



VOICE RECOGNITION

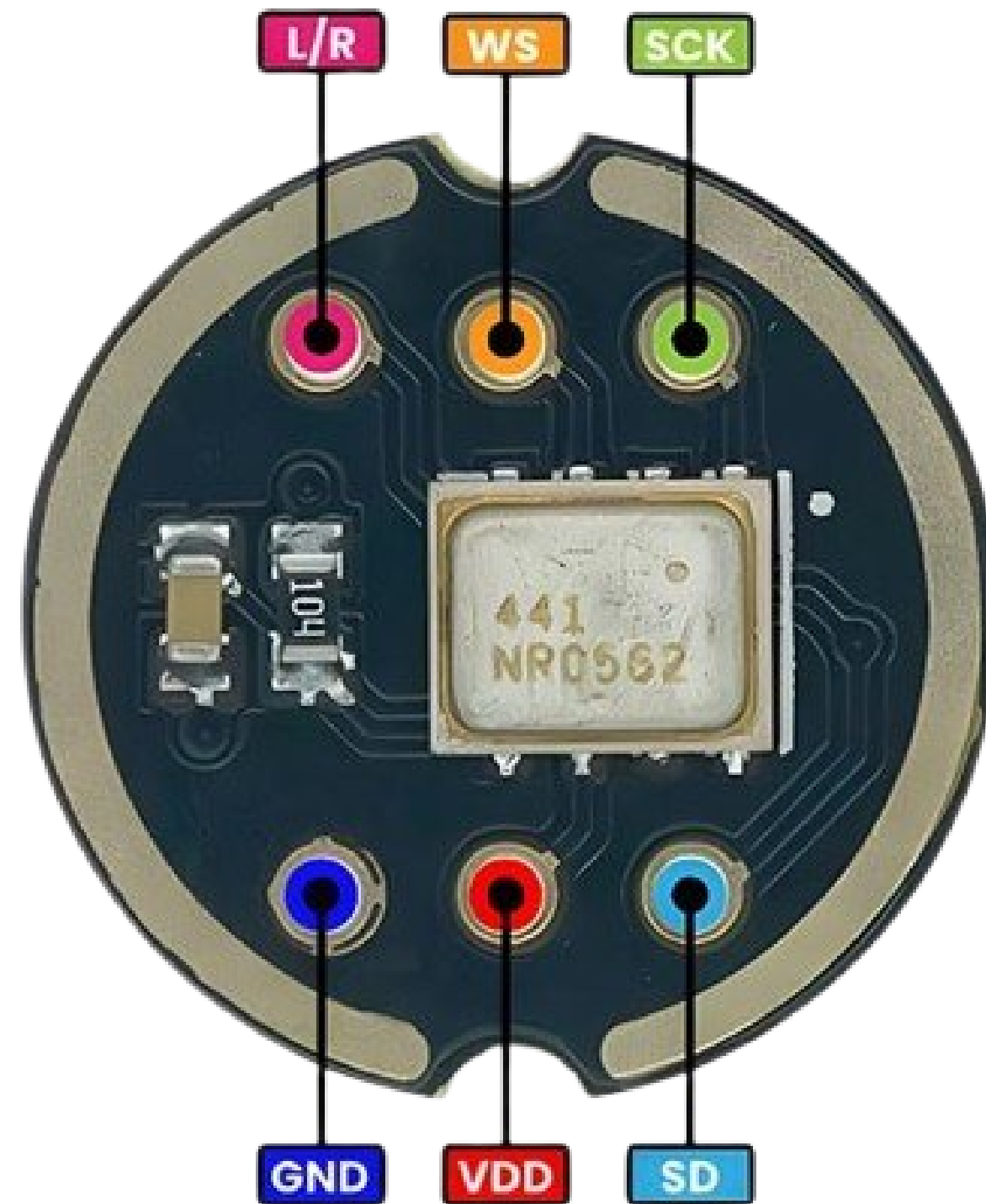
REAL WORLD APPLICATIONS

- **Virtual Assistants:** Voice recognition powers assistants like Alexa, Siri, and Google Assistant, enabling hands-free control of smart devices and access to information.
- **Security:** Voice biometrics are used for authentication in banking and secure systems, allowing user verification through unique vocal traits.
- **Customer Service:** Call centers use voice recognition to route calls and automate responses, enhancing customer support.
- **Translation:** Real-time voice translation helps overcome language barriers in travel, business, and education.
- **Accessibility:** Voice recognition aids individuals with disabilities by enabling hands-free device operation and real-time captioning.

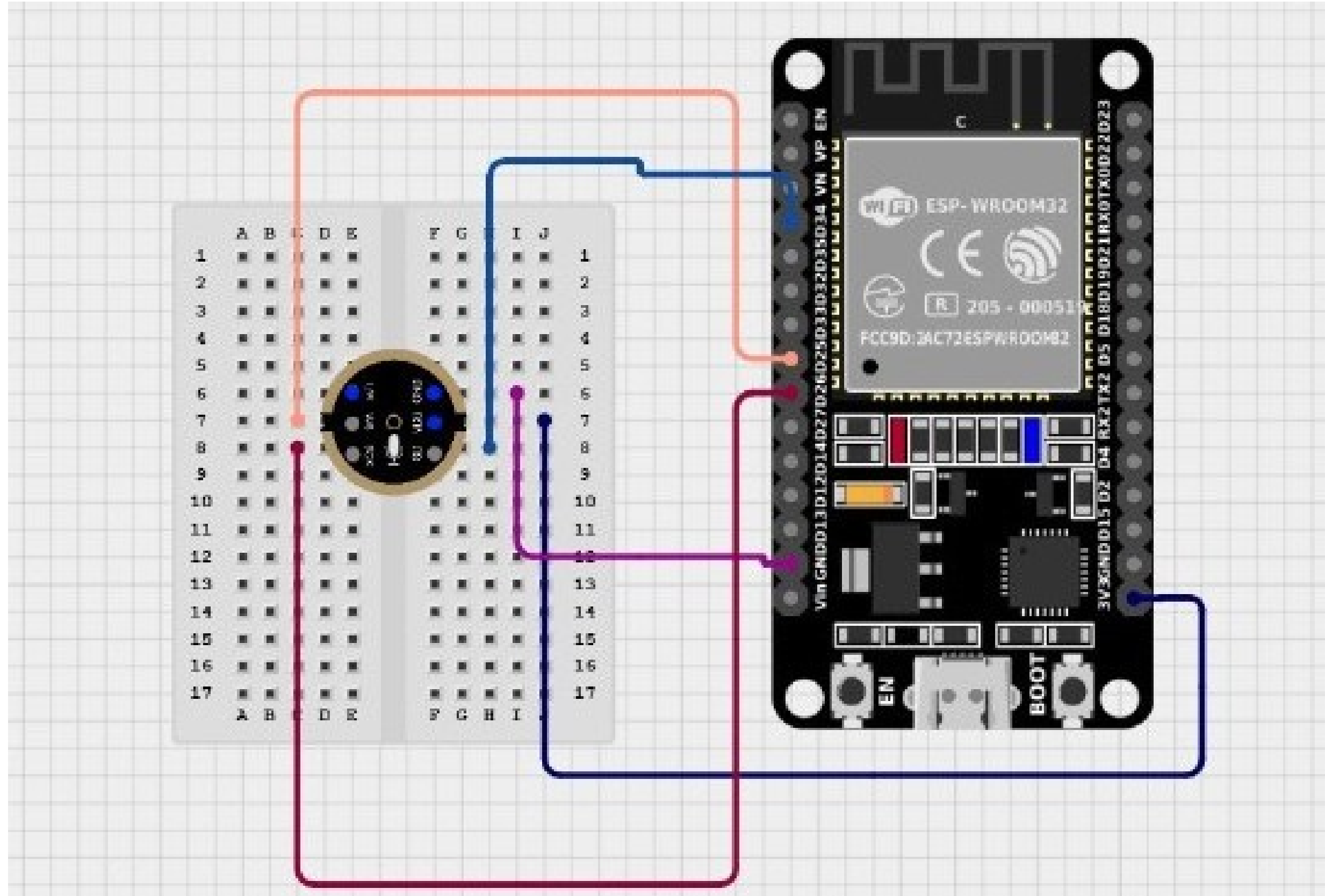
WORKING

The voice recognition project uses the INMP441 digital MEMS microphone to capture sound, converting it into digital data via I2S output. Edge Impulse processes this data with machine learning models to identify voice patterns or commands in real time. This setup enables quick, accurate voice recognition on edge devices with low power use, making it ideal for smart home controls, voice assistants, and other hands-free applications.

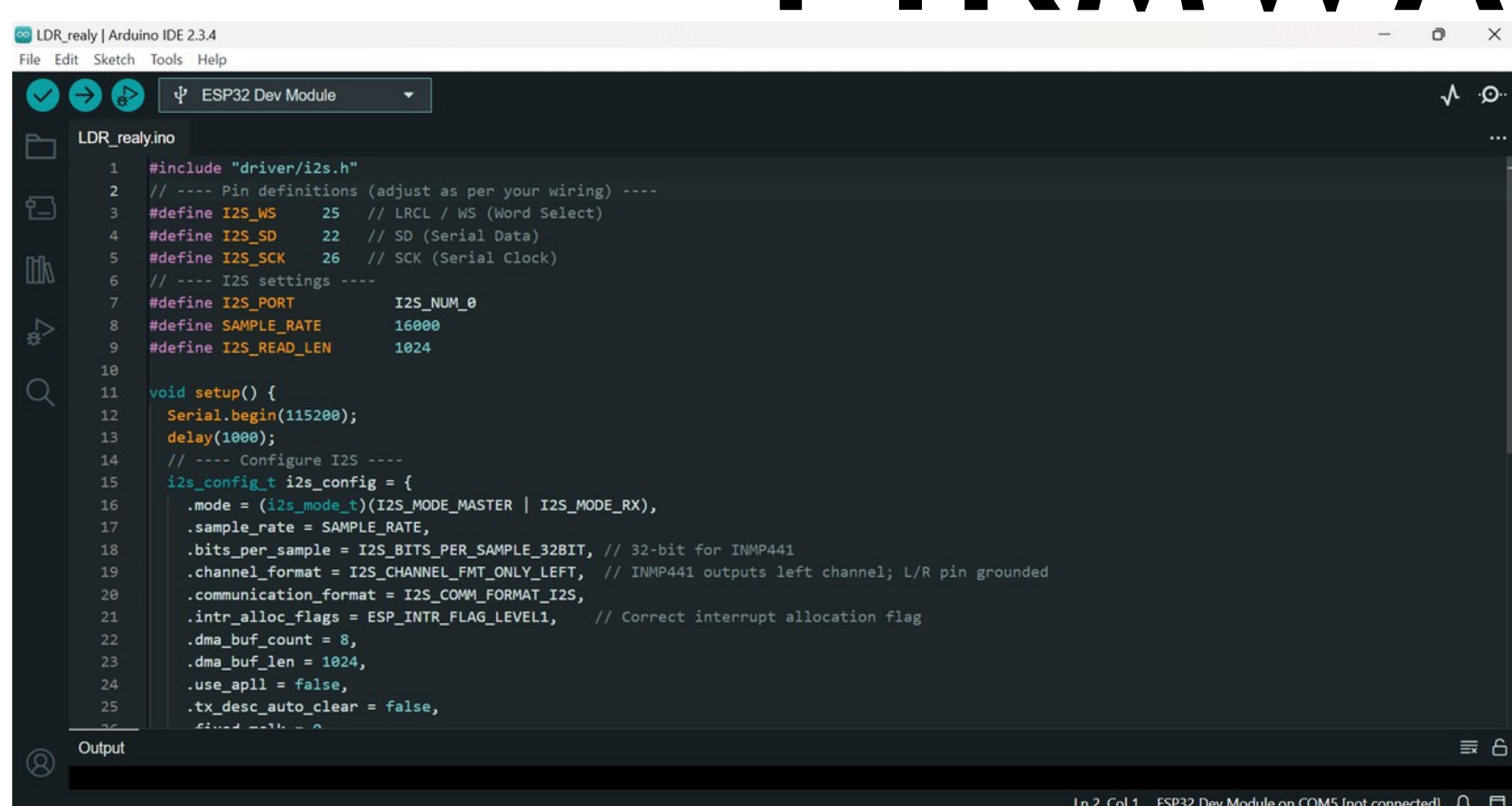
PINOUT



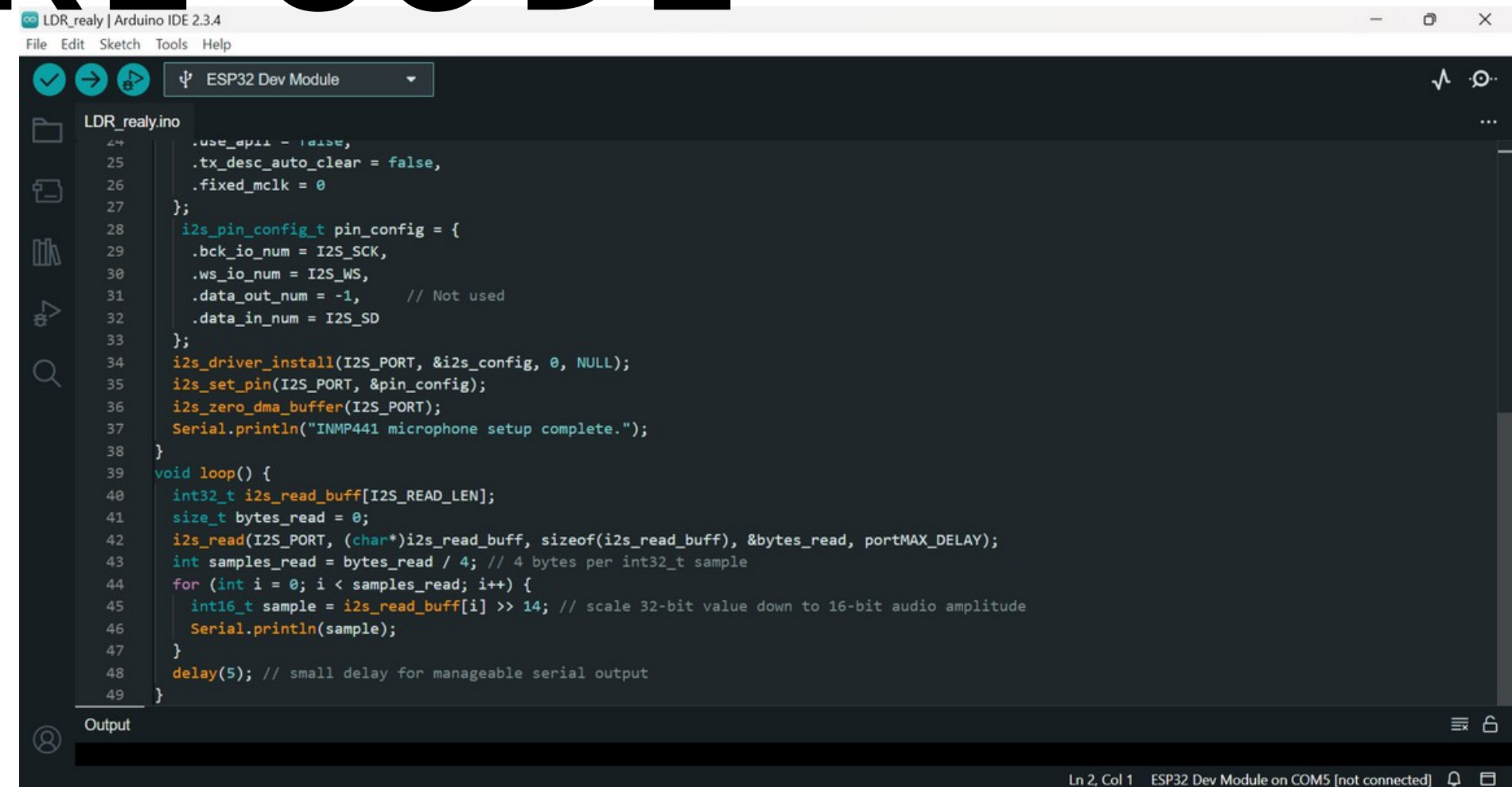
EK NAZAR IDHAR BHI



FIRMWARE CODE



```
LDR_realy.ino
1  #include "driver/i2s.h"
2  // ---- Pin definitions (adjust as per your wiring) ----
3  #define I2S_WS    25 // LRCL / WS (Word Select)
4  #define I2S_SD    22 // SD (Serial Data)
5  #define I2S_SCK   26 // SCK (Serial Clock)
6  // ---- I2S settings ----
7  #define I2S_PORT    I2S_NUM_0
8  #define SAMPLE_RATE 16000
9  #define I2S_READ_LEN 1024
10
11 void setup() {
12     Serial.begin(115200);
13     delay(1000);
14     // ---- Configure I2S ----
15     i2s_config_t i2s_config = {
16         .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
17         .sample_rate = SAMPLE_RATE,
18         .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT, // 32-bit for INMP441
19         .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT, // INMP441 outputs left channel; L/R pin grounded
20         .communication_format = I2S_COMM_FORMAT_I2S,
21         .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1, // Correct interrupt allocation flag
22         .dma_buf_count = 8,
23         .dma_buf_len = 1024,
24         .use_apll = false,
25         .tx_desc_auto_clear = false,
26         .fixed_mclk = 0
27     };
28     i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
29     i2s_set_pin(I2S_PORT, &i2s_config);
30     i2s_zero_dma_buffer(I2S_PORT);
31     Serial.println("INMP441 microphone setup complete.");
32 }
33
34 void loop() {
35     int32_t i2s_read_buff[I2S_READ_LEN];
36     size_t bytes_read = 0;
37     i2s_read(I2S_PORT, (char*)i2s_read_buff, sizeof(i2s_read_buff), &bytes_read, portMAX_DELAY);
38     int samples_read = bytes_read / 4; // 4 bytes per int32_t sample
39     for (int i = 0; i < samples_read; i++) {
40         int16_t sample = i2s_read_buff[i] >> 14; // scale 32-bit value down to 16-bit audio amplitude
41         Serial.println(sample);
42     }
43     delay(5); // small delay for manageable serial output
44 }
```



```
LDR_realy.ino
24  .use_apll = false,
25  .tx_desc_auto_clear = false,
26  .fixed_mclk = 0
27  };
28  i2s_pin_config_t pin_config = {
29      .bck_io_num = I2S_SCK,
30      .ws_io_num = I2S_WS,
31      .data_out_num = -1, // Not used
32      .data_in_num = I2S_SD
33  };
34  i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
35  i2s_set_pin(I2S_PORT, &pin_config);
36  i2s_zero_dma_buffer(I2S_PORT);
37  Serial.println("INMP441 microphone setup complete.");
38  }
39
40 void loop() {
41     int32_t i2s_read_buff[I2S_READ_LEN];
42     size_t bytes_read = 0;
43     i2s_read(I2S_PORT, (char*)i2s_read_buff, sizeof(i2s_read_buff), &bytes_read, portMAX_DELAY);
44     int samples_read = bytes_read / 4; // 4 bytes per int32_t sample
45     for (int i = 0; i < samples_read; i++) {
46         int16_t sample = i2s_read_buff[i] >> 14; // scale 32-bit value down to 16-bit audio amplitude
47         Serial.println(sample);
48     }
49     delay(5); // small delay for manageable serial output
50 }
```

This code is used a firmware to your esp which enable it to send your data through A bridge called Edge impulse CLI. This is already uploaded in your esp..

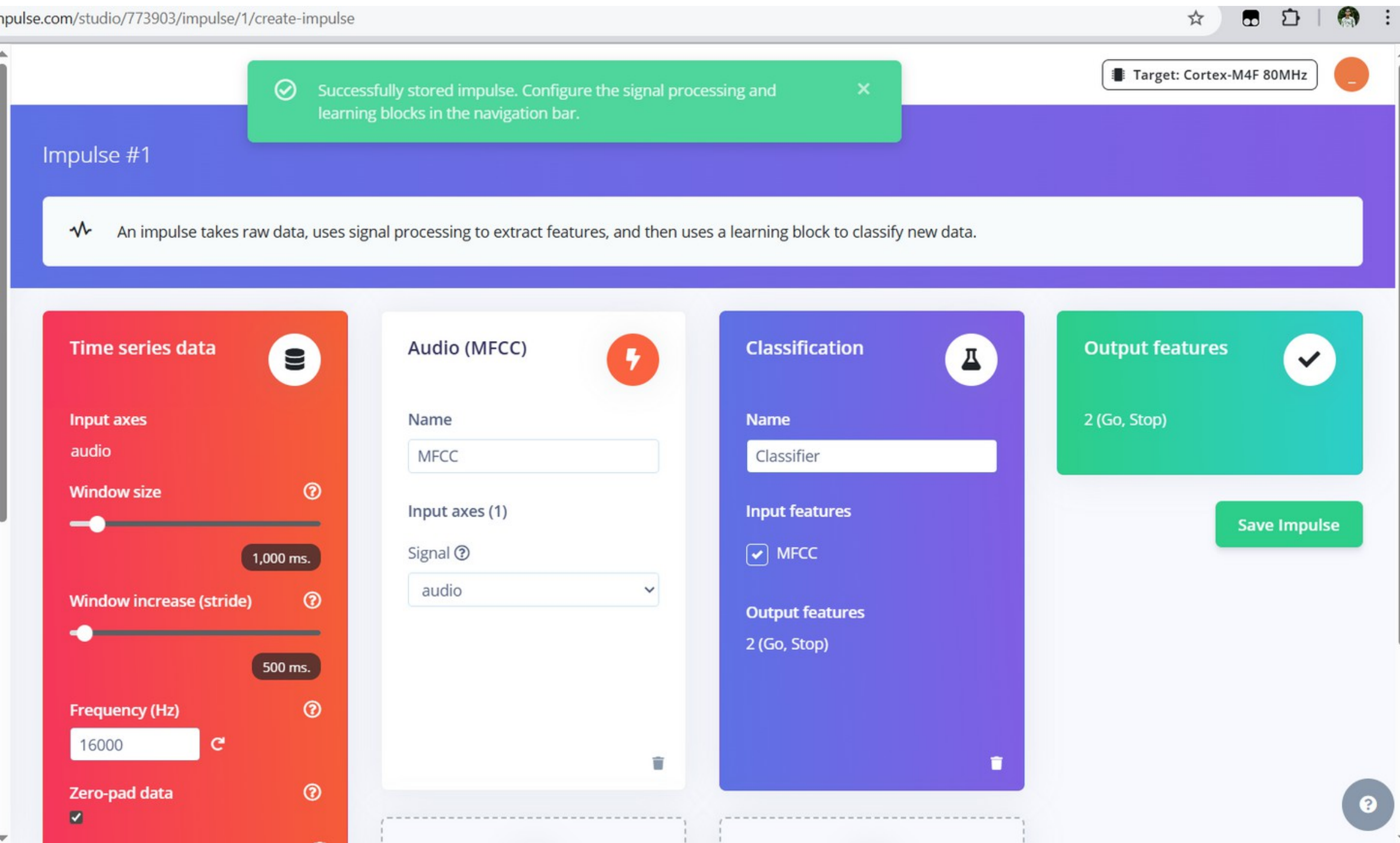
DATA ACQUISITION

The screenshot displays the Edge Impulse Studio web interface. On the left is a sidebar with navigation links: Retrain model, Live classification, Model testing, Perf. calibration, Deployment, Versioning, GETTING STARTED, Documentation, Forums, and Discord. Below these is an 'Upgrade Plan' section. The main area has a top navigation bar with tabs: Dataset, Data explorer, Data sources, Synthetic data, AI labeling (marked 'NEW'), and CSV Wizard. The 'Dataset' tab is active, showing 'DATA COLLECTED 10m 10s' and 'TRAIN / TEST SPLIT 77% / 23%'. Below this is a table of dataset samples. To the right, the 'Collect data' panel shows settings for Device (No devices connected), Label (Label name), Sample length (5000 ms), Sensor, and Frequency, with a 'Start sampling' button. At the bottom right, a 'RAW DATA' section shows the selected sample 'Stop.63rb49jn'.

SAMPLE NAME	LABEL	ADDED	LENGTH
Stop.63rb49jn	Stop	Today, 05:25:00	10s
Stop.63rb31s4	Stop	Today, 05:24:20	10s
Stop.63rb2iqh	Stop	Today, 05:24:04	10s
Stop.63rb26ds	Stop	Today, 05:23:51	10s
Stop.63rb1pvf	Stop	Today, 05:23:39	10s
Stop.63rb1dc7	Stop	Today, 05:23:26	10s
Stop.63ravpab	Stop	Today, 05:22:32	10s

We will use premade dataset for
'Data Acquisition'.

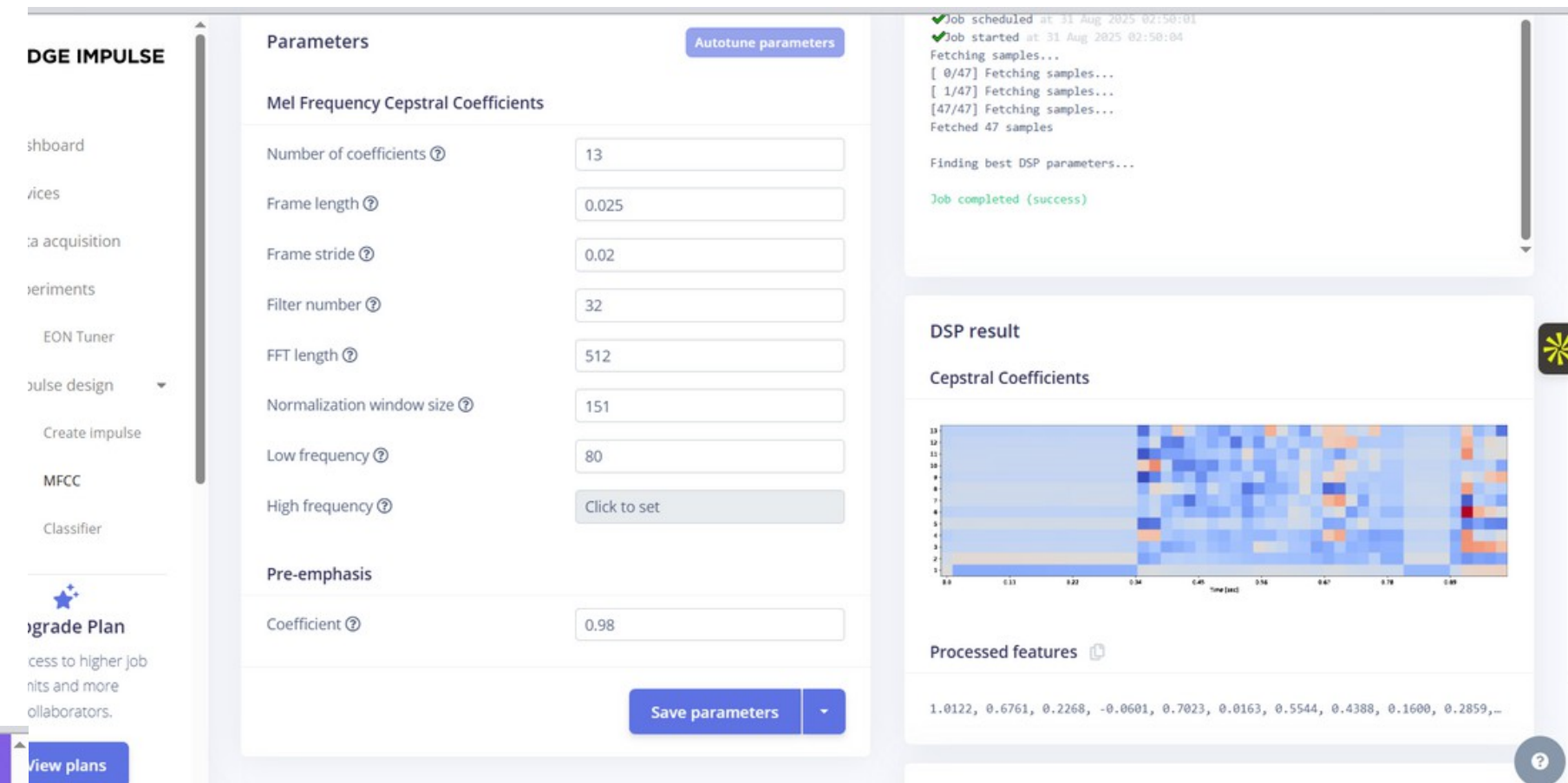
DESIGNING AN IMPULSE



Go to the **Create impulse** tab, add a **Time series data**, an **Audio (MFCC)** and a **Classification (Keras)** block. Leave the window size to 1second (as that's the length of our audio samples in the dataset) and click **Save Impulse**.

CONFIGURE THE MFCC BLOCK

Now click on the MFCC tab in the left hand navigation menu. You just click on the autotune parameters and then save parameters.

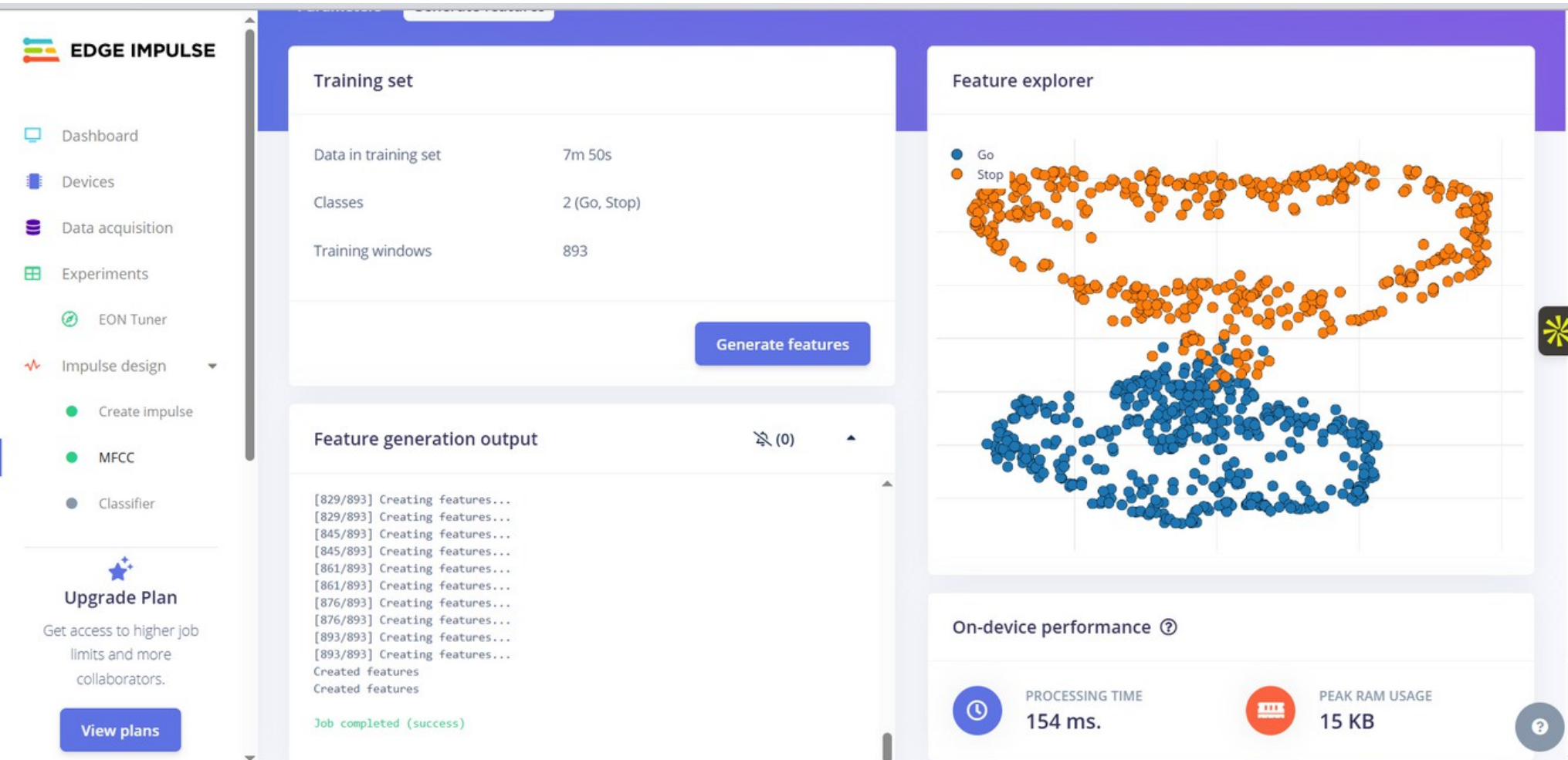


The screenshot shows the 'DGE IMPULSE' interface with the 'MFCC' tab selected in the left navigation menu. The main panel is titled 'Parameters' and contains a section for 'Mel Frequency Cepstral Coefficients'. The parameters are as follows:

Parameter	Value
Number of coefficients	13
Frame length	0.025
Frame stride	0.02
Filter number	32
FFT length	512
Normalization window size	151
Low frequency	80
High frequency	Click to set
Pre-emphasis Coefficient	0.98

Buttons for 'Autotune parameters' and 'Save parameters' are visible. On the right, a 'DSP result' section shows a 'Cepstral Coefficients' heatmap and a list of 'Processed features'.

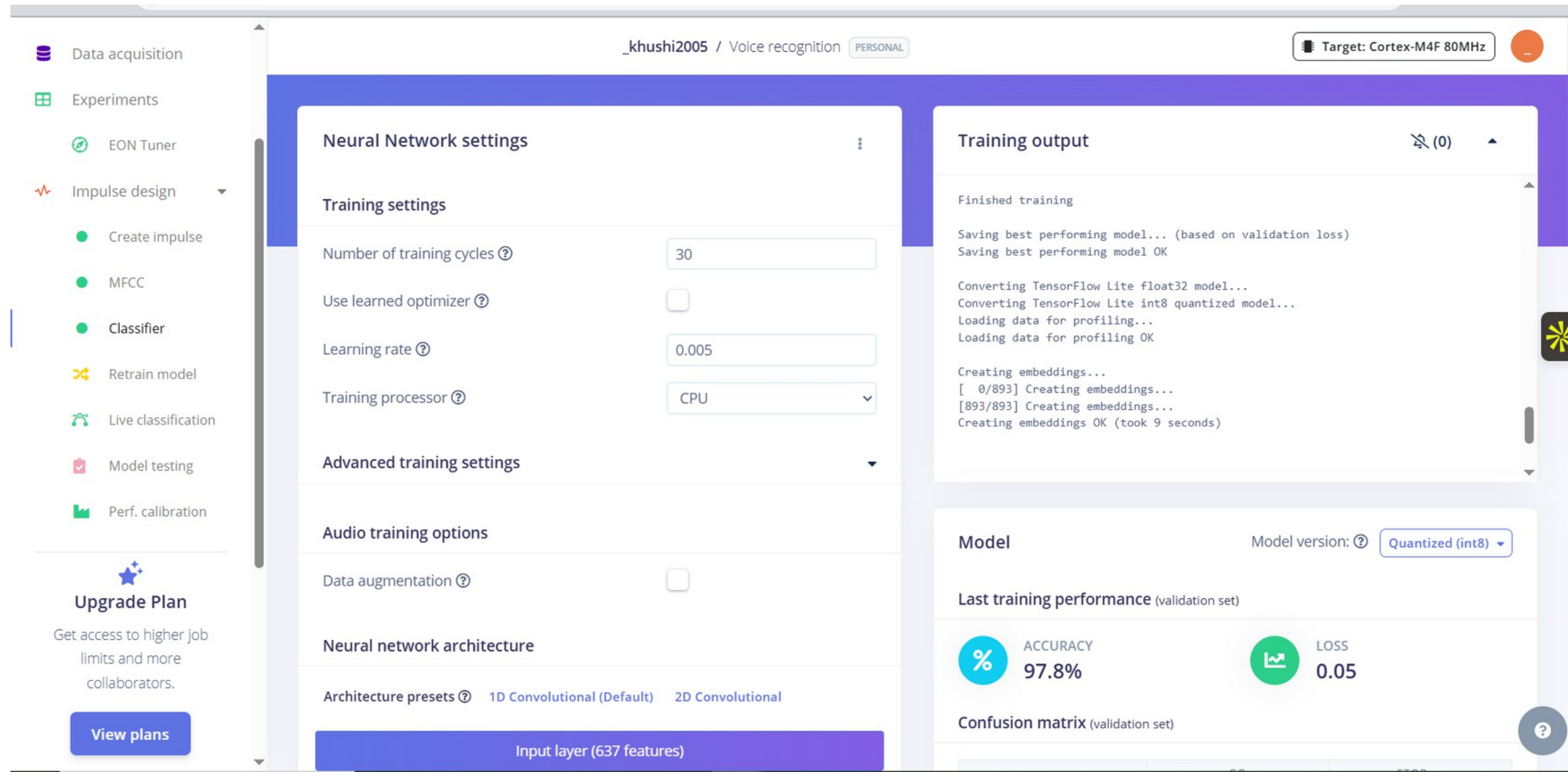
You will be redirected to feature generation window. Click on generate features and then you can analyse the graph.



The screenshot shows the 'EDGE IMPULSE' interface with the 'MFCC' block selected in the left navigation menu. The main panel is titled 'Training set' and contains a 'Generate features' button. Below this, the 'Feature generation output' section shows a list of features being created. The 'Feature explorer' section displays a scatter plot of the generated features, with orange dots for 'Go' and blue dots for 'Stop'. The 'On-device performance' section shows the following metrics:

Metric	Value
PROCESSING TIME	154 ms.
PEAK RAM USAGE	15 KB

CONFIGURE THE NEURAL NETWORK



Click on **Classifier** in the left hand menu.
Change the number of training cycles to 50 and
train the model.

CLASSIFYING TEST DATA



Add Test Data in Data Acquisition window if not present already. To run your model against the test set, head to **Model testing**, select all items and click **Classify** selected. Click the three dots (⋮) next to a sample and select Show classification.

The screenshot shows the JLSE web interface for a 'Voice recognition' project. The 'Test data' table lists samples with their expected outcomes and lengths. A 'Classify all' button is visible. The 'Model testing output' section is currently empty.

SAMPLE NA...	EXPECTED OUT...	LENG...	ACCURACY	RESULT
Stop.63rb3...	Stop	10s		⋮
Stop.63rb3...	Stop	10s		⋮
Stop.63rb1...	Stop	10s		⋮
Stop.63rb0...	Stop	10s		⋮
Stop.63rb0...	Stop	10s		⋮
Stop.63rap...	Stop	10s		⋮
Stop.63ral...	Stop	10s		⋮

DEVELOPMENT

Add the given deployment code in your Arduino IDE and then start running your model. As you will say something the model will predict it and will display in the serial monitor.



DEPLOYMENT CODE

```
LDR_realy | Arduino IDE 2.3.4
File Edit Sketch Tools Help

ESP32 Dev Module

LDR_realy.ino
1 #include <driver/i2s.h>
2 #include <Arduino.h>
3
4 // I2S Microphone config
5 #define I2S_SAMPLE_RATE 16000
6 #define I2S_SAMPLE_BITS 32
7 #define I2S_READ_LEN 2048
8 #define I2S_MIC_CHANNEL I2S_CHANNEL_FMT_ONLY_LEFT
9 #define I2S_FORMAT I2S_COMM_FORMAT_I2S
10 #define I2S_PORT I2S_NUM_0
11
12 #define PIN_I2S_WS 25
13 #define PIN_I2S_SD 32
14 #define PIN_I2S_SCK 26
15
16 // Buffer for audio samples
17 static int32_t i2s_read_buffer[I2S_READ_LEN];
18
19 // Setup I2S
20 void i2s_init() {
21     i2s_config_t i2s_config = {
22         .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
23         .sample_rate = I2S_SAMPLE_RATE,
24         .bits_per_sample = I2S_SAMPLE_BITS,
25         .channel_format = I2S_MIC_CHANNEL,
26         .communication_format = I2S_FORMAT,
27         .i2s_port = I2S_PORT,
28         .use_apll = false,
29         .dma_buf_count = 4,
30         .dma_buf_len = 512,
31         .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
32     };
33     i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
34     i2s_set_pin(I2S_PORT, &pin_config);
35     i2s_zero_dma_buffer(I2S_PORT);
36 }
37
38 // Read samples from I2S and convert to int16_t
39 int i2s_read(int16_t *samples, size_t len) {
40     size_t bytes_read = 0;
41     i2s_read(I2S_PORT, (void*)i2s_read_buffer, len * sizeof(int32_t), &bytes_read, portMAX_DELAY);
42     int samples_read = bytes_read / sizeof(int32_t);
43     for (int i = 0; i < samples_read; i++) {
44         samples[i] = (int16_t)(i2s_read_buffer[i] >> 14); // shift 32-bit to 16-bit
45     }
46     return samples_read;
47 }
48
49 void setup() {
50     Serial.begin(115200);
51     Serial.println("Edge Impulse ESP32 Voice Recognition");
52     i2s_init();
53 }
54
55 void loop() {
56     // Buffer for model input
57     static int16_t audio_buffer[EI_CLASSIFIER_SLICE_SIZE];
58
59     // Fill buffer with microphone samples
60     int samples_collected = i2s_read(audio_buffer, EI_CLASSIFIER_SLICE_SIZE);
61
62     if (samples_collected == EI_CLASSIFIER_SLICE_SIZE) {
63         // Wrap into Edge Impulse signal
64         signal_t signal;
65         numpy::signal_from_buffer(audio_buffer, EI_CLASSIFIER_SLICE_SIZE, &signal);
66     }
67 }
```

```
LDR_realy | Arduino IDE 2.3.4
File Edit Sketch Tools Help

ESP32 Dev Module

LDR_realy.ino
25     .channel_format = I2S_MIC_CHANNEL,
26     .communication_format = I2S_FORMAT,
27     .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
28     .dma_buf_count = 4,
29     .dma_buf_len = 512,
30     .use_apll = false
31 };
32
33 i2s_pin_config_t pin_config = {
34     .bck_io_num = PIN_I2S_SCK,
35     .ws_io_num = PIN_I2S_WS,
36     .data_out_num = I2S_PIN_NO_CHANGE,
37     .data_in_num = PIN_I2S_SD
38 };
39
40 i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
41 i2s_set_pin(I2S_PORT, &pin_config);
42 i2s_zero_dma_buffer(I2S_PORT);
43 }
44
45 // Read samples from I2S and convert to int16_t
46 int i2s_read(int16_t *samples, size_t len) {
47     size_t bytes_read = 0;
48     i2s_read(I2S_PORT, (void*)i2s_read_buffer, len * sizeof(int32_t), &bytes_read, portMAX_DELAY);
49     int samples_read = bytes_read / sizeof(int32_t);
50     for (int i = 0; i < samples_read; i++) {
51         samples[i] = (int16_t)(i2s_read_buffer[i] >> 14); // shift 32-bit to 16-bit
52     }
53     return samples_read;
54 }
55
56 void setup() {
57     Serial.begin(115200);
58     Serial.println("Edge Impulse ESP32 Voice Recognition");
59     i2s_init();
60 }
61
62 void loop() {
63     // Buffer for model input
64     static int16_t audio_buffer[EI_CLASSIFIER_SLICE_SIZE];
65
66     // Fill buffer with microphone samples
67     int samples_collected = i2s_read(audio_buffer, EI_CLASSIFIER_SLICE_SIZE);
68
69     if (samples_collected == EI_CLASSIFIER_SLICE_SIZE) {
70         // Wrap into Edge Impulse signal
71         signal_t signal;
72         numpy::signal_from_buffer(audio_buffer, EI_CLASSIFIER_SLICE_SIZE, &signal);
73     }
74 }
```

```
LDR_realy | Arduino IDE 2.3.4
File Edit Sketch Tools Help

ESP32 Dev Module

LDR_realy.ino
50     int samples_read = bytes_read / sizeof(int32_t);
51     for (int i = 0; i < samples_read; i++) {
52         samples[i] = (int16_t)(i2s_read_buffer[i] >> 14); // shift 32-bit to 16-bit
53     }
54     return samples_read;
55 }
56
57 void setup() {
58     Serial.begin(115200);
59     Serial.println("Edge Impulse ESP32 Voice Recognition");
60     i2s_init();
61 }
62
63 void loop() {
64     // Buffer for model input
65     static int16_t audio_buffer[EI_CLASSIFIER_SLICE_SIZE];
66
67     // Fill buffer with microphone samples
68     int samples_collected = i2s_read(audio_buffer, EI_CLASSIFIER_SLICE_SIZE);
69
70     if (samples_collected == EI_CLASSIFIER_SLICE_SIZE) {
71         // Wrap into Edge Impulse signal
72         signal_t signal;
73         numpy::signal_from_buffer(audio_buffer, EI_CLASSIFIER_SLICE_SIZE, &signal);
74     }
75 }
```

```
LDR_realy | Arduino IDE 2.3.4
File Edit Sketch Tools Help

ESP32 Dev Module

LDR_realy.ino
68 // Fill buffer with microphone samples
69 int samples_collected = i2s_read(audio_buffer, EI_CLASSIFIER_SLICE_SIZE);
70
71 if (samples_collected == EI_CLASSIFIER_SLICE_SIZE) {
72     // Wrap into Edge Impulse signal
73     signal_t signal;
74     numpy::signal_from_buffer(audio_buffer, EI_CLASSIFIER_SLICE_SIZE, &signal);
75
76     // Run classifier
77     ei_impulse_result_t result;
78     EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);
79
80     if (res != EI_IMPULSE_OK) {
81         Serial.printf("ERROR: Classifier failed (%d)\n", res);
82         return;
83     }
84
85     // Print predictions
86     Serial.println("Predictions:");
87     for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
88         Serial.printf("  %s: %.5f\n", result.classification[ix].label, result.classification[ix].value);
89     }
90     Serial.println();
91 }
92 }
```




THANK YOU
