Edge Computing Lab

Class: TY-AIEC

School of Computing, MIT Art Design Technology University

Academic Year: 2024-25

Experiment No. 8

Introduction

The "magic wand" project that can recognize gestures using an accelerometer and an ML classification model on Edge Devices

Objective: Build a project to detect the accelerometer values and convert them into gestures

Tasks:

- Generate the dataset for Accelerometer Motion (Up-Down, Left-Right)
- Configure BLE Sense / Mobile for Edge Impulse
- Building and Training a Model
- Deploy on Nano BLE Sense / Mobile Phone

Introduction

Edge Impulse is a development platform for machine learning on edge devices, targeted at developers who want to create intelligent device solutions. The " Accelerometer Motion "sensor reading equivalent in Edge Impulse would typically involve creating a simple machine learning model that can run on an edge device, like classifying sensor data or recognizing a basic pattern.

Materials Required

- Nano BLE Sense Board

Theory

GPIO (General Purpose Input/Output) pins on the Raspberry Pi are used for interfacing with other electronic components. BCM numbering refers to the pin numbers in the Broadcom SOC channel, which is a more consistent way to refer to the GPIO pins across different versions of the

Here's a high-level overview of steps you'd follow to create a "Hello World" project on Edge Impulse:

Steps to Configure the Edge Impulse:

1. Create an Account and New Project:
    - Sign up for an Edge Impulse account.
    - Create a new project from the dashboard.
2. Connect a Device:

    - You can use a supported development board or your smartphone as a sensor device.
    - Follow the instructions to connect your device to your Edge Impulse project.
3. Collect Data:
    - Use the Edge Impulse mobile app or the Web interface to collect data from the onboard sensors.
    - For a "Hello World" project, you could collect accelerometer data, for instance.
4. Create an Impulse:
    - Go to the 'Create impulse' page.
    - Add a processing block (e.g., time-series data) and a learning block (e.g., classification).
    - Save the impulse, which defines the machine learning pipeline.
5. Design a Neural Network:
    - Navigate to the 'NN Classifier' under the 'Learning blocks'.
    - Design a simple neural network. Edge Impulse provides a default architecture that works well for most basic tasks.
6. Train the Model:
    - Click on the 'Start training' button to train your machine learning model with the collected data.
7. Test the Model:
    - Once the model is trained, you can test its performance with new data in the 'Model Testing' tab.
8. Deploy the Model:
    - Go to the 'Deployment' tab.
    - Select the deployment method that suits your edge device (e.g., Arduino library, WebAssembly, container, etc.).
    - Follow the instructions to deploy the model to your device.
9. Run Inference:

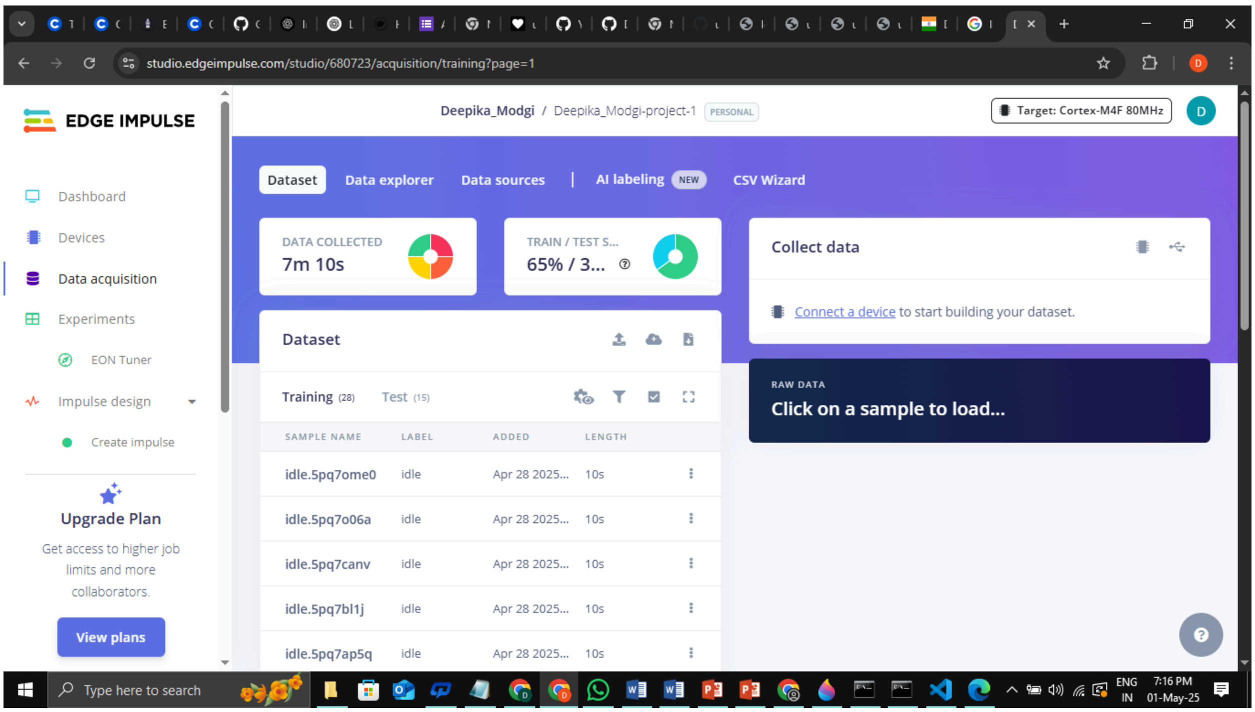
- With the model deployed, run inference on the edge device to see it classifying data in real-time.

10. Monitor:

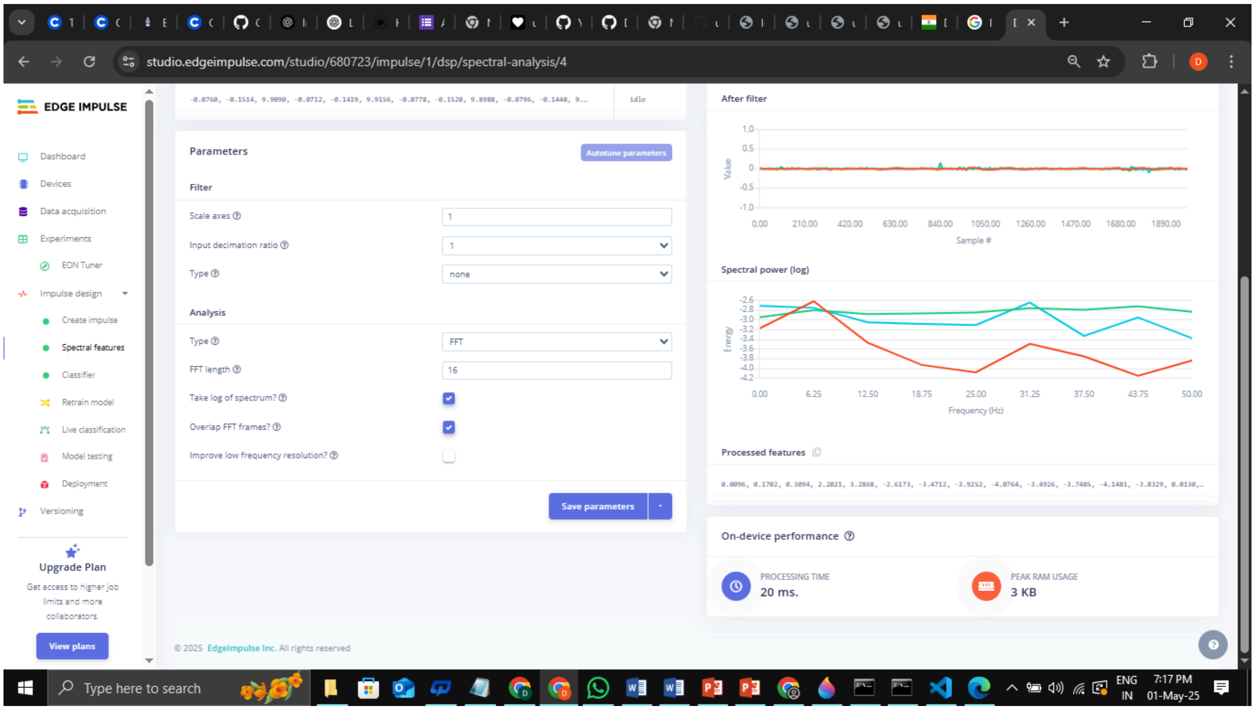
- You can monitor the performance of your device through the Edge Impulse studio.

Outcomes:

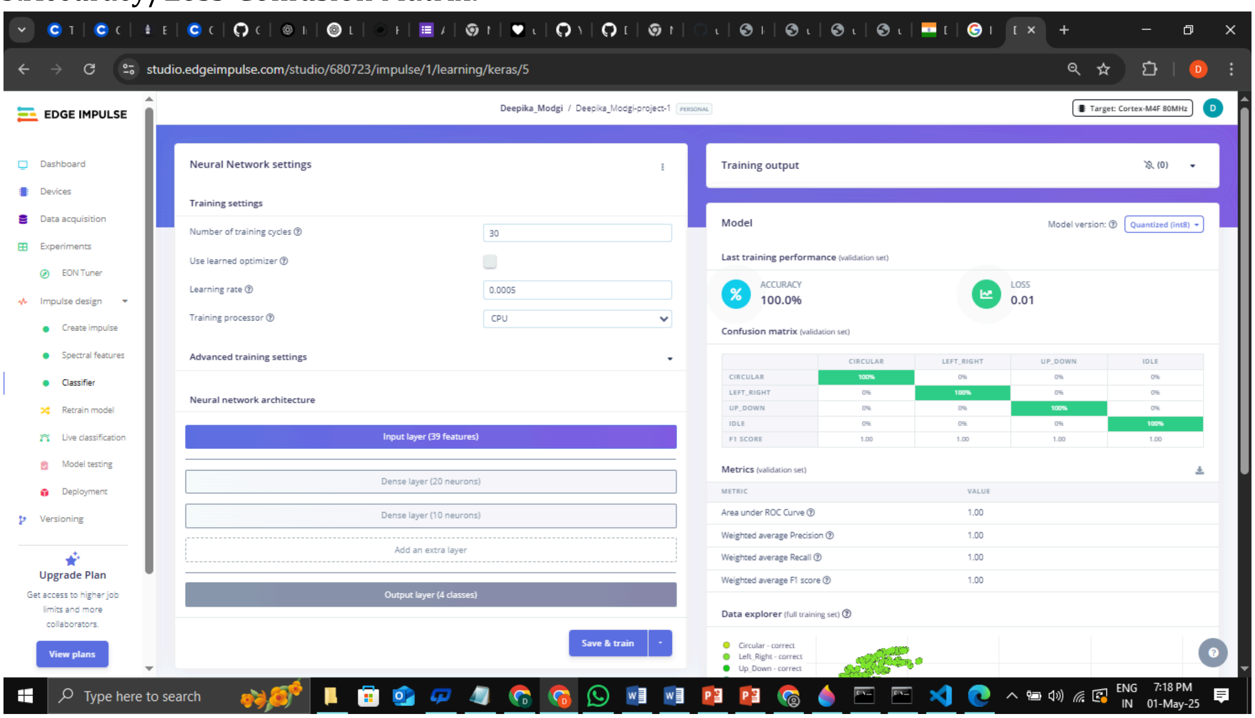
1.Dataset :

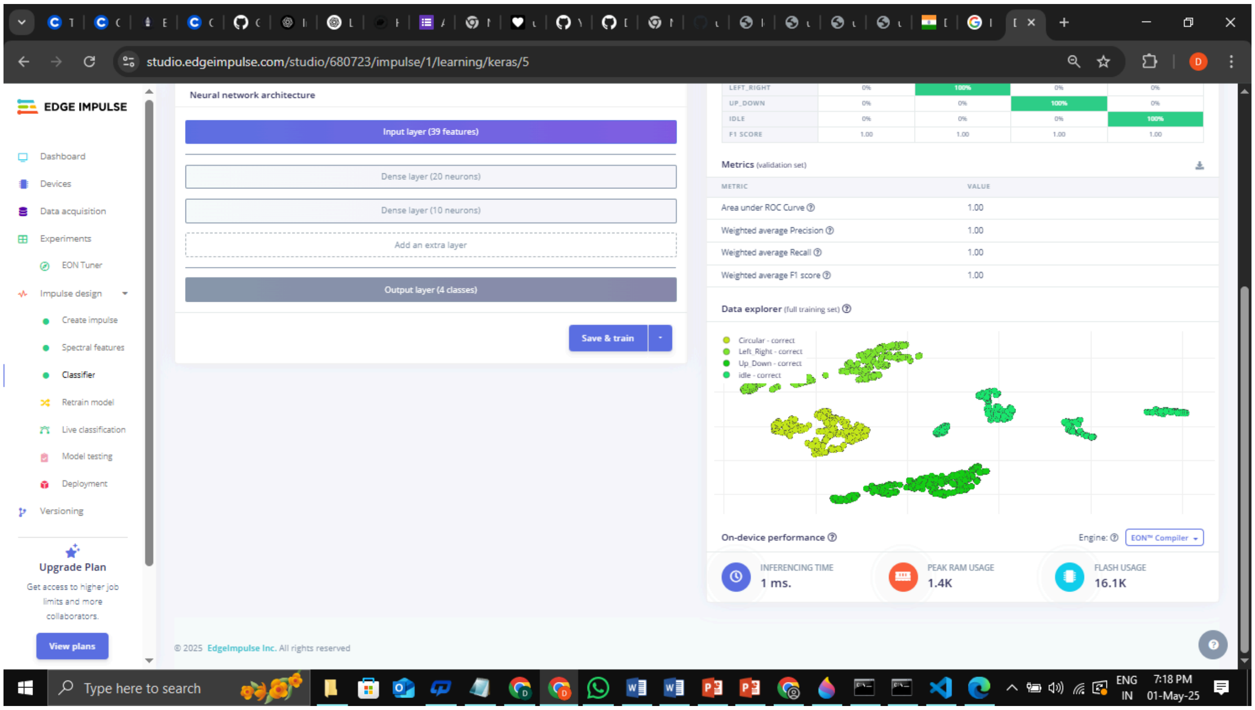


2.Feature Extraction:

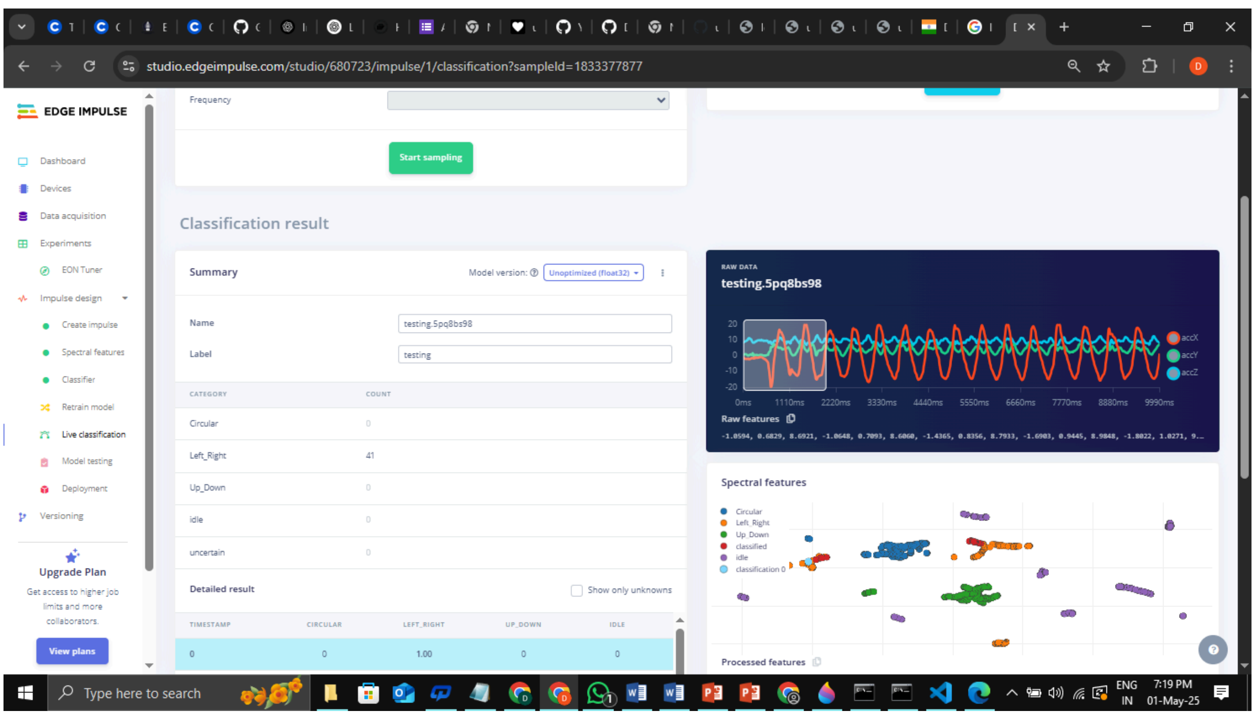


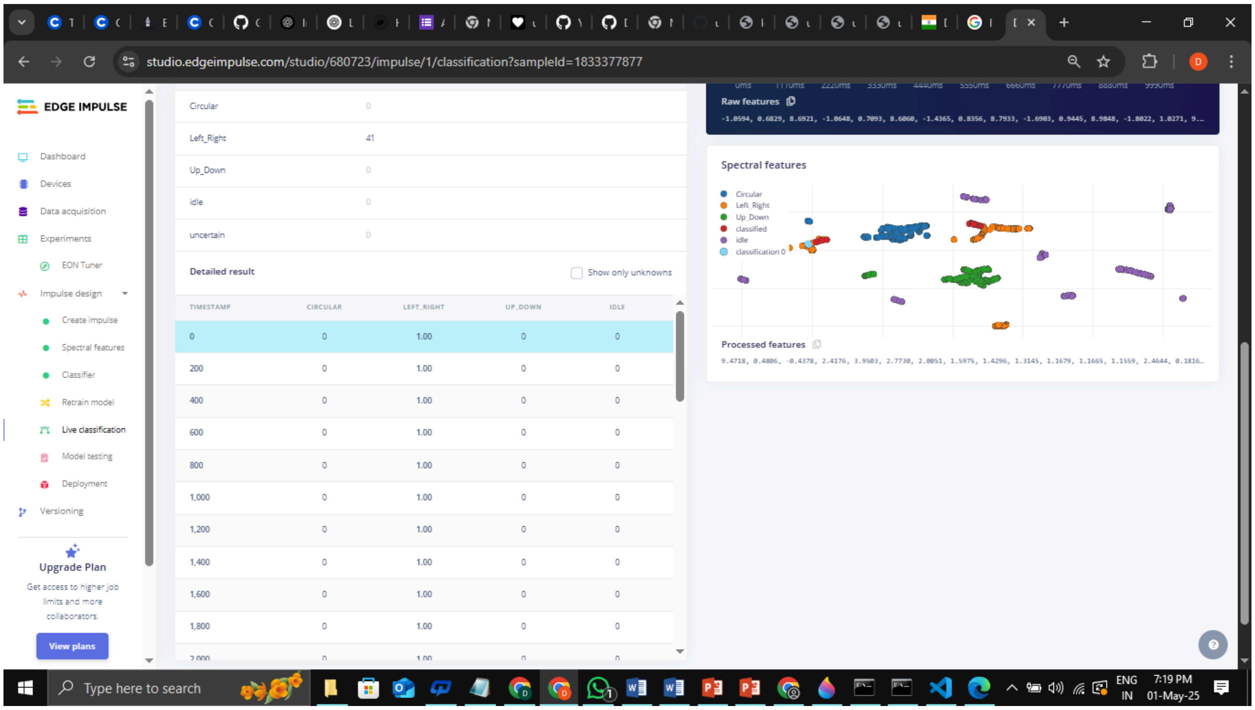
3.Accuracy/Loss-Confusion Matrix:





4.Validation Result:





5.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 <Deepika_Modgi-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-ble-sense/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-the-i
nstalled-cores-located-)
 ** 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


/**
* @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;

    }

}


/**

 * @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      Get data and run inferencing

*

* @param[in]  debug  Get debug info if true

*/

void loop()

{

    ei_printf("\nStarting inferencing in 2 seconds...\n");
```

```
    delay(2000);

    ei_printf("Sampling...\n");

    // Allocate a buffer here for the values we'll read from the IMU
    float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };

    for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3)
{
        // Determine the next tick (and then sleep later)
        uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS *
1000);

        IMU.readAcceleration(buffer[ix], buffer[ix + 1], buffer[ix + 2]);

        for (int i = 0; i < 3; i++) {
            if (fabs(buffer[ix + i]) > MAX_ACCEPTED_RANGE) {
                buffer[ix + i] = ei_get_sign(buffer[ix + i]) *
MAX_ACCEPTED_RANGE;
            }
        }

        buffer[ix + 0] *= CONVERT_G_TO_MS2;
        buffer[ix + 1] *= CONVERT_G_TO_MS2;
        buffer[ix + 2] *= CONVERT_G_TO_MS2;

        delayMicroseconds(next_tick - micros());
    }
```

```
    // Turn the raw buffer in a signal which we can the classify
    signal_t signal;
    int err = numpy::signal_from_buffer(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(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label,
result.classification[ix].value);
    }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
```

```
    ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif
}


#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR !=
EI_CLASSIFIER_SENSOR_ACCELEROMETER
#error "Invalid model for current sensor"
#endif
```

6.Arduino Sketch Screenshot:

(Forgot to take a screenshot of the running code. I'm extremely sorry.)