#include <Arduino\_LSM9DS1.h>

#include <ArduinoBLE.h>

#include <TensorFlowLite.h>

#include <tensorflow/lite/micro/all\_ops\_resolver.h>

#include <tensorflow/lite/micro/micro\_error\_reporter.h>

#include <tensorflow/lite/micro/micro\_interpreter.h>

#include <tensorflow/lite/schema/schema\_generated.h>

#include <tensorflow/lite/version.h>

#include "model.h"

const float accelerationThreshold = 2.5; // threshold of significant in G's // 以G为单位的显著阈值

const int numSamples = 50;

int samplesRead = numSamples;

int a = 0;

// global variables used for TensorFlow Lite (Micro)

tflite::MicroErrorReporter tflErrorReporter;

// pull in all the TFLM ops, you can remove this line and

// only pull in the TFLM ops you need, if would like to reduce

// the compiled size of the sketch.

tflite::AllOpsResolver tflOpsResolver;

const tflite::Model\* tflModel = nullptr;

tflite::MicroInterpreter\* tflInterpreter = nullptr;

TfLiteTensor\* tflInputTensor = nullptr;

TfLiteTensor\* tflOutputTensor = nullptr;

// Create a static memory buffer for TFLM, the size may need to

// be adjusted based on the model you are using

constexpr int tensorArenaSize = 6 \* 1024;

byte tensorArena[tensorArenaSize] \_\_attribute\_\_((aligned(16)));

// array to map gesture index to a name

const char\* GESTURES[] = {

"Walking",

"Jogging",

"Upstairs",

"Downstairs",

"Sitting",

"Standing"

};

BLEService Erkennung("180F");

// BLE Battery Level Characteristic

BLEUnsignedCharCharacteristic ErkennungChar("10", // standard 16-bit characteristic UUID

BLERead | BLENotify); // remote clients will be able to get notifications if this characteristic changes

void setup() {

Serial.begin(9600);

while (!Serial);

// initialize the IMU

if (!IMU.begin()) {

Serial.println("Failed to initialize IMU!");

while (1);

}

// begin initialization

if (!BLE.begin()) {

Serial.println("starting BLE failed!");

while (1);

}

// print out the samples rates of the IMUs

Serial.print("Accelerometer sample rate = ");

Serial.print(IMU.accelerationSampleRate());

Serial.println(" Hz");

Serial.println();

// get the TFL representation of the model byte array

tflModel = tflite::GetModel(model);

if (tflModel->version() != TFLITE\_SCHEMA\_VERSION) {

Serial.println("Model schema mismatch!");

while (1);

}

// Create an interpreter to run the model

tflInterpreter = new tflite::MicroInterpreter(tflModel, tflOpsResolver, tensorArena, tensorArenaSize, &tflErrorReporter);

// Allocate memory for the model's input and output tensors

tflInterpreter->AllocateTensors();

// Get pointers for the model's input and output tensors

tflInputTensor = tflInterpreter->input(0);

tflOutputTensor = tflInterpreter->output(0);

BLE.setLocalName("Erkennung");

BLE.setAdvertisedService(Erkennung); // add the service UUID

Erkennung.addCharacteristic(ErkennungChar); // add the battery level characteristic

BLE.addService(Erkennung); // Add the battery service

/\* Start advertising BLE. It will start continuously transmitting BLE

advertising packets and will be visible to remote BLE central devices

until it receives a new connection \*/

// start advertising

BLE.advertise();

Serial.println("Bluetooth device active, waiting for connections...");

}

void loop() {

float aX, aY, aZ;

samplesRead = 0;

// wait for a BLE central

BLEDevice central = BLE.central();

// if a central is connected to the peripheral:

if (central) {

Serial.print("Connected to central: ");

// print the central's BT address:

Serial.println(central.address());

// while the central is connected:

while (central.connected()) {

while (samplesRead < numSamples) {

// check if new acceleration data is available

if (IMU.accelerationAvailable()) {

// read the acceleration and gyroscope data

IMU.readAcceleration(aX, aY, aZ);

tflInputTensor->data.f[samplesRead \* 3 + 0] = aZ;

tflInputTensor->data.f[samplesRead \* 3 + 1] = -aY;

tflInputTensor->data.f[samplesRead \* 3 + 2] = -aX;

delay(32);

samplesRead++;

if (samplesRead == numSamples) {

// Run inferencing

tflInterpreter->Invoke();

// Loop through the output tensor values from the model

for (int i = 0; i < 6; i++) {

Serial.print(GESTURES[i]);

Serial.print(": ");

Serial.println(tflOutputTensor->data.f[i], 6);

}

for (int i = 0; i < 6; i++) {

if (tflOutputTensor->data.f[i] > tflOutputTensor->data.f[a]) {

a = i;

}

}

Serial.print(GESTURES[a]);

ErkennungChar.writeValue(a);

a = 0;

Serial.println();

}

}

}

}

}

}