

# **GESTURE RECOGNITION**

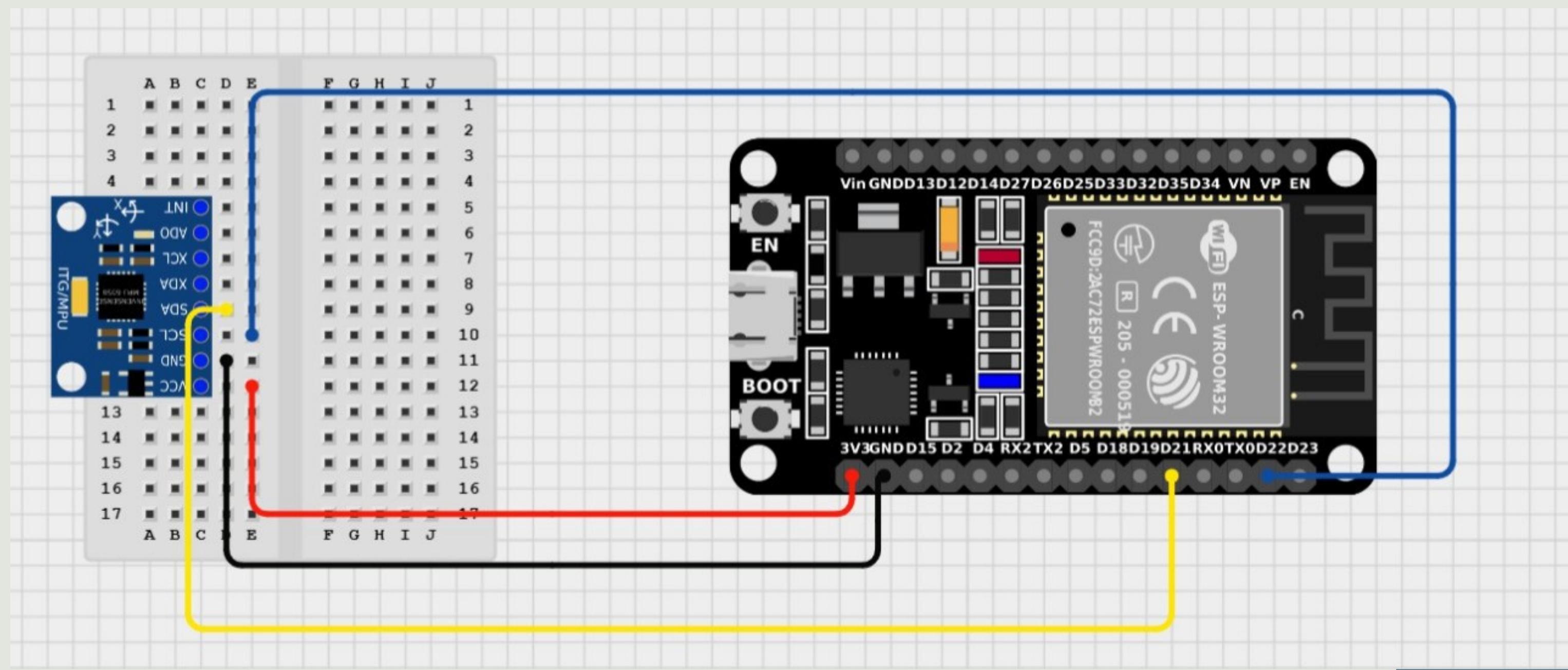
# REAL WORLD APPLICATIONS

- **Human-Computer Interaction (HCI)**: Control devices like smart TVs, gaming consoles, and VR systems using natural hand gestures without physical controllers.
- **Retail and Marketing**: Analyze customer behavior in stores by tracking gestures to optimize layout and product placement.
- **Virtual and Augmented Reality**: Enable immersive interactions within virtual spaces through hand and body motion tracking.
- **Robot Control**: Gesture commands help robots perform tasks and interact with humans in industries and assistive services.
- **Sign Language Recognition**: Translate sign languages in real-time, aiding communication for the deaf and mute.

# WORKING OF THE PROJECT

- Connect the MPU6050 sensor to the ESP32 and use Edge Impulse's data forwarder to send gesture data during motion.
- Label the collected data in Edge Impulse Studio by categorizing each sample according to its gesture.
- Design and train a model in Edge Impulse Studio using the labeled data with motion-optimized processing and classification blocks.
- Download the trained model and deploy it to Arduino IDE to enable real-time processing of live sensor data.
- The device now instantly detects and classifies hand gestures using new data from the MPU6050 sensor.

# PEHLE CONNECTION TOH DEKHLO!!



# FIRMWARE CODE

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..

```
sketch_feb2a.ino
 6
 7 // 1. Include MPU6050 libraries
 8 #include <Adafruit_MPU6050.h>
 9 #include <Adafruit_Sensor.h>
10 #include <Wire.h>
11
12 // 2. Set the sampling frequency (Hz)
13 // Common frequencies for gesture recognition are 62.5Hz or 100Hz
14 #define FREQUENCY_HZ      100
15 #define INTERVAL_MS        (1000 / FREQUENCY_HZ)
16
17 // Globals for sensor and timing
18 Adafruit_MPU6050 mpu;
19 unsigned long last_interval_ms = 0;
20
21 void setup() {
22     Serial.begin(115200);
23
24     // Initialize the MPU6050
25     if (!mpu.begin()) {
26         Serial.println("Failed to find MPU6050 chip");
27         while (1) {
28             delay(10);
29         }
30     }
31
32     // Set a reasonable accelerometer range
33     mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
34
35     Serial.println("MPU6050 Found! Ready to collect data.");
36 }
37
38 void loop() {
39
40 }
```

```
sketch_feb2a.ino
 23
 24 // Initialize the MPU6050
 25 if (!mpu.begin()) {
 26     Serial.println("Failed to find MPU6050 chip");
 27     while (1) {
 28         delay(10);
 29     }
30 }
31
32 // Set a reasonable accelerometer range
33 mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
34
35 Serial.println("MPU6050 Found! Ready to collect data.");
36 }
37
38 void loop() {
39     // Ensure we are sampling at the correct frequency
40     if (millis() < last_interval_ms + INTERVAL_MS) {
41         return;
42     }
43     last_interval_ms = millis();
44
45     // Get a new sensor event
46     sensors_event_t a, g, temp;
47     mpu.getEvent(&a, &g, &temp);
48
49     // Print the accelerometer data in the required format (CSV)
50     Serial.print(a.acceleration.x);
51     Serial.print(',');
52     Serial.print(a.acceleration.y);
53     Serial.print(',');
54     Serial.println(a.acceleration.z);
55 }
```

# CLI INSTALLATION

CONNECT THE ESP SETUP TO YOUR COMPUTER AND  
OPEN THE COMMAND PROMPT. THEN ENTER THE  
COMMAND

NPM INSTALL -G EDGE-IMPULSE-CLI --FORCE

THIS WILL SUCCESSFULLY INSTALL THE EDGE IMPULSE  
CLI IN YOUR COMPUTER WHICH WILL HELP US TO  
FORWARD DATA TO EDGE IMPULSE

# ENABLING DATA FORWARDER

NOW IN YOUR EDGE IMPULSE CREATE A NEW PROJECT. THEN OPEN THE COMMAND PROMPT AND  
ENTER  
**EDGE-IMPULSE-DATA-FORWARDER**

```
C:\Windows\system32\cmd.exe - "node" "C:\Users\admin\AppData\Roaming\npm\node_modules\edge-impulse-cli\build\cli\data-forwarder.js"
Microsoft Windows [Version 10.0.19045.6216]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>edge-impulse-data-forwarder
Edge Impulse data forwarder v1.34.0
? What is your user name or e-mail address (edgeimpulse.com)? _khushi2005
? What is your password? [hidden]
Endpoints:
  Websocket: wss://remote-mgmt.edgeimpulse.com
  API: https://studio.edgeimpulse.com
  Ingestion: https://ingestion.edgeimpulse.com

? Which device do you want to connect to? (press type to search) COM9
[SER] Connecting to COM9
[SER] Serial is connected (00:01)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

? To which project do you want to connect this device? (press type to search) 769868
[SER] Detecting data frequency...
[SER] Detected data frequency: 100Hz
? 3 sensor axes detected (example values: [0.08,-0.17,10.44]). What do you want to call them? Separate the names with ',': x-axis,y-axis,z-axis
[WS ] Device "esp32" is now connected to project "_khushi2005-project-1". To connect to another project, run `edge-impulse-data-forwarder --clean
[WS ] Go to https://studio.edgeimpulse.com/studio/769868/acquisition/training to build your machine learning model!
```

ENTER YOUR  
ID, PASSWORD, NEW  
PROJECT THAT YOU ARE  
WORKING ON AND IF  
ASKED THEN THE COM  
PORT IN WHICH ESP IS  
CONNECTED

# CONNECTED DEVICE IN EDGE IMPULSE

OPEN YOUR EDGE IMPULSE AND GO TO THE DEVICES OPTION. THERE YOU WILL SEE YOUR DEVICE NAME AND A GREEN DOT WITH IT

The screenshot shows the Edge Impulse web interface. On the left, there is a sidebar with the following menu items:

- Dashboard
- Devices** (selected)
- Data acquisition
- Experiments
- EON Tuner
- Impulse design
- Create impulse
- Spectral features
- Classifier

The main content area has a header with the user's name, project name, and target settings:

\_khushi2005 / \_khushi2005-project-1 PERSONAL  
Target: Cortex-M4F 80MHz

## Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

DEVICE	ID	TYPE
esp32	00:01	DATA_FORWARDER

Details for the esp32 device:  
Connected to data acquisition (Sensor with 3 axes (x-axis, y-axis, z-axis))

At the bottom, there is a copyright notice: © 2025 EdgImpulse Inc. All rights reserved.

# DATA ACQUISITION

GO TO THE DATA ACQUISITION OPTION. IN THE COLLECT DATA WINDOW, YOUR DEVICE NAME WILL BE SHOWN AUTOMATICALLY. SET THE SAMPLE LENGTH AS 10000 AND ENTER A LABEL FOR ANY GESTURE AND THEN PERFORM THAT GESTURE TO RECORD THE DATA.

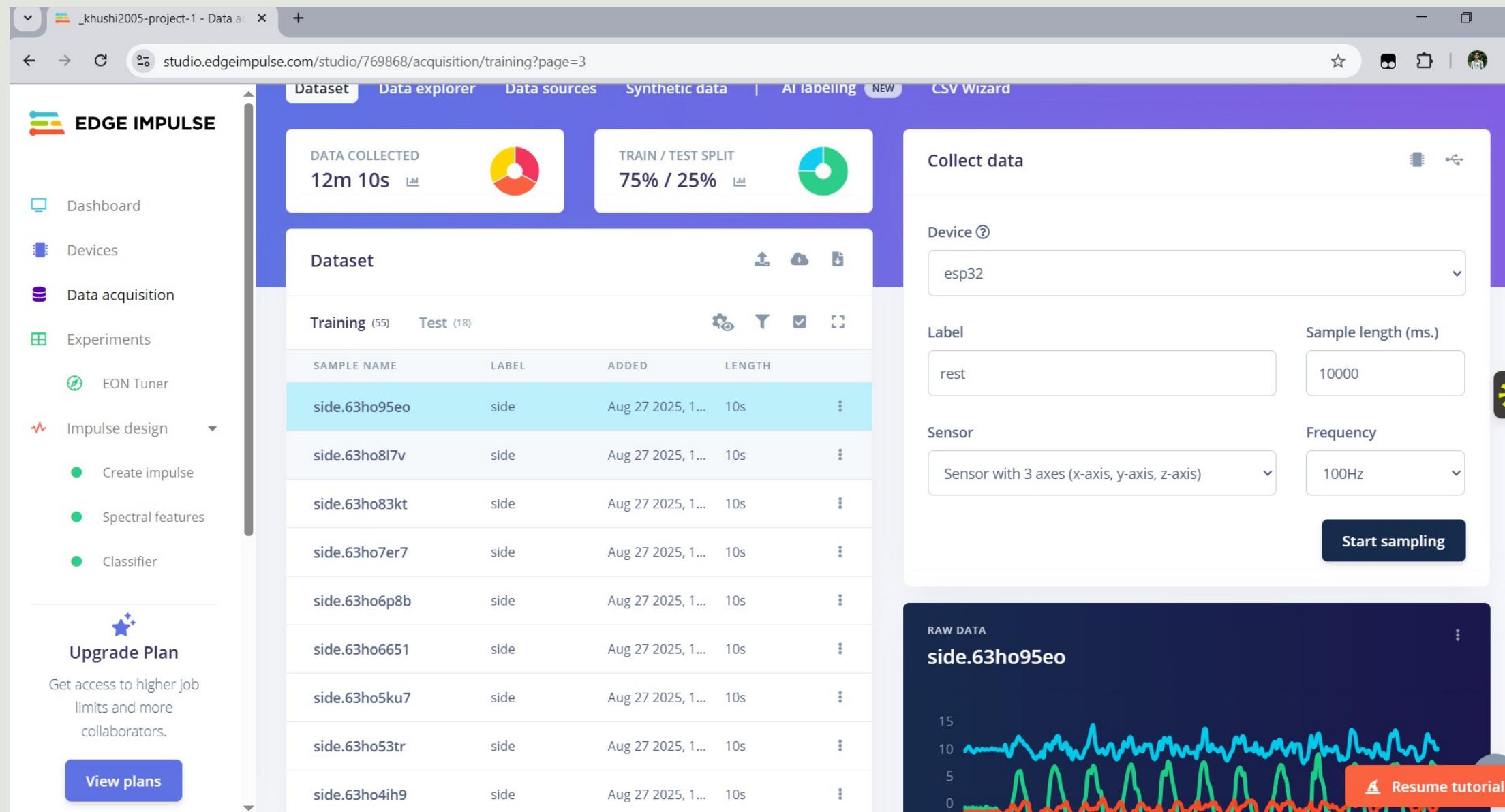
The screenshot shows a software interface for data acquisition. On the left, there is a sidebar with the following menu items:

- Dashboard
- Devices
- Data acquisition
- Experiments
- EON Tuner
- Impulse design
  - Create impulse
  - Spectral features
  - Classifier
- Upgrade Plan
  - Get access to higher job limits and more collaborators.
  - [View plans](#)

The main area has tabs at the top: Dataset, Data explorer, Data sources, Synthetic data, AI labeling (NEW), and CSV Wizard. The Dataset tab is selected. It displays two summary boxes: "DATA COLLECTED 12m 10s" with a pie chart icon, and "TRAIN / TEST SPLIT 75% / 25%" with a pie chart icon. Below these is a "Dataset" section with tables for "Training" (55 samples) and "Test" (18 samples). The "Training" table includes columns for SAMPLE NAME, LABEL, ADDED, and LENGTH. Sample names listed include rest.63hp0deq, rest.63hovtku, rest.63hovc5l, rest.63hoqcef, rest.63hopt28, rest.63hopcbj, and rest.63hoorre. All samples are labeled "rest".  
The "Test" table includes columns for SAMPLE NAME, LABEL, ADDED, and LENGTH. Sample names listed include rest.63hp0deq, rest.63hovtku, rest.63hovc5l, rest.63hoqcef, rest.63hopt28, rest.63hopcbj, and rest.63hoorre. All samples are labeled "rest".  
On the right, the "Collect data" window is open. It contains fields for "Device" (set to esp32), "Label" (set to rest), "Sample length (ms.)" (set to 10000), "Sensor" (set to "Sensor with 3 axes (x-axis, y-axis, z-axis)"), and "Frequency" (set to 100Hz). A large blue oval highlights the "Start sampling" button at the bottom right of this window. At the bottom of the main area, there is a dark bar with the text "RAW DATA" and "Click on a sample to load...".

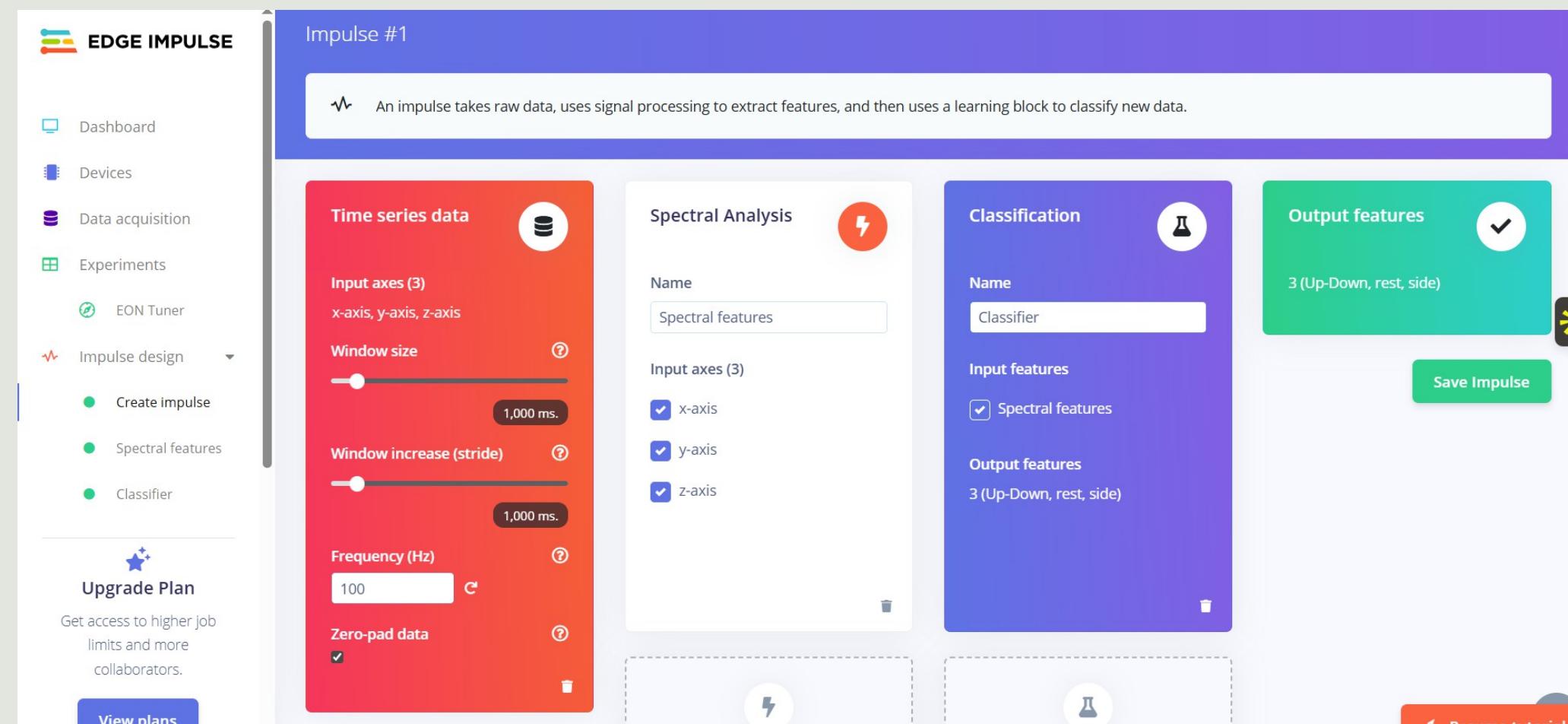
# DATA ACQUISITION

PROPERLY LABEL ALL THE DATA AND CREATE ATLEAST THREE CLASSES SUCH AS UP-DOWN, SIDE AND REST. ENSURE THAT MINIMUM 3 MINUTES OF DATA IS PRESENT IN EACH CLASS. ALSO ENTER SOME DATA IN THE TEST WINDOW. THE DATA SPLIT FOR EACH CLASS SHOULD BE MINIMUM 80/20 FOR TRAIN/TEST SPLIT



# CREATING IMPULSE BLOCK

AFTER ALL THE DATA IS COLLECTED,CLICK ON THE CREATE NEW IMPULSE OPTION. CLICK ON THE ADD A PROCESSING BLOCK AND ADD ‘SPECTRAL FEATURES’ AS THE PROCESSING BLOCK. THEN CLICK ON ADD LEARNING BLOCK AND ADD ‘CLASSIFIER’ AS THE LEARNING BLOCK. THEN JUST SAVE THE IMPULSE



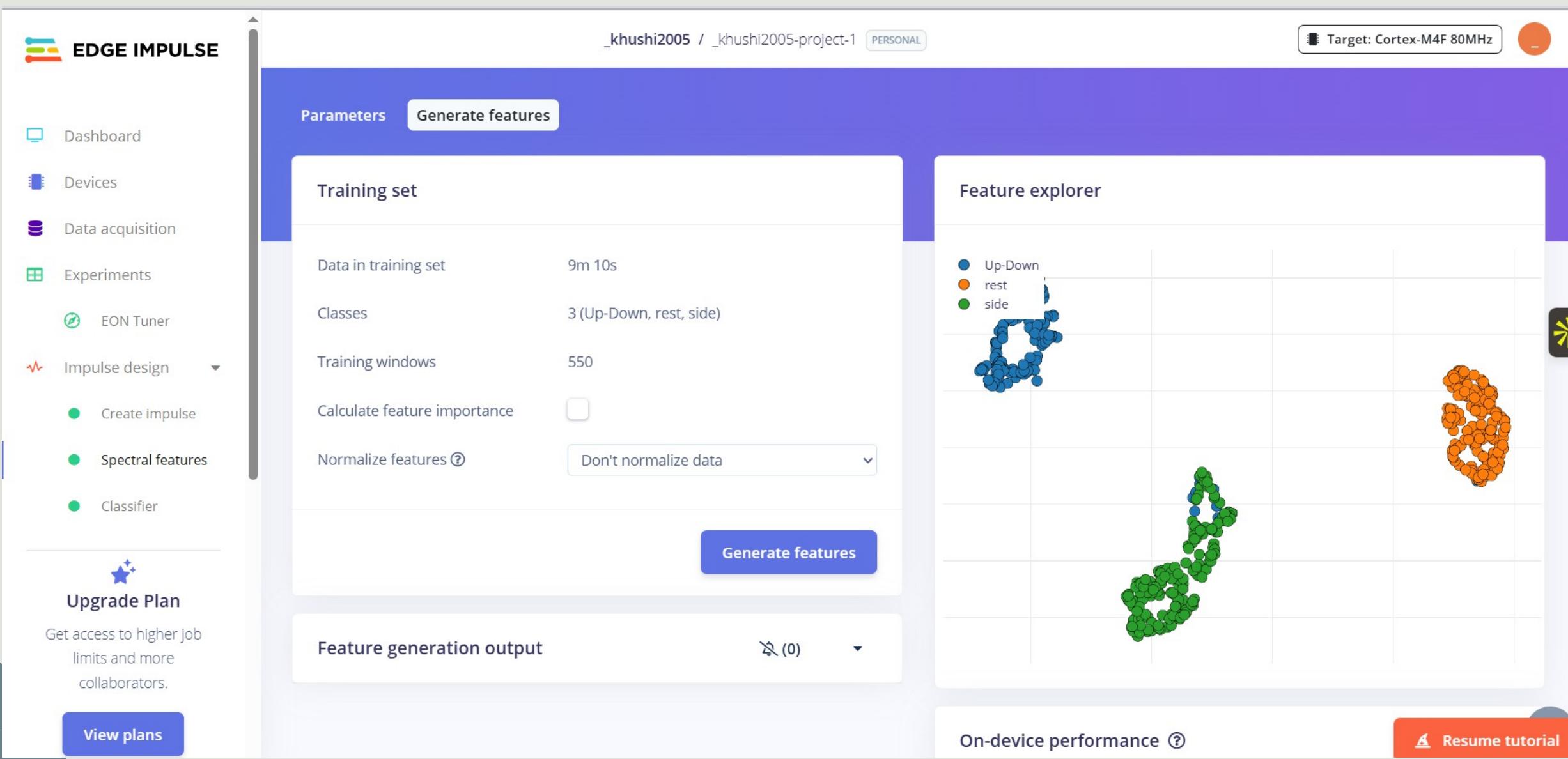
# SPECTRAL ANALYSIS AND FEATURE GENERATION

CLICK ON THE SPECTRAL FEATURES OPTION AND CLICK ON THE AUTOTUNE PARAMETERS. AFTER THE PARAMETERS ARE SET, JUST SAVE THE PARAMETERS.

The screenshot shows the Edge Impulse software interface. On the left, a sidebar lists navigation options: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse design (with sub-options Create impulse, Spectral features, and Classifier), and Upgrade Plan. The main area is titled "Parameters". It contains two sections: "Filter" and "Analysis". In the "Filter" section, there are fields for Scale axes (0.042301185525321264), Input decimation ratio (3), Type (low), Cut-off frequency (13.28125), and Order (6). In the "Analysis" section, there are fields for Type (FFT), FFT length (32), and checkboxes for Take log of spectrum? (checked), Overlap FFT frames? (unchecked), and Improve low frequency resolution? (unchecked). To the right of the parameters are three plots: "After filter" (a spectrogram showing a red curve peaking at ~13.28 Hz), "Value" (a line graph showing a signal fluctuating around zero), and "Spectral power (log)" (a line graph showing energy levels across a frequency range from 0.00 to 12.50 Hz). A blue oval highlights the "Autotune parameters" button in the top right of the parameter section. Another blue oval highlights the "Save parameters" button at the bottom right of the parameter section. A small yellow asterisk icon is located on the far right edge of the interface.

# SPECTRAL ANALYSIS AND FEATURE GENERATION

YOU WILL BE REDIRECTED TO THE FEATURE GENERATION WINDOW WHERE YOU WILL JUST VERIFY YOUR DATA PARAMETERS AND GENERATE THE FEATURES OF YOUR DATA.



# TRAINING OF THE MODEL

GO TO THE TRANSFER LEARNING OPTION AND SET THE NO. OF TRAINING CYCLES AS 30 AND THEN CLICK ON SAVE AND TRAIN. AFTER SOME TIME THE TRAINING STATISTICS WILL BE SHOWN WHICH CAN BE ANALYSED

The screenshot shows the Edge Impulse Studio interface for training a machine learning model. On the left, a sidebar lists options like Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse design, Create impulse, Spectral features, and Classifier. A prominent 'Upgrade Plan' section offers higher job limits and more collaborators. The main area is divided into two sections: 'Advanced training settings' and 'Neural network architecture'. In 'Advanced training settings', the learning rate is set to 0.0005 and the training processor is set to CPU. In 'Neural network architecture', the model consists of an input layer (54 features), a dense layer (20 neurons), another dense layer (10 neurons), and an output layer (3 classes). A button labeled 'Save & train' is at the bottom. To the right, a large panel displays training statistics. It shows 'Model training complete' and 'Job completed (success)'. The 'Model' section indicates a 'Quantized (int8)' version. Under 'Last training performance (validation set)', it shows an accuracy of 98.2% and a loss of 0.07. A 'Confusion matrix (validation set)' table is provided:

	UP-DOWN	REST	SIDE
UP-DOWN	100%	0%	0%
REST	0%	100%	0%
SIDE	4.8%	0%	95.2%
F1 SCORE	0.96	1.00	0.98

Below this, a 'Metrics (validation set)' table shows the Area under ROC Curve at 1.00 and a Weighted average Precision of 0.98.

# TESTING OF THE MODEL

YOU CAN CLICK ON THE MODEL TESTING OPTION TO TEST THE MODEL PREDICTIONS AND ITS ACCURACY

The screenshot shows the Edge Impulse Studio interface with the URL `studio.edgeimpulse.com/studio/769868/impulse/1/validation` in the browser. The left sidebar contains navigation links for Data acquisition, Experiments, EON Tuner, Impulse design (Create impulse, Spectral features, Classifier, Retrain model), Live classification, Model testing (selected), Deployment, Versioning, and an Upgrade Plan. The main area is divided into two sections: 'Test data' and 'Model testing output'. The 'Test data' section shows a table with columns: SAMPLE NAME, EXPECTED OUTCOME, LENGTH, ACCURACY, and RESULT. It lists several rows, including three green-highlighted rows for 'Up-Down....' samples with 100% accuracy and one orange-highlighted row for 'rest.63hpo...' with 0% accuracy. A 'Classify all' button is at the top of this section. The 'Model testing output' section shows logs of the classification process, indicating a job was scheduled and started successfully, and a summary was generated. The results section displays an accuracy of 90.00% and a table of classifier metrics: Area under ROC Curve (1.00), Weighted average Precision (0.94), Weighted average Recall (0.91), and Weighted average F1 score (0.91). A 'Resume tutorial' button is at the bottom right.

Test data

Model testing output

Classifying data for Classifier...

Classifying data for float32 model...

✓ Job scheduled at 29 Aug 2025 17:44:52

[spinner-done] Job started at 29 Aug 2025 17:44:52

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.

Classifying data for Classifier OK

Generating model testing summary...

Finished generating model testing summary

Job completed (success)

Results

Model version: Unoptimized (float32)

ACCURACY  
90.00%

Metrics for Classifier

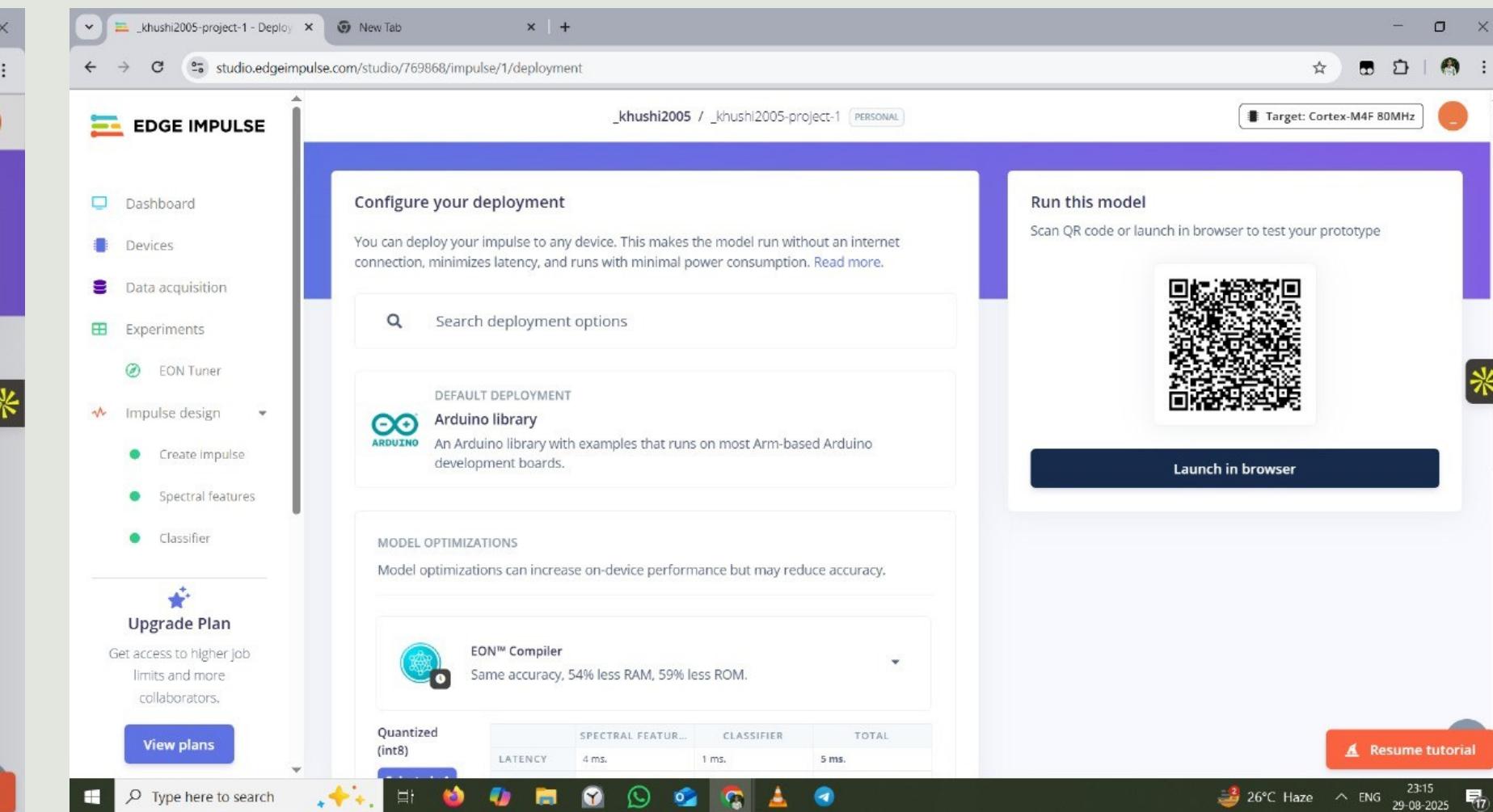
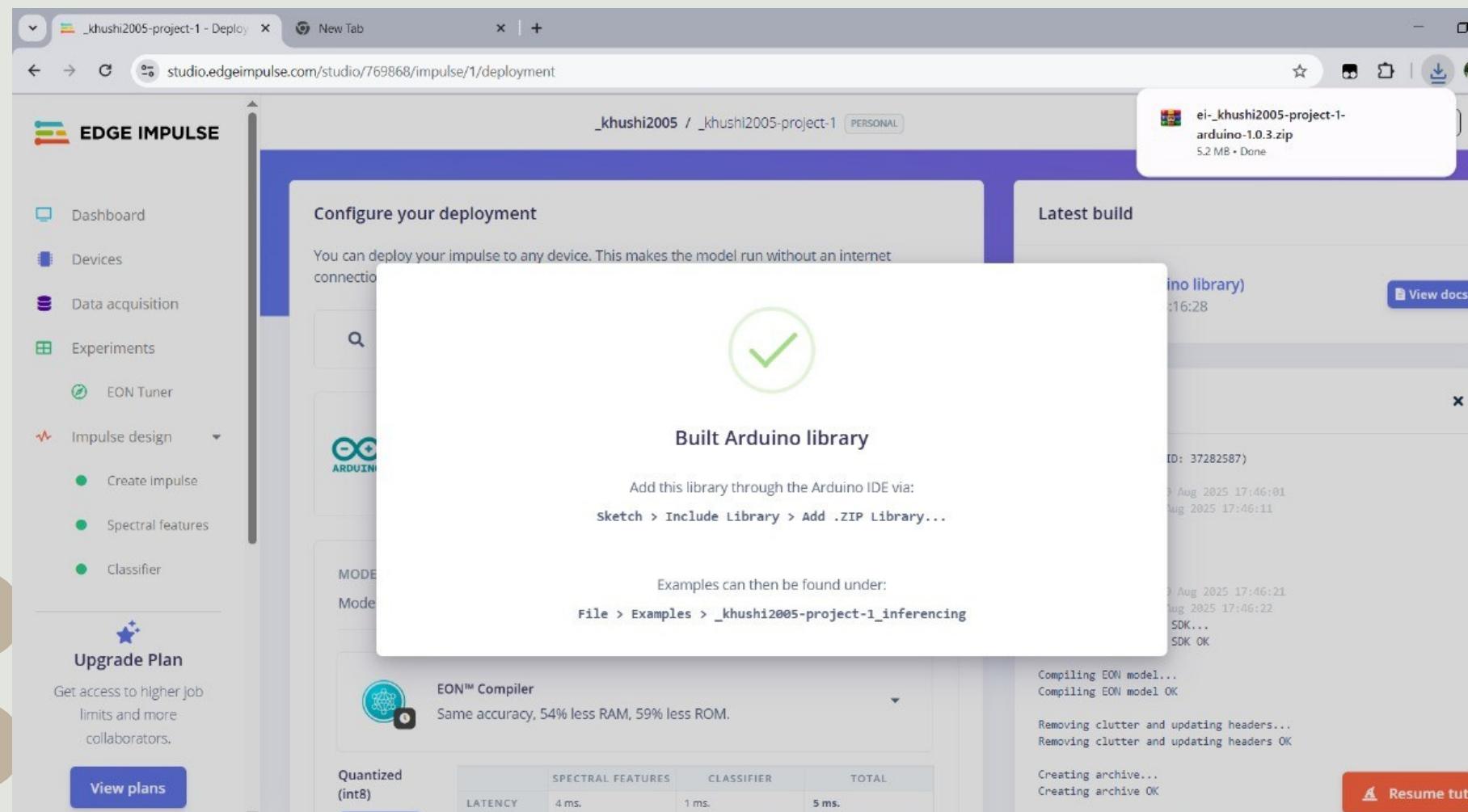
METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.94
Weighted average Recall	0.91
Weighted average F1 score	0.91

View plans

Resume tutorial

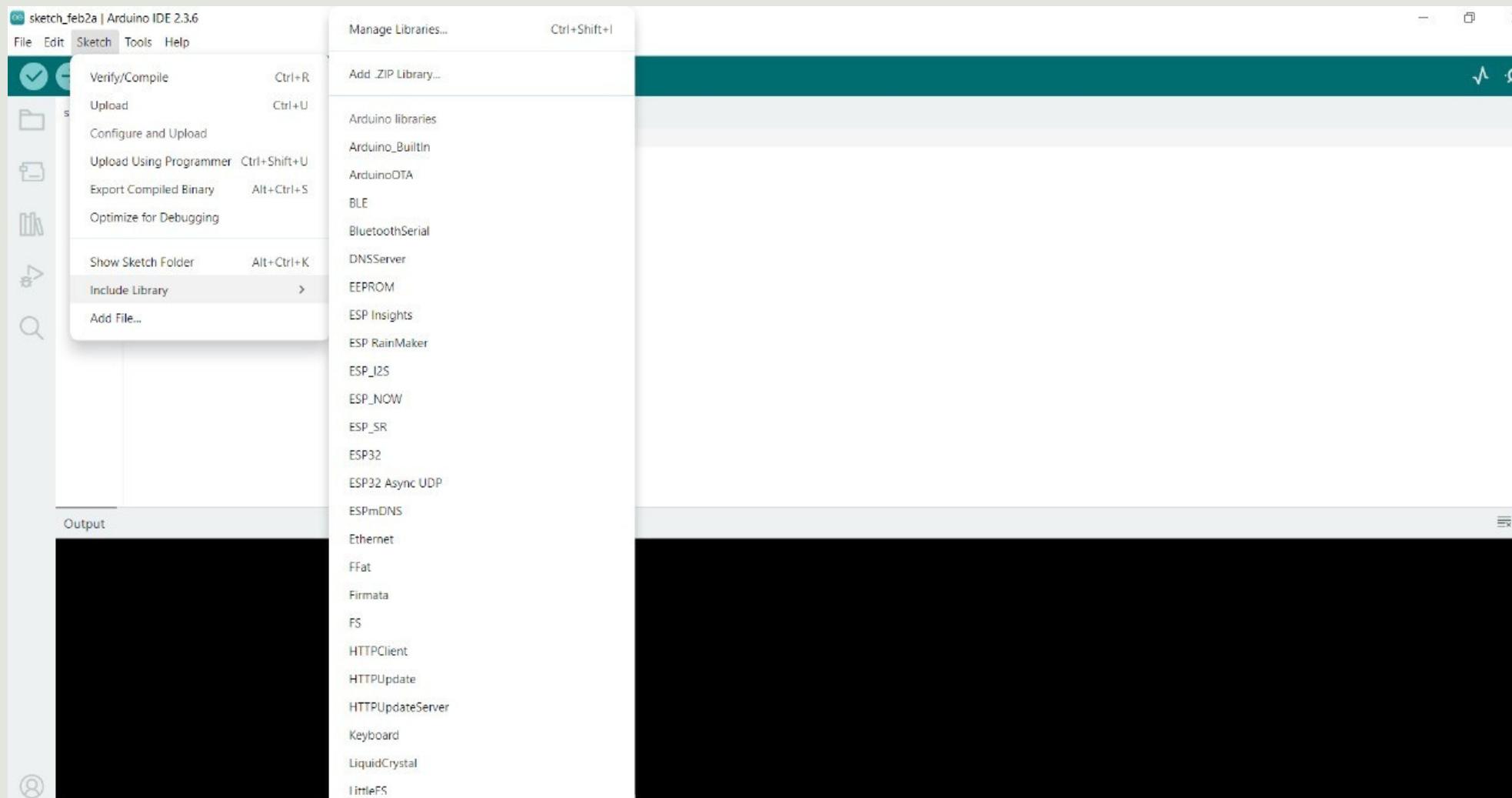
# DEPLOYMENT OF MODEL

DEPLOY THE MODEL USING THE MODEL DEPLOYMENT TAB AND SELECT THE DEFAULT DEPLOYMENT AS ARDUINO LIBRARY, WHICH WILL DOWNLOAD A LIBRARY FILE INTO YOUR LAPTOP



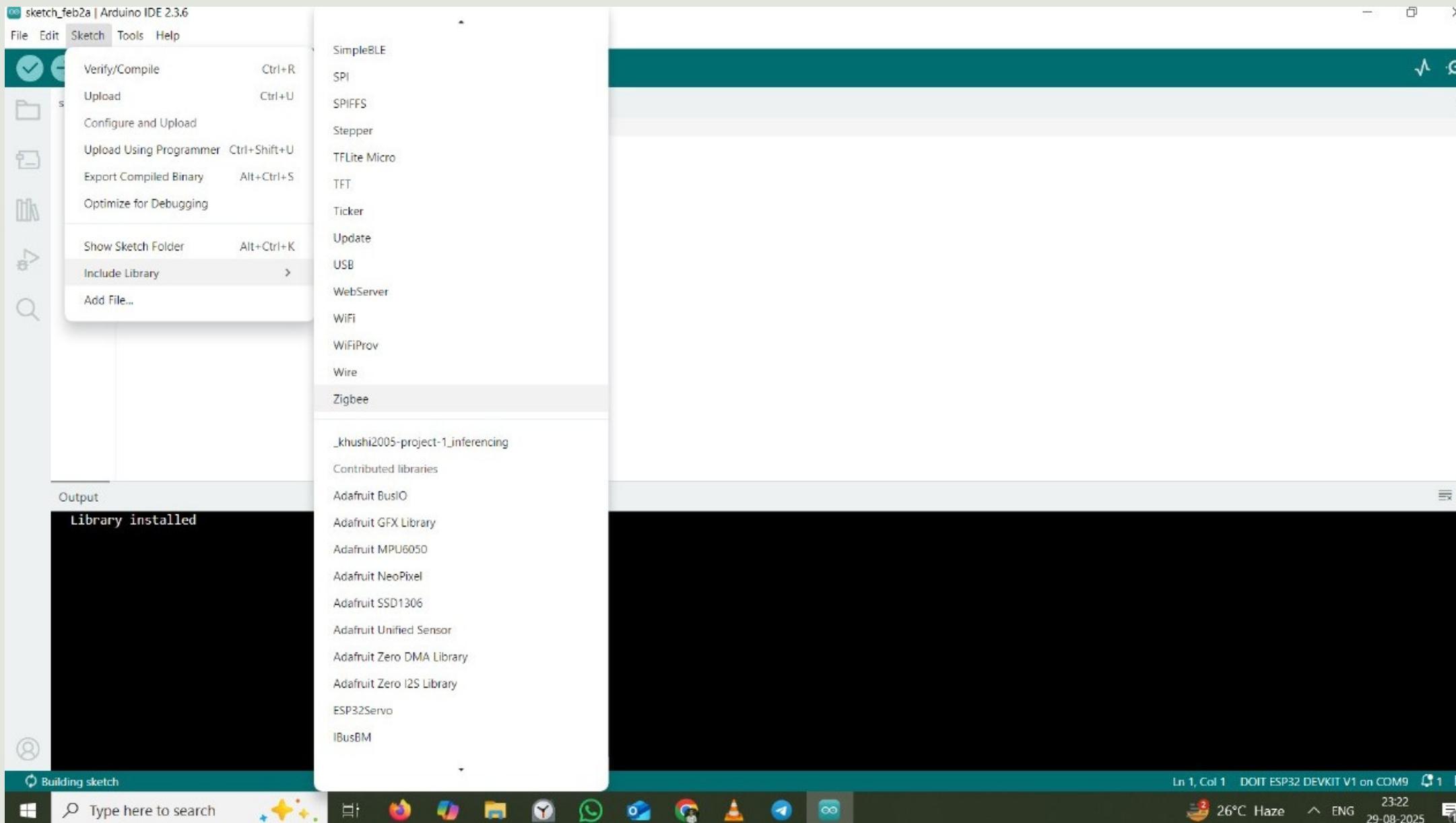
# LIBRARY INSTALLATION IN ARDUINO IDE

IN ARDUINO IDE, CLICK ON SKETCH, THEN INCLUDE LIBRARIES AND ON THE TOP, CLICK ON ADD .ZIP LIBRARY AND SELECT YOUR DOWNLOADED ZIP FILE. THIS WILL INSTALL THE MODEL LIBRARY ON YOUR ARDUINO IDE.



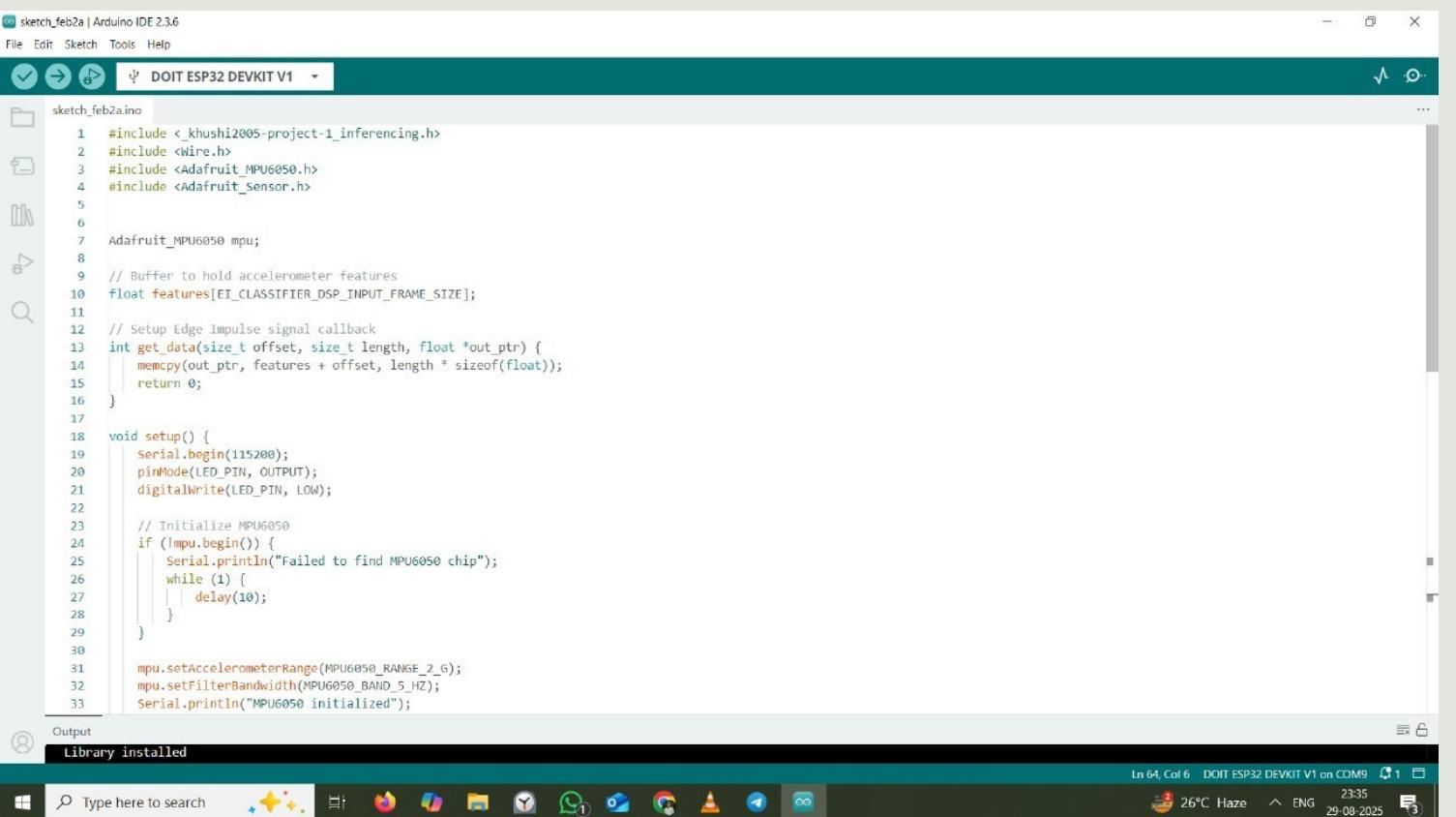
# USING THE MODEL

AFTER YOUR LIBRARY IS INSTALLED, ONCE AGAIN GO TO SKETCH AND INCLUDE LIBRARY. THEN CLICK ON YOUR MODEL LIBRARY.

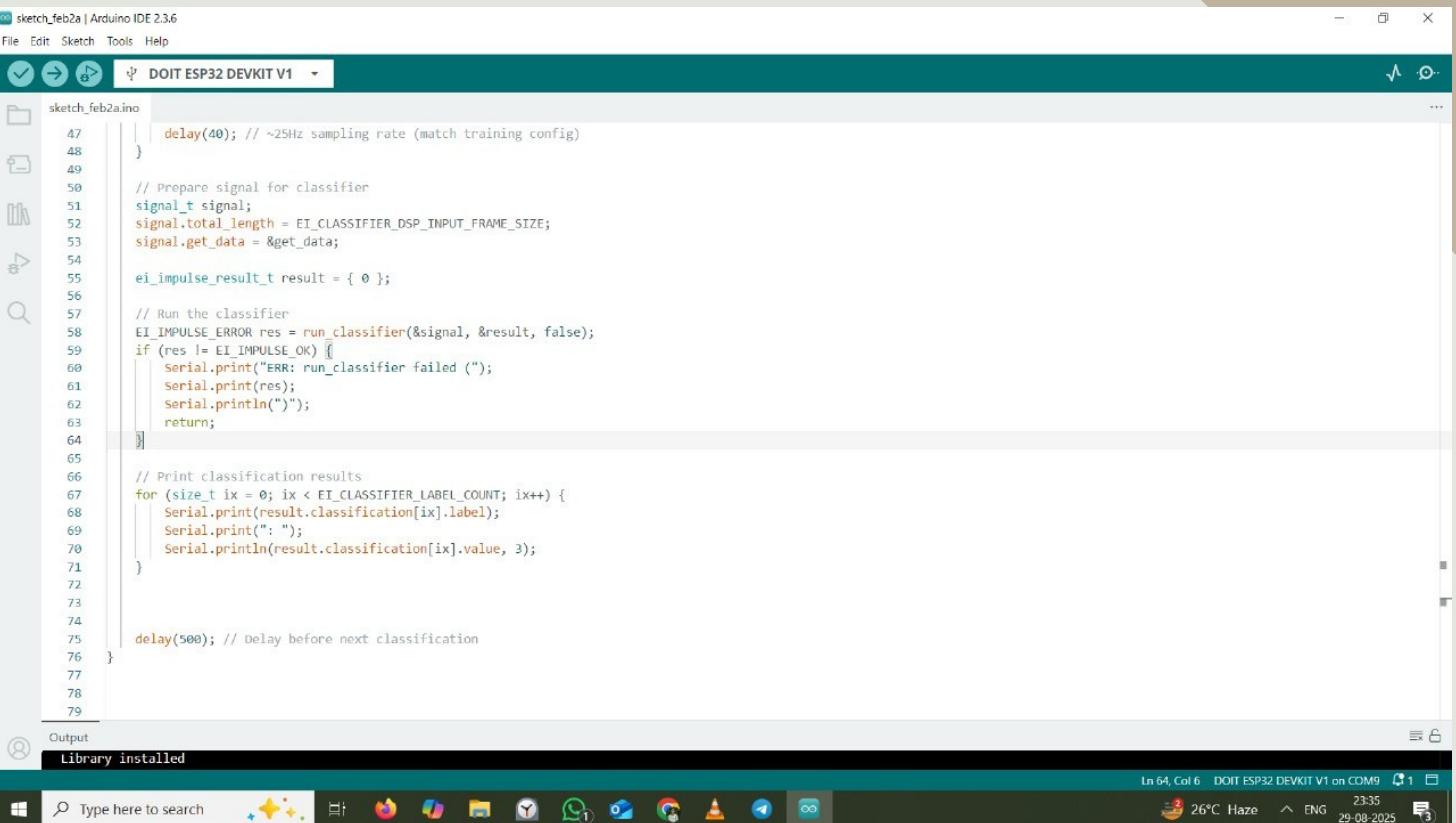


# FURTHER CODE AND USE

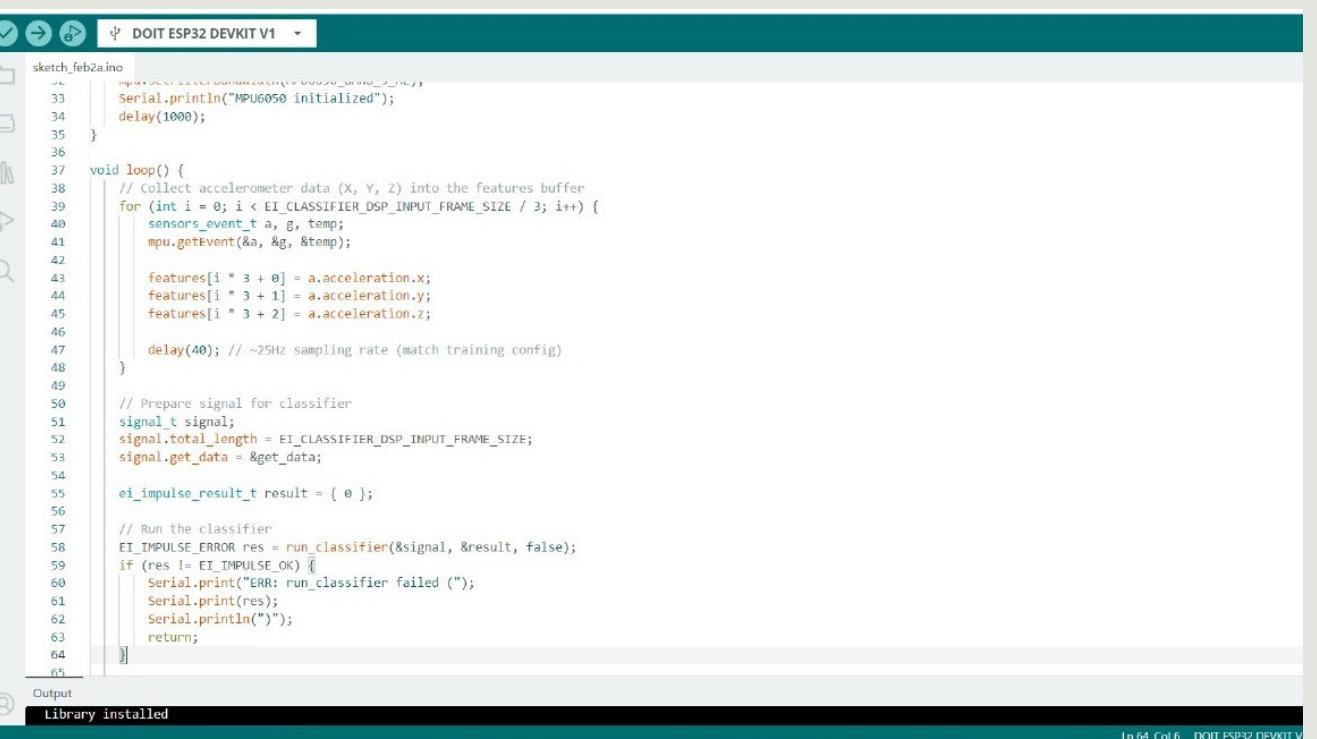
WRITE THIS CODE WITH YOUR LIBRARY. THIS CODE WILL EXTRACT DATA FROM YOUR MPU6050 AND THEN CLASSIFY IT AND PREDICT THE GESTURE ON THE SERIAL MONITOR



```
sketch_feb2a.ino
1 #include <khush12005-project-1_inferencing.h>
2 #include <Wire.h>
3 #include <Adafruit_MPU6050.h>
4 #include <Adafruit_Sensor.h>
5
6 Adafruit_MPU6050 mpu;
7
8 // Buffer to hold accelerometer features
9 float features[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];
10
11 // Setup Edge Impulse signal callback
12 int get_data(size_t offset, size_t length, float *out_ptr) {
13     memcpy(out_ptr, features + offset, length * sizeof(float));
14     return 0;
15 }
16
17 void setup() {
18     Serial.begin(115200);
19     pinMode(LED_PIN, OUTPUT);
20     digitalWrite(LED_PIN, LOW);
21
22     // Initialize MPU6050
23     if (!mpu.begin()) {
24         Serial.println("Failed to find MPU6050 chip");
25         while (1) {
26             delay(10);
27         }
28     }
29
30     mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
31     mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
32     Serial.println("MPU6050 initialized");
33 }
```



```
sketch_feb2a.ino
47     delay(40); // ~25Hz sampling rate (match training config)
48
49
50     // Prepare signal for classifier
51     signal_t signal;
52     signal.total_length = EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE;
53     signal.get_data = &get_data;
54
55     ei_impulse_result_t result = { 0 };
56
57     // Run the classifier
58     EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);
59     if (res != EI_IMPULSE_OK) {
60         Serial.print("ERR: run_classifier failed ('");
61         Serial.print(res);
62         Serial.println("')");
63     }
64
65
66     // Print classification results
67     for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
68         Serial.print(result.classification[ix].label);
69         Serial.print(": ");
70         Serial.println(result.classification[ix].value, 3);
71     }
72
73
74     delay(500); // Delay before next classification
75
76
77
78
79 }
```



```
sketch_feb2a.ino
33     Serial.println("MPU6050 initialized");
34     delay(1000);
35 }
36
37 void loop() {
38     // Collect accelerometer data (X, Y, Z) into the features buffer
39     for (int i = 0; i < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE / 3; i++) {
40         sensors_event_t a, g, temp;
41         mpu.getEvent(&a, &g, &temp);
42
43         features[i * 3 + 0] = a.acceleration.x;
44         features[i * 3 + 1] = a.acceleration.y;
45         features[i * 3 + 2] = a.acceleration.z;
46
47         delay(40); // ~25Hz sampling rate (match training config)
48
49     // Prepare signal for classifier
50     signal_t signal;
51     signal.total_length = EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE;
52     signal.get_data = &get_data;
53
54     ei_impulse_result_t result = { 0 };
55
56     // Run the classifier
57     EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);
58     if (res != EI_IMPULSE_OK) {
59         Serial.print("ERR: run_classifier failed ('");
60         Serial.print(res);
61         Serial.println("')");
62     }
63 }
```

# **THANK YOU!!!**

