

# 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.

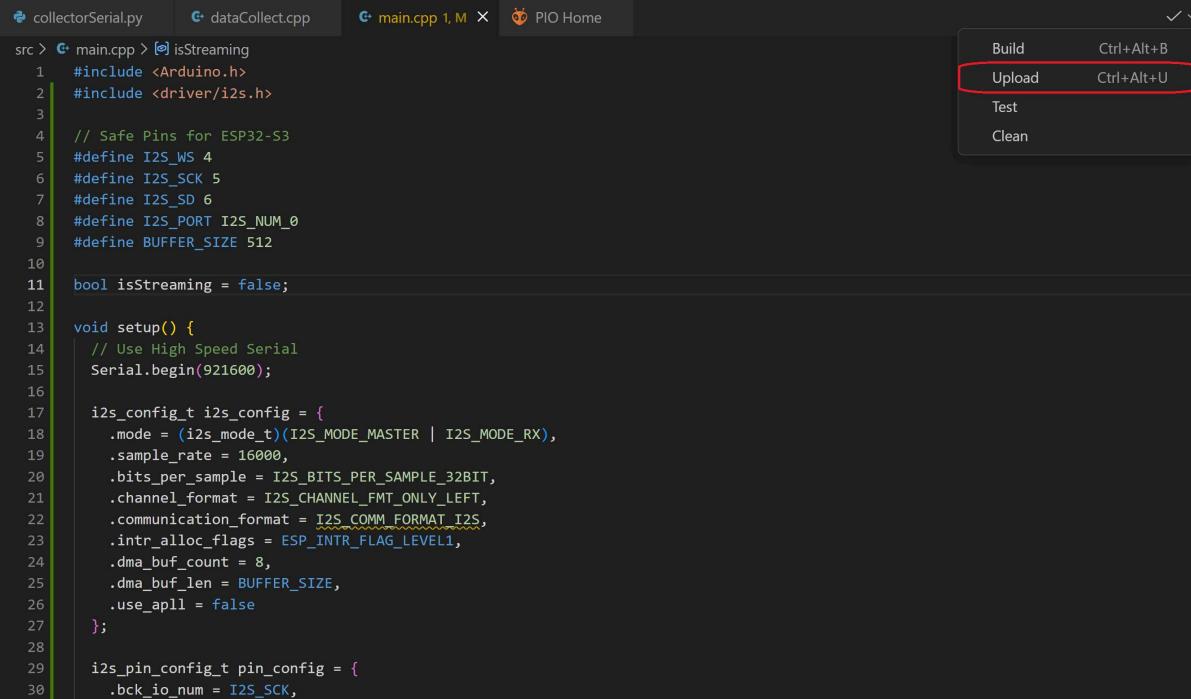
## Part 1: Data Collection

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.



```
src > main.cpp > isStreaming
1 #include <Arduino.h>
2 #include <driver/i2s.h>
3
4 // Safe Pins for ESP32-S3
5 #define I2S_WS 4
6 #define I2S_SCK 5
7 #define I2S_SD 6
8 #define I2S_PORT I2S_NUM_0
9 #define BUFFER_SIZE 512
10
11 bool isStreaming = false;
12
13 void setup() {
14     // Use High Speed Serial
15     Serial.begin(921600);
16
17     i2s_config_t i2s_config = {
18         .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
19         .sample_rate = 16000,
20         .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
21         .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
22         .communication_format = I2S_COMM_FORMAT_I2S,
23         .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
24         .dma_buf_count = 8,
25         .dma_buf_len = BUFFER_SIZE,
26         .use_apll = false
27     };
28
29     i2s_pin_config_t pin_config = {
30         .bck_io_num = I2S_SCK,
31         .bck_io_num = I2S_WS,
```

## 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 screenshots illustrate the Edge Impulse studio interface for project management and data collection.

**Top Screenshot (Project Overview):**

- Left Sidebar:** Includes links for Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse #2 (selected), Create impulse, MFCC, Classifier, Retrain model, Live classification, Model testing, and Upgrade Plan.
- Header:** Shows the project name "quocanmeomeo / quocanmeomeo-project-1" and a "PERSONAL" badge.
- Central Content:** Displays the project title "quocanmeomeo-project-1" and a brief description: "This is your Edge Impulse project. From here you acquire new training data, design impulses and train models." Below this are sections for "KEYWORD SPOTTING" and "Getting started". The "Getting started" section contains three buttons: "Add existing data", "Collect new data" (which is highlighted with a red box), and "Upload your model".
- Right Sidebar:** Includes a "Sharing" section (set to "Private") stating "This project is private, only invited users can edit and view.", and a "Published versions (0)" section with a "Publish a version of your project" button.

**Bottom Screenshot (Dataset Tab):**

- Left Sidebar:** Same as the top screenshot.
- Header:** Shows the project name and a "PERSONAL" badge.
- Central Content:** Displays the "Dataset" tab with the following sections:
  - DATA COLLECTED:** Shows "16s".
  - TRAIN / TEST SPLIT:** Shows "0% / 100%".
  - Collect data:** Contains a "Connect a device" link and a "Upload data" button (which is highlighted with a black box).
  - Dataset:** Shows "Training (0)" and "Test (16)".
  - Add data:** Includes a "Add data" button and a "Start building your dataset by adding some data." note.
- Right Sidebar:** Includes a "RUN THIS MODEL" button.

**URL:** https://studio.edgeimpulse.com/studio/865771/acquisition/training?page=1#

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 ?

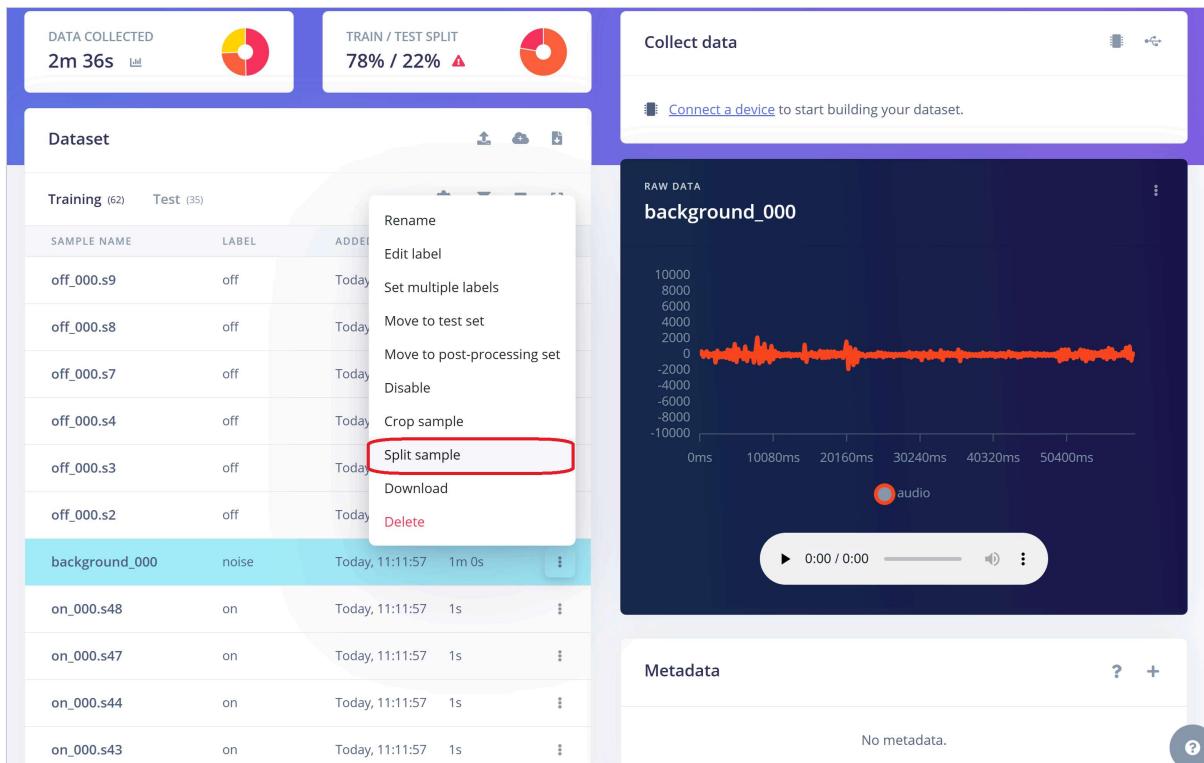
Leave data unlabeled ?

Enter label:

## 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**.



## 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

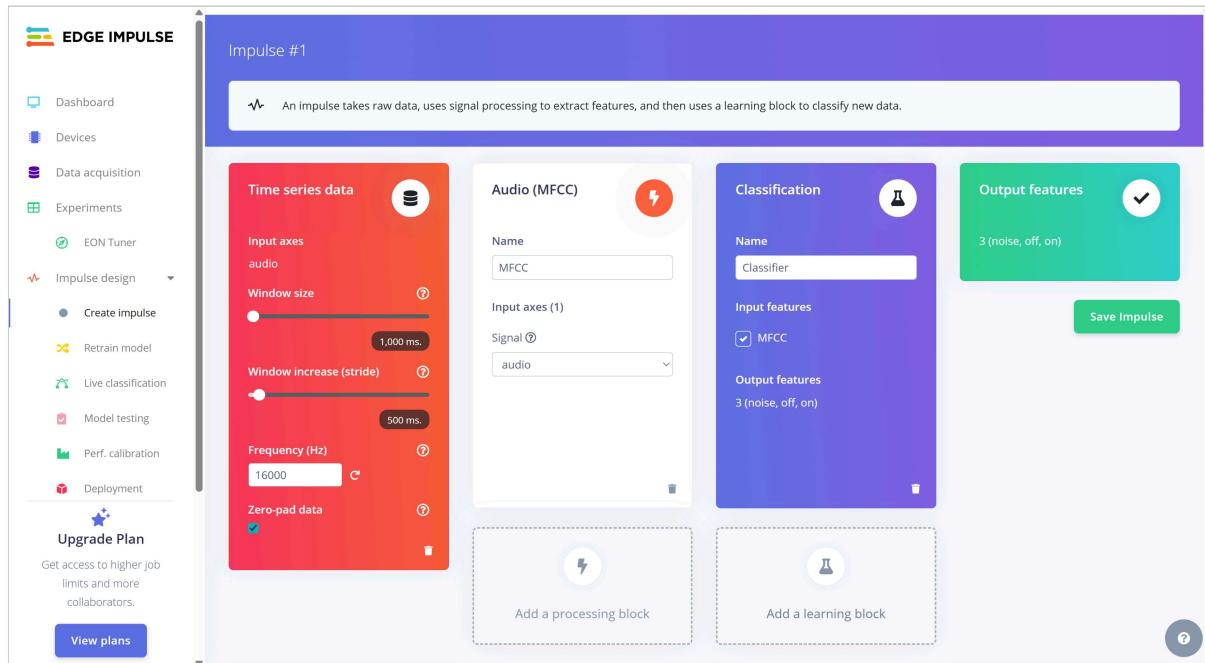
Navigate to the **Create Impulse** tab.

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

## ⚠ CRITICAL: Frequency Setting

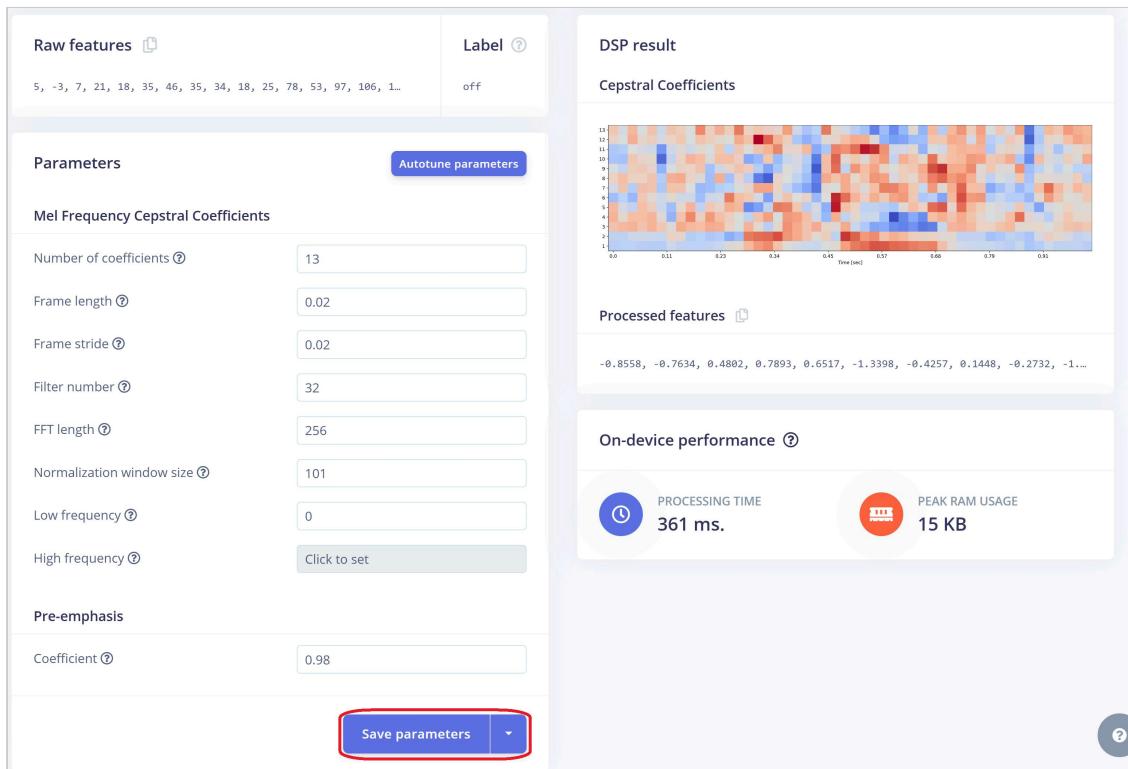
This must match your hardware exactly. If your data says **6392Hz**, type **6392** here. Do not use **16000Hz** unless the data is actually **16000Hz**.

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

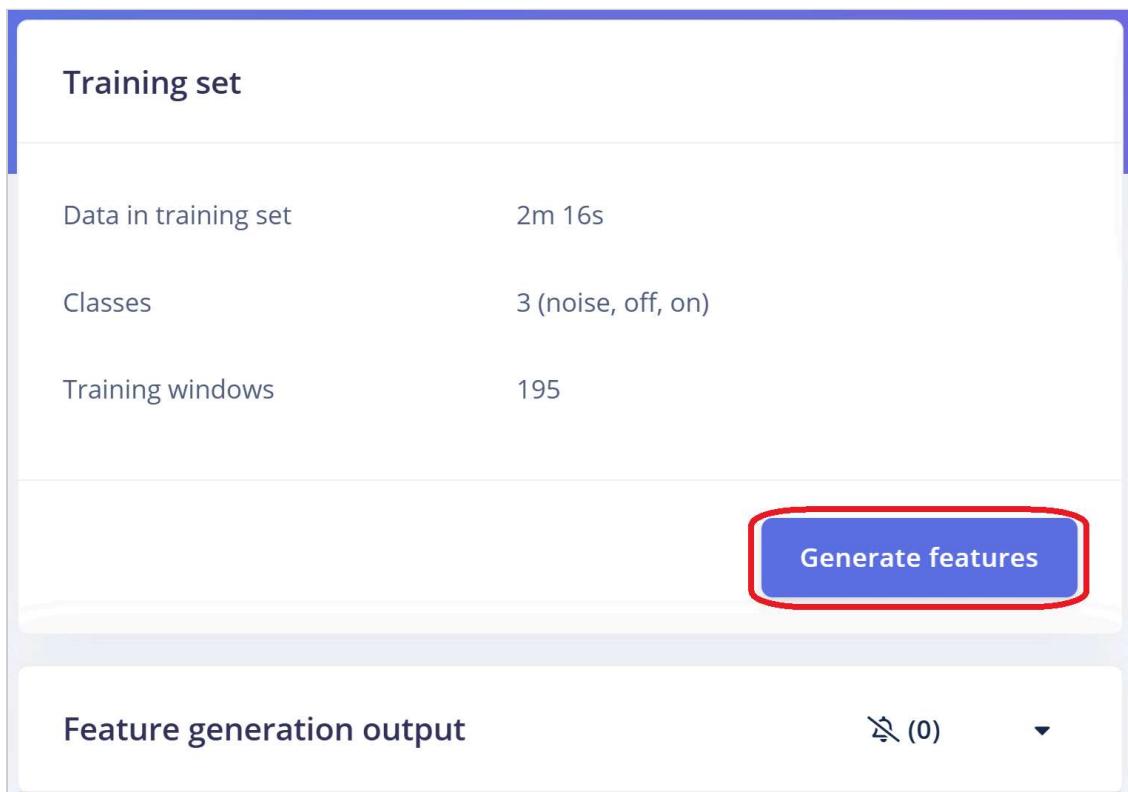


## Step 2: Generate Features (MFCC)

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



3. Click **Generate features**.



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

- **Good:** Distinct clusters of colors.
- **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**.

The screenshot shows the 'Neural Network settings' configuration page. It includes sections for 'Training settings' (Number of training cycles: 100, Use learned optimizer: unchecked, Learning rate: 0.005, Training processor: CPU), 'Advanced training settings' (Data augmentation: unchecked), 'Audio training options' (unchecked), and 'Neural network architecture' (Neural network selected). The architecture consists of an input layer (650 features), followed by a series of layers: 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, and an output layer (3 classes). A red box highlights the 'Save & train' button at the bottom right.

Neural Network settings

Training settings

Number of training cycles ② 100

Use learned optimizer ②

Learning rate ② 0.005

Training processor ② CPU

Advanced training settings

Audio training options

Data augmentation ②

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.

Model Model version: ② Quantized (int8) ▾

Last training performance (validation set)

ACCURACY 100.0% LOSS 0.01

Confusion matrix (validation set)

	NOISE	OFF	ON
NOISE	100%	0%	0%
OFF	0%	100%	0%
ON	0%	0%	100%
F1 SCORE	1.00	1.00	1.00

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve ②	1.00
Weighted average Precision ②	1.00
Weighted average Recall ②	1.00
Weighted average F1 score ②	1.00

Data explorer (full training set) ②

The Data Explorer section displays a scatter plot of training data points. The x-axis represents the 'noise' category, with points colored yellow ('noise - correct'), green ('off - correct'), and dark green ('on - correct'). The y-axis represents the 'off' category. The plot shows a clear separation between the three classes, with most 'noise' points clustered at the top, 'off' points in the middle, and 'on' points at the bottom. A legend on the left identifies the colors: yellow for 'noise - correct', green for 'off - correct', and dark green for 'on - correct'.

On-device performance ②

Calculating... this could take a few minutes  
Show logs

## 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\_Undo - [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!