFlow Lite

Model optimizations and performance

Model optimizations can increase on-device performance but may reduce accuracy. Performance estimate for Yolo_Uno - [Change target](#)

Quantized (int8)

Selected ✓

	MFCC	CLASSIFIER	TOTAL
LATENCY	361 ms.	4 ms.	365 ms.
RAM	15.4K	6.4K	15.4K
FLASH	-	51.6K	-
ACCURACY			-

Unoptimized (float32)

Select

	MFCC	CLASSIFIER	TOTAL
LATENCY	361 ms.	35 ms.	396 ms.
RAM	15.4K	10.7K	15.4K
FLASH	-	54.3K	-
ACCURACY			-

To compare model accuracy, run model testing for all available optimizations.

Run model testing

Build

Step 2: Install in PlatformIO

1. **Unzip** the downloaded file on your computer.
2. **Drag & Drop:** Move the extracted folder (e.g., `ESP_VOICE_inferencing`) into your PlatformIO project's `lib/` folder.

Structure Check: Verify your project folder looks exactly like this:

```
MyProject/
├─ lib/
```

```
|   └─ ESP_VOICE_inferencing/  <-- The folder you just added
|   └─ src/
|       └─ main.cpp
|       └─ modelRun.cpp
└─ platformio.ini
```

Step 3: The Final Code

Now we replace the data collection script with the actual AI logic.

1. Locate the file named `modelRun.cpp` in your file explorer.
2. Copy **all the content** from `modelRun.cpp`.
3. Open `main.cpp` (located in the `src` folder).
4. Replace the entire content of `main.cpp` with the code you just copied.
5. **Uncomment** the code if necessary.
6. **Important:** Update the library include line at the top of the code to match your folder name:

```
// ----- //
// REPLACE WITH YOUR LIBRARY
#include "quocanmeomeo-project-1_inferencing.h"
// ----- //
```

Step 4: Upload & Test

1. Connect your ESP32.
2. Click **Upload** in PlatformIO.
3. Open the Serial Monitor and start speaking your keywords!