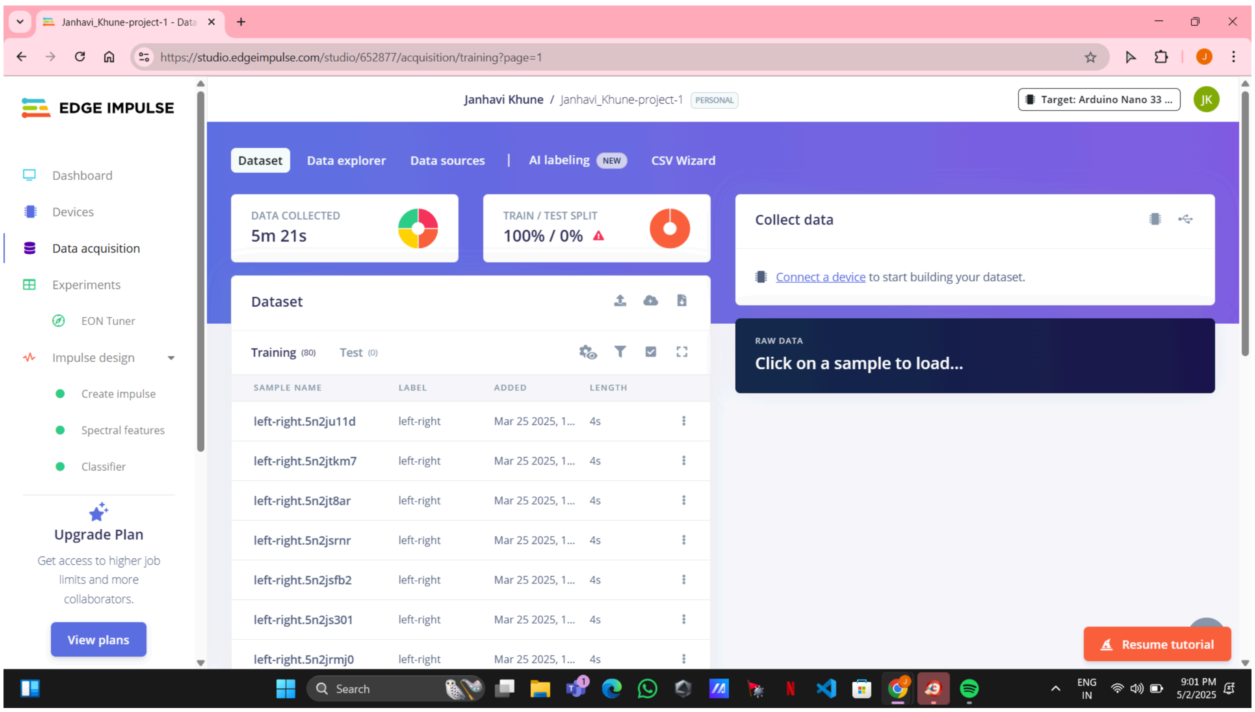
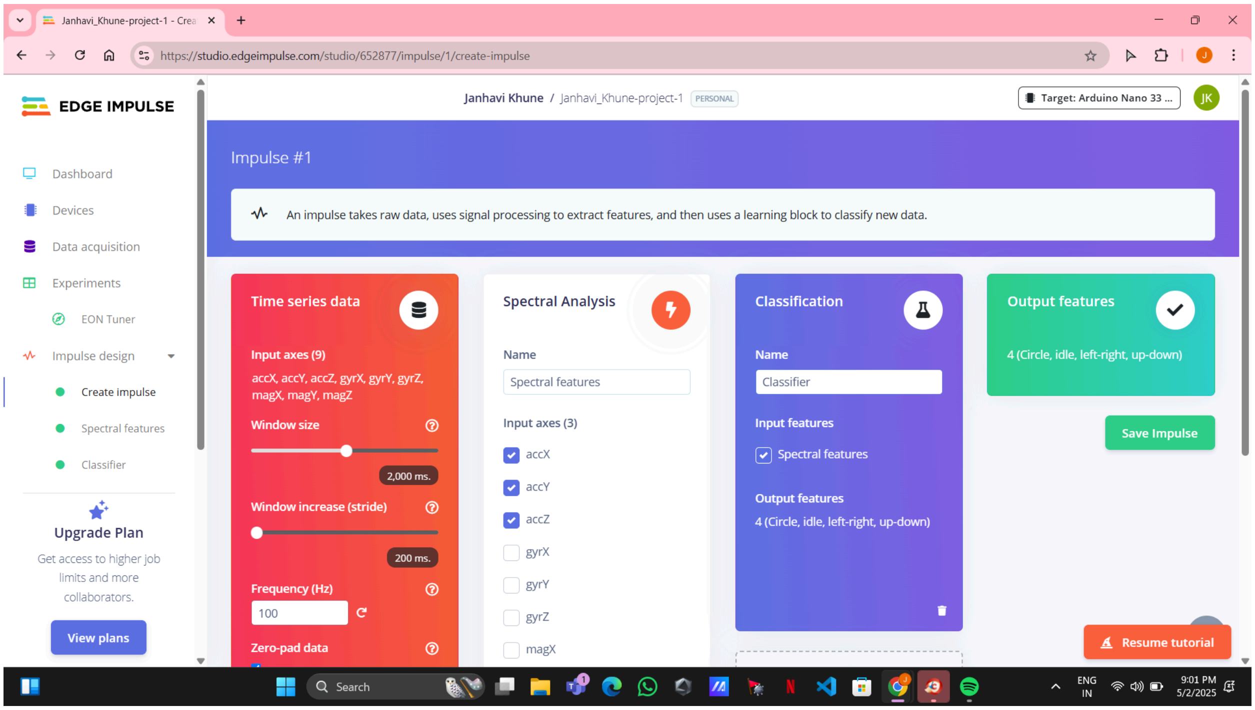
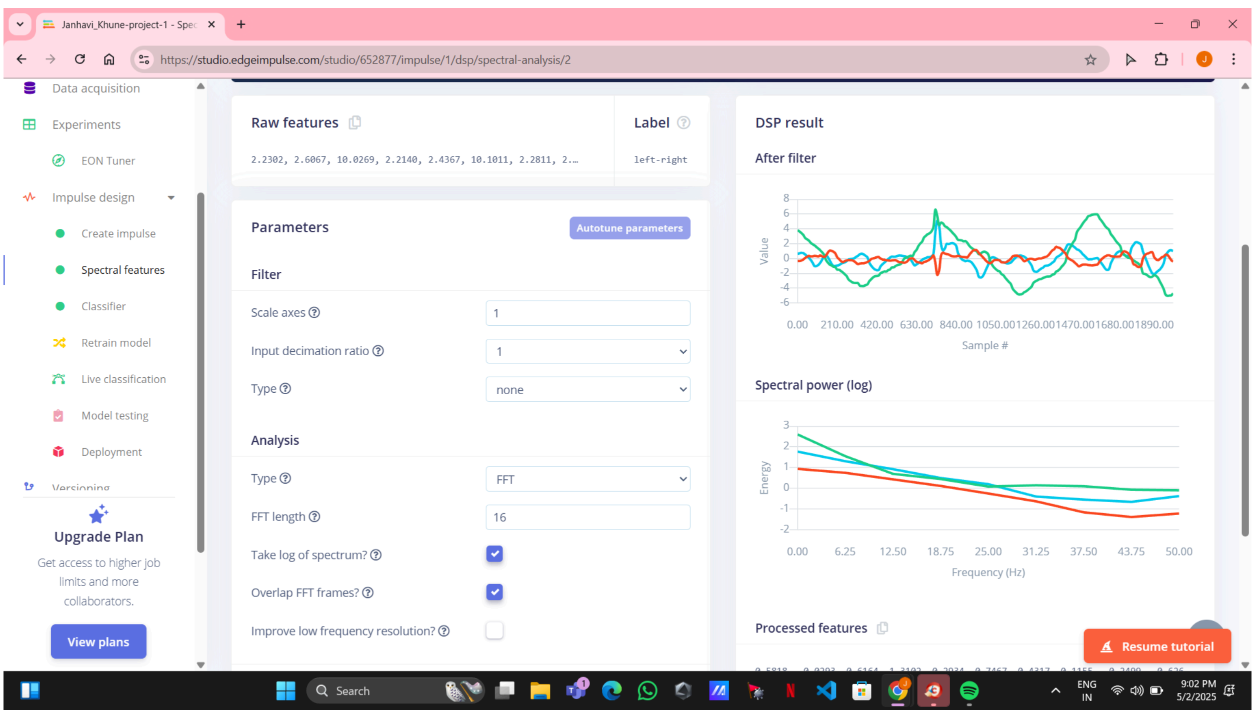
Name : Janhavi Avinash Khune
Class: TY-15 (AIEC-1)
Roll no. 222S3862
Subject: ECL

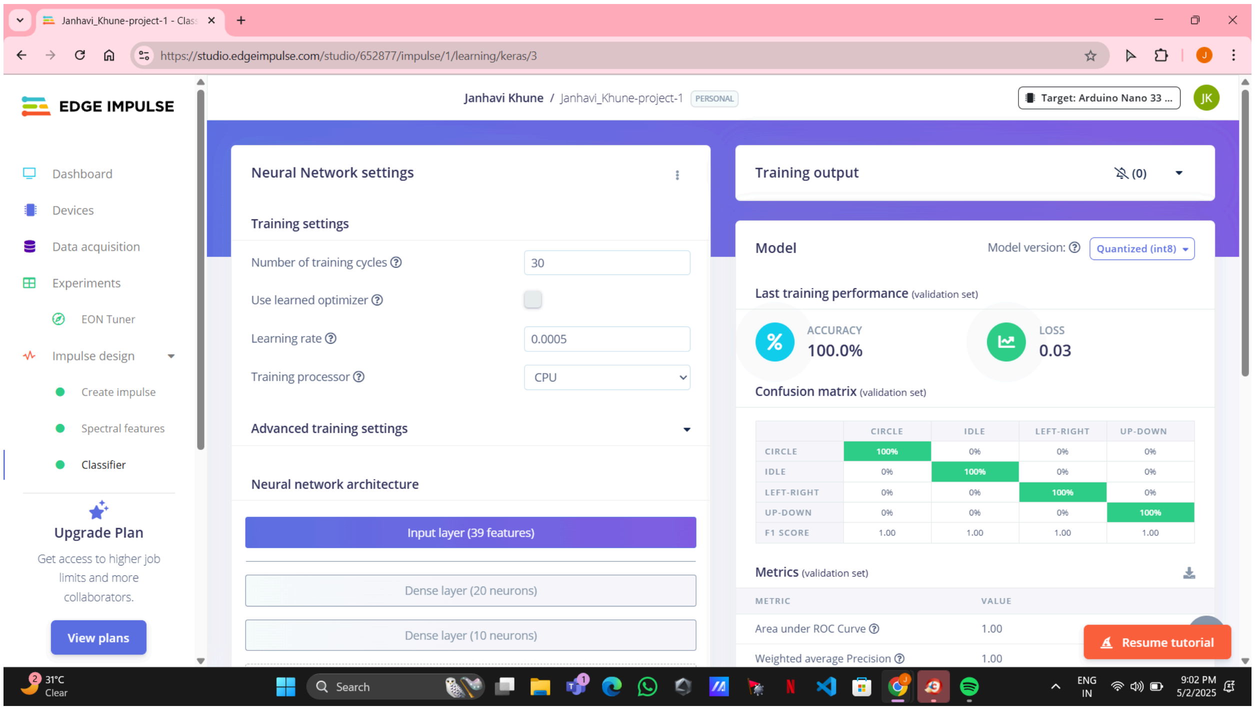
Experiment 8

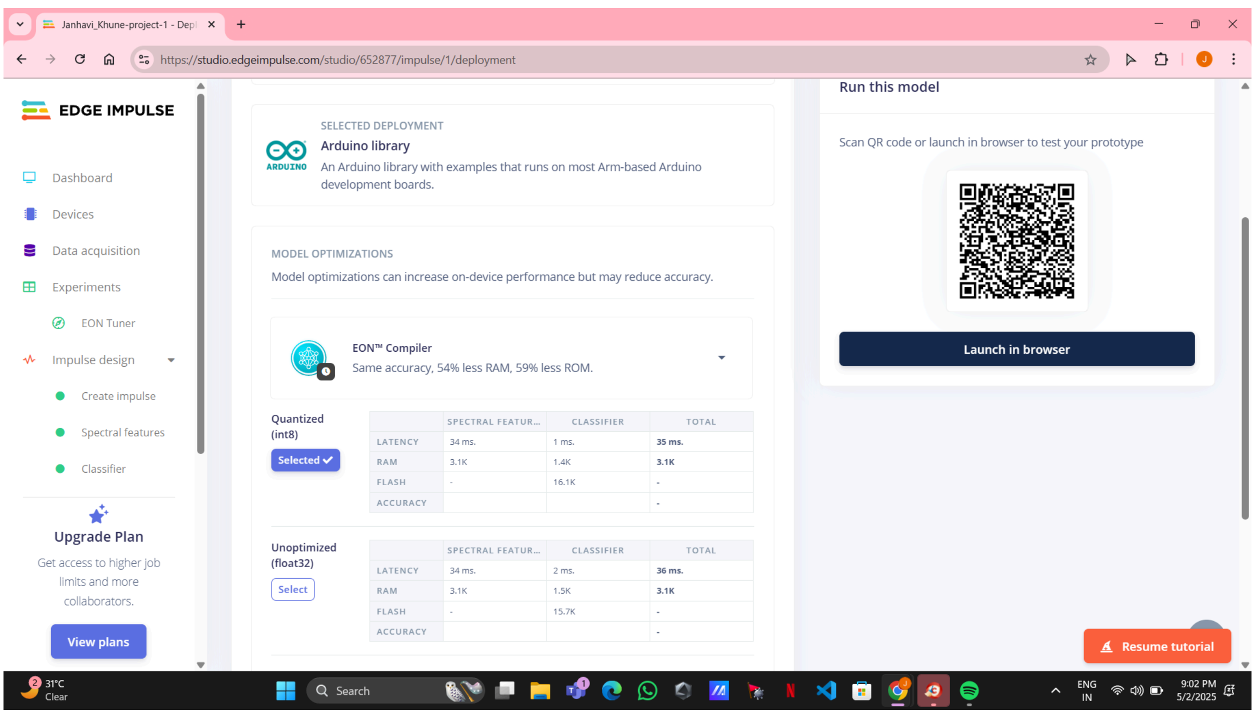
Dataset training -




Janhavi_Khune-project-1 - Crea
https://studio.edgeimpulse.com/studio/652877/impulse/1/create-impulse
EDGE IMPULSE
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
Janhavi Khune / Janhavi_Khune-project-1 PERSONAL
Target: Arduino Nano 33 ...
Impulse #1
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Time series data
Input axes (9)
accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ
Window size
2,000 ms.
Window increase (stride)
200 ms.
Frequency (Hz)
100
Zero-pad data
Spectral Analysis
Name
Spectral features
Input axes (3)
accX
accY
accZ
gyrX
gyrY
gyrZ
magX
Classification
Name
Classifier
Input features
Spectral features
Output features
4 (Circle, idle, left-right, up-down)
Output features
4 (Circle, idle, left-right, up-down)
Save Impulse
Resume tutorial
Search
ENG IN
9:01 PM 5/2/2025


Janhavi_Khune-project-1 - Spec
https://studio.edgeimpulse.com/studio/652877/impulse/1/dsp/spectral-analysis/2
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
Spectral features
Classifier
Retrain model
Live classification
Model testing
Deployment
Upgrade Plan
Get access to higher job limits and more collaborators.
View plans
Raw features
2.2302, 2.6067, 10.0269, 2.2140, 2.4367, 10.1011, 2.2811, 2...
Label
left-right
Parameters
Autotune parameters
Filter
Scale axes
1
Input decimation ratio
1
Type
none
Analysis
Type
FFT
FFT length
16
Take log of spectrum?
Overlap FFT frames?
Improve low frequency resolution?
DSP result
After filter
Value
Sample #
Spectral power (log)
Energy
Frequency (Hz)
Processed features
Resume tutorial
Search
ENG IN
9:02 PM 5/2/2025


Janhavi_Khune-project-1 - Clas
https://studio.edgeimpulse.com/studio/652877/impulse/1/learning/keras/3
EDGE IMPULSE
Janhavi Khune / Janhavi_Khune-project-1 PERSONAL
Target: Arduino Nano 33 ...
JK
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
Neural Network settings
Training settings
Number of training cycles
30
Use learned optimizer
Learning rate
0.0005
Training processor
CPU
Advanced training settings
Neural network architecture
Input layer (39 features)
Dense layer (20 neurons)
Dense layer (10 neurons)
Training output
(0)
Model
Model version:
Quantized (int8)
Last training performance (validation set)
ACCURACY
100.0%
LOSS
0.03
Confusion matrix (validation set)
CIRCLE IDLE LEFT-RIGHT UP-DOWN
CIRCLE 100% 0% 0% 0%
IDLE 0% 100% 0% 0%
LEFT-RIGHT 0% 0% 100% 0%
UP-DOWN 0% 0% 0% 100%
F1 SCORE 1.00 1.00 1.00 1.00
Metrics (validation set)
METRIC VALUE
Area under ROC Curve 1.00
Weighted average Precision 1.00
Resume tutorial
31°C
Clear
Search
ENG
IN
9:02 PM
5/2/2025


Janhavi_Khune-project-1 - Dep
https://studio.edgeimpulse.com/studio/652877/impulse/1/deployment
EDGE IMPULSE
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
SELECTED DEPLOYMENT
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.
MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.
EON™ Compiler
Same accuracy, 54% less RAM, 59% less ROM.
Quantized (int8)
Selected
SPECTRAL FEATUR... CLASSIFIER TOTAL
LATENCY 34 ms. 1 ms. 35 ms.
RAM 3.1K 1.4K 3.1K
FLASH - 16.1K -
ACCURACY -
Unoptimized (float32)
Select
SPECTRAL FEATUR... CLASSIFIER TOTAL
LATENCY 34 ms. 2 ms. 36 ms.
RAM 3.1K 1.5K 3.1K
FLASH - 15.7K -
ACCURACY -
Run this model
Scan QR code or launch in browser to test your prototype
Launch in browser
Resume tutorial
31°C
Clear
Search
ENG
IN
9:02 PM
5/2/2025

CODE:-

```
/* Edge Impulse ingestion SDK
* Copyright (c) 2022 EdgeImpulse Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*
*/
/* Includes ---------------------------------------------------------------
*/
#include <janhavi_khune-project-1_inferencing.h>
#include <Arduino_LSM9DS1.h> //Click here to get the library:
https://www.arduino.cc/reference/en/libraries/arduino_lsm9ds1/
/* Constant defines -------------------------------------------------------
*/
#define CONVERT_G_TO_MS2 9.80665f
/**
* When data is collected by the Edge Impulse Arduino Nano 33 BLE Sense
* firmware, it is limited to a 2G range. If the model was created with a
* different sample range, modify this constant to match the input values.
* See
https://github.com/edgeimpulse/firmware-arduino-nano-33-blesense/blob/master/src/sensors/ei_
lsm9ds1.cpp
* for more information.
*/
#define MAX_ACCEPTED_RANGE 2.0f
/*
** NOTE: If you run into TFLite arena allocation issue.
**
** This may be due to may dynamic memory fragmentation.
** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt
(create
** if it doesn't exist) and copy this file to
**
`<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed_core>/<core_version>/`.
```

```
**
** See
**
(https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-theinstalled-cores-locate
d-)
** to find where Arduino installs cores on your machine.
**
** If the problem persists then there's not enough memory for this model and
application.
*/
/* Private variables -------------------------------------------------------
*/
static bool debug_nn = false; // Set this to true to see e.g. features
generated from the raw signal
static uint32_t run_inference_every_ms = 200;
static rtos::Thread inference_thread(osPriorityLow);
static float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };
static float inference_buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];
/* Forward declaration */
void run_inference_background();
/**
* @brief Arduino setup function
*/
void setup()
{
// put your setup code here, to run once:
Serial.begin(115200);
// comment out the below line to cancel the wait for USB connection
(needed for native USB)
while (!Serial);
Serial.println("Edge Impulse Inferencing Demo");
if (!IMU.begin()) {
ei_printf("Failed to initialize IMU!\r\n");
}
else {
ei_printf("IMU initialized\r\n");
}
if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be equal to
3 (the 3 sensor axes)\n");
return;
}
inference_thread.start(mbed::callback(&run_inference_background));
}
```

```
/**
* @brief Return the sign of the number
*
* @param number
* @return int 1 if positive (or 0) -1 if negative
*/
float ei_get_sign(float number) {
return (number >= 0.0) ? 1.0 : -1.0;
}
/**
* @brief Run inferencing in the background.
*/
void run_inference_background()
{
// wait until we have a full buffer
delay((EI_CLASSIFIER_INTERVAL_MS * EI_CLASSIFIER_RAW_SAMPLE_COUNT) + 100);
// This is a structure that smoothens the output result
// With the default settings 70% of readings should be the same before
classifying.
ei_classifier_smooth_t smooth;
ei_classifier_smooth_init(&smooth, 10 /* no. of readings */, 7 /* min.
readings the same */, 0.8 /* min. confidence */, 0.3 /* max anomaly */);
while (1) {
// copy the buffer
memcpy(inference_buffer, buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE *
sizeof(float));
// Turn the raw buffer in a signal which we can the classify
signal_t signal;
int err = numpy::signal_from_buffer(inference_buffer,
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
if (err != 0) {
ei_printf("Failed to create signal from buffer (%d)\n", err);
return;
}
// Run the classifier
ei_impulse_result_t result = { 0 };
err = run_classifier(&signal, &result, debug_nn);
if (err != EI_IMPULSE_OK) {
ei_printf("ERR: Failed to run classifier (%d)\n", err);
return;
}
// print the predictions
ei_printf("Predictions ");
ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
```

```
result.timing.dsp, result.timing.classification,
result.timing.anomaly);
ei_printf(": ");
// ei_classifier_smooth_update yields the predicted label
const char *prediction = ei_classifier_smooth_update(&smooth,
&result);
ei_printf("%s ", prediction);
// print the cumulative results
ei_printf(" [ ");
for (size_t ix = 0; ix < smooth.count_size; ix++) {
ei_printf("%u", smooth.count[ix]);
if (ix != smooth.count_size + 1) {
ei_printf(", ");
}
else {
ei_printf(" ");
}
}
ei_printf("]\n");
delay(run_inference_every_ms);
}
ei_classifier_smooth_free(&smooth);
}
/**
* @brief Get data and run inferencing
*
* @param[in] debug Get debug info if true
*/
void loop()
{
while (1) {
// Determine the next tick (and then sleep later)
uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
// roll the buffer -3 points so we can overwrite the last one
numpy::roll(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, -3);
// read to the end of the buffer
IMU.readAcceleration(
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3],
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 2],
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 1]
);
for (int i = 0; i < 3; i++) {
if (fabs(buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3 + i]) >
MAX_ACCEPTED_RANGE) {
```

```
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3 + i] =
ei_get_sign(buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3 + i]) *
MAX_ACCEPTED_RANGE;
}
}
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3] *= CONVERT_G_TO_MS2;
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 2] *= CONVERT_G_TO_MS2;
buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 1] *= CONVERT_G_TO_MS2;
// and wait for next tick
uint64_t time_to_wait = next_tick - micros();
delay((int)floor((float)time_to_wait / 1000.0f));
delayMicroseconds(time_to_wait % 1000);
}
}
#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR !=
EI_CLASSIFIER_SENSOR_ACCELEROMETER
#error "Invalid model for current sensor"
#endif
```

Output :-