UNIVERSIDAD
DE GRANADA

TRABAJO FIN DE GRADO

GRADO EN INGENIERIA INFORMATICA

Detección de escritura mediante un lápiz con giroscopio integrado haciendo uso de Machine Learning en un sistema empotrado

---

TinyML

Autor
Antonio Priego Raya

Directores
Juan José Escobar Pérez
Jesús González Peñalver

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2022

Detección de escritura mediante un lápiz con giroscopio integrado haciendo uso de Machine Learning en un sistema empotrado
TinyML

Antonio Priego Raya

Palabras clave: *software libre*

Resumen

Same, but in English

Student's name

Keywords: *open source*, *floss*

Abstract

D. **Tutora/e(s)**, Profesor(a) del ...

Informo:

Que el presente trabajo, titulado ***Detección de escritura mediante un lápiz con giroscopio integrado haciendo uso de Machine Learning en un sistema empotrado***, ha sido realizado bajo mi supervisión por **Antonio Priego Raya**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2022.

El/la director(a)/es:

Juan José Escobar Pérez **Jesús González Peñalver**

Agradecimientos

A

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Interfaz de usuario

En primer lugar, hice un diseño aproximado para la interfaz de usuario que necesitaba haciendo uso de ***QT Design Studio***. Obteniendo el siguiente resultado:

Figura 1.1: Primer boceto en QT Design Studio

Con este primer acercamiento, dispuse los elementos que pensé imprescindibles. No obstante, al finalizar esta primera iteración de diseño, pude ver que las carencias a nivel estético; facilmente resolubles cuando comience con la implementación real en ***QT Creator***. También vi que faltaban algunos botones como el de sincronización con el ***DeepPen***, que *Read Mode* no era un buen nombre para indicar el comportamiento del modo, o que había un espacio en el centro de la barra superior sin usar y que podía usar para indicar el modo actual.

Teniendo estas apreciaciones en mente, las corregí en la siguiente iteración de aproximación al diseño con el que partir en la implementación.

Figura 1.2: Segundo boceto en QT Design Studio

Para pasar esta aproximación de diseño a una implementación real, utilicé ***QT Creator*** con ***C++***. Resultando en lo siguiente:

Figura 1.3: Primera implementación de la interfaz de usuario del ***DeepPen***

Capítulo 2

Entrenamiento

Para comenzar con Arduino+TensorFlow, instalaré la bibliotecas necesarias para usar TensorFlow y la *Nano 33 Sense*:

1. **Arduino_TensorFlowLite**: Permite construir aplicaciones con aplicaciones para AI/ML.
2. **Arduino_LSM9DS1**: Provee de las herramientas para acceder al acelerómetro, magnetometro y giroscopop del *Nano 33 BLE Sense*.

[6]Además, hay que realizar una serie de cambios en la biblioteca **LSM9DS1.cpp**.
Añadir justo antes de que la función **LSM9DS1Class::begin()** retorne:

```
1    // Enable FIFO (see docs https://www.st.com/resource/en/datasheet/DM00103319.pdf)
2    writeRegister(LSM9DS1_ADDRESS, 0x23, 0x02);
3    // Set continuous mode
4    writeRegister(LSM9DS1_ADDRESS, 0x2E, 0xC0);
```

También debemos editar **LSM9DS1Class::accelerationAvailable()** :

```
1    int LSM9DS1Class::accelerationAvailable()
2    {
3        /********************************* OLD *********************************
4        if (readRegister(LSM9DS1_ADDRESS, LSM9DS1_STATUS_REG) & 0x01) {
5            return 1;
6        }
7        ********************************* OLD *********************************/
8
9        // Read FIFO_SRC. If any of the rightmost 8 bits have a value, there is data
10       if (readRegister(LSM9DS1_ADDRESS, 0x2F) & 63) {
11           return 1;
12       }
13
14       return 0;
15   }
```

Si queremos acceder al puerto serie de la placa o usar el IDE de Arduino, debemos conceder permisos al dispositivo:

```
~$ ls -l /dev/ttyACM*                    # En mi caso
~$ sudo usermod -a -G dialout <usuario>
```

Cuando ya tenemos la biblioteca para los sensores y el acceso al puerto garantizado, podemos pasar a probar el programa. Si funciona con el entrenamiento por defecto, continuamos finalmente con nuestro entrenamiento.
Realizaremos el entrenamiento recogiendo muestras para cada caracter a reconocer. Una vez tenemos suficientes muestras para todos los caracteres, podemos realizar el entrenamiento del modelo, que realizaremos en *Google Colab*. El script de entrenamiento es el siguiente: [**METER ENLACE**].

Toma de muestras, para el software que utilizaremos para tomar los datos para crear el dataset con el que entrenar el modelo, necesitaremos instalar las siguientes bibliotecas:
[1] Arduino_APDS9960: para disponer de librerías para algunos sensores adicionales.
[2] Arduino_CMSISDSP: para disponer de arm_math.h.
[4] Arduino_LPS22HB: herramientas para disponer del sensor de presión.
[3] Arduino_HTS221: herramientas para el sensor de temperatura y humedad.

Capítulo 3

Apéndice

3.1. Problemas técnicos

3.1.1. Si la placa deja de ser detectada

En la primera subida del programa con los datos de entrenamiento creados por mí, la placa dejó de ser detectada por el ordenador.
Lo cual me llevó a pensar que el bootloader se había corrompido. Sin embargo, buscando posibles causas, encontré una solución: restaurar manualmente la placa, cosa que solo funciona, evidentemente, si el bootloader no está dañado. Pulsando el botón reset de la placa rápidamente varias veces justo al conectarlo al ordenador, puede restaurarse la placa. Si ha funcionado, el "L"led se encenderá.

Esto ocurría al incorporar el modelo de datos que yo mismo entrenaba y fue uno de los problemas que más tiempo me llevó solucionar, sobre todo porque no tenía ningún tipo de referencia de por qué no estaba respondiendo correctamente el programa con el nuevo modelo.
Por suerte llegué a la solución:

3.1.2. El modelo de datos entrenado no responde [7]

Como explico en caso anterior, la incorporar el modelo de datos entrenado con el dataset de ejemplo que proporciona TensorFlow, al subir el programa a la placa, esta dejaba de reconocerse por el ordenador, teniendo que resetear la flash de la placa.
Esto se debía a que el proyecto Arduino no soportaba una de las operaciones que realiza nuestro modelo(reshape).

Podemos solucionar esto de dos formas:

1. (No probada) Simplemente hacer uso de AllOpsResolver, de forma que el interprete tendrá acceso a todas las operaciones disponibles.
2. Esta segunda es la más sofisticada, ya que al no disponer de todas las operaciones para el interprete, reduciremos la cantidad de memoria que ocupamos.
   Es la que yo implementé.

Añadir la siguiente línea al archivo ***.ino** de nuestro proyecto:

```
1 micro_mutable_op_resolver.AddBuiltin(tflite::BuiltinOperator_RESHAPE,
2                                      tflite::ops::micro::Register_RESHAPE());
```

De esta forma estamos añadiendo la operación al repertorio de las que tendrá disponibles el interprete*(static_interpreter)*, **micro_mutable_op_resolver**.

```
72 static tflite::MicroMutableOpResolver micro_mutable_op_resolver; // NOLINT
73 micro_mutable_op_resolver.AddBuiltin(
74       tflite::BuiltinOperator_DEPTHWISE_CONV_2D,
75       tflite::ops::micro::Register_DEPTHWISE_CONV_2D());
76 micro_mutable_op_resolver.AddBuiltin(
77       tflite::BuiltinOperator_MAX_POOL_2D,
78       tflite::ops::micro::Register_MAX_POOL_2D());
79 micro_mutable_op_resolver.AddBuiltin(tflite::BuiltinOperator_CONV_2D,
80                                      tflite::ops::micro::Register_CONV_2D());
81 micro_mutable_op_resolver.AddBuiltin(
82       tflite::BuiltinOperator_FULLY_CONNECTED,
83       tflite::ops::micro::Register_FULLY_CONNECTED());
84 micro_mutable_op_resolver.AddBuiltin(tflite::BuiltinOperator_SOFTMAX,
85                                      tflite::ops::micro::Register_SOFTMAX());
86 micro_mutable_op_resolver.AddBuiltin(tflite::BuiltinOperator_RESHAPE,
87                                      tflite::ops::micro::Register_RESHAPE());
88
89 // Build an interpreter to run the model with
90 static tflite::MicroInterpreter static_interpreter(
91     model, micro_mutable_op_resolver, tensor_arena, kTensorArenaSize,
92     error_reporter);
93 interpreter = &static_interpreter;
```

Figura 3.1: Fragmento de ***.ino** de nuestro proyecto.

Este fragmento de código ilustra lo que acabo de explicar.

3.1.3. Error durante el entrenamiento

Al comenzar con mi primer pequeño dataset para comprobar cuán realizable es mi idea para este modelo, tuve un error que me tuvo durante un buen tiempo ocupado.

Tras revisar que no se debía a la longitud de las secuencias de datos, como hace pensar el mensaje de error, fui a intentar reducir el tamaño de las muestras del dataset y fue cuando

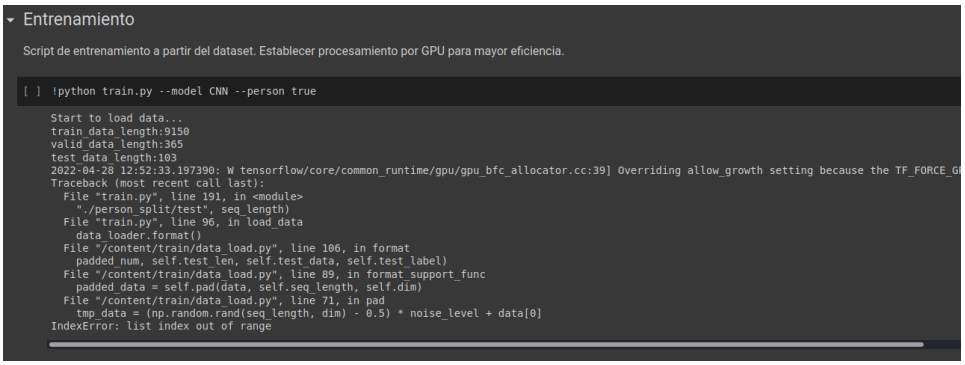

Figura 3.2: Error durante primer entrenamiento con un dataset propio.

vi que había un error en una de las muestras, de forma que había un separador (' -,-,-') sin información.
Al eliminarlo, el entrenamiento ya no arrojaba errores.

3.1.4. Error al cambiar el tamaño de las secuencias de movimientos

Los movimientos que se deben registrar para las letras son, en general, más complejos que los que incluye el dataset de ejemplo. Por lo tanto las secuencias de registro de movimiento, serán mayores. Esto supone un problema porque el modelo está ajustado al dataset de ejemplo y por tanto, se queda algo corto para nuestro propósito. El error se da al cambiar el tamaño de dicha secuencia (**seq_length**) en **train.py** y **train_test.py** .

Figura 3.3: Error al cambiar el tamaño de secuencia de movimientos.

Tras mucha documentación sobre Tensorflow, y no obtener la causa del error; pensé que esto ya me pasó por culpa de la configuración del ***.ino** . Así que fui a revisarlo y efectivamente, hay un parámetro en este archivo sujeto a la longitud de los datos. Tras cambiarlo, funcionaba correctamente de nuevo, aunque sin detectar la letra que estaba probando en el nuevo dataset.

3.1.5. Valores nulos al leer por Bluetooth QT

Tras mucho tiempo implementando e intentando dar con el error de por qué no se leía ninguna characteristic del dispositivo pese a estar detectándolas y estar correctamente conectado, se debía a dos factores.

El primero estar haciendo uso de una librería anterior a la documentación con la que estaba trabajando(Librería de QLowEnergyService). Hay grandes diferencias en el comportamiento de algunos métodos de esta librería de las versiones 5.x a la 6.x, aunque por desgracia, estas no provocan errores, hacen que el el código no funcione como se esperaría(Enums con valores diferentes o inexistentes, etc).
En segundo lugar estaba llamando a un método cuando todavía no se había recibido la characteristic. Por tanto esta aparentaba estar bien registrada, ya que podía obtener su Uuid, pero no contenía ningún valor. Estaba leyendo en connectToService() y no en serviceDetailsDiscovered().

He instalado ArduinoBLE.h para probar sketch de harvard
https://create.arduino.cc/projecthub/sibo_gao/harry-potter-magic-wand-384477

Orientación para vTensorFlow no Harvard: https://github.com/andriyadi/MagicWand-TFLite-Arduin
blob/master/src/accelerometer_handler.cpp

Para modo pizarra: https://github.com/petewarden/magic_wand/blob/main/website/index.html. https://tinyml.seas.harvard.edu/magic_wand/

Capítulo 4

Intento Harvard

Comienzo ejecutando tal cual el código del repo https://github.com/petewarden/magic_wand.

Importante tener la biblioteca de tensorflow actualizada(en este caso la provista por pete warden).

Visto cómo toma pete los datos, pruebo a hacerlo. Cargar el programa de recolección de datos en la placa(creo que funciona el magic_wand, PROBAR). Acceder a la web de recolección. Recolectar. Para acceder a la web, hay que hacerlo con chrome y con la flag 'Experimental Web Platform features' activa en 'chrome://flags/'

Hay un error con el DataCollector, además de que asigna mal los índices(index) al borrar muestras, no guarda todas las muestras que tomas, da la sensación de que hay un límite. Para contar el número de muestras que hay en el JSON, uso la siguiente línea bash:

```
~$ grep -o index <nombre_archivo>.json | wc -l
```

CAMBIAR ORIENTACIÓN DE LA PLACA

Como no hay referencias de la orientación original de los sensores en ningún datasheet, con el propio sketch que hice originalmente para tomar las muestras, comprobé la posición con la que funciona, la comparé con la posición que quiero tener y gracias a eso pude hacer el siguiente cambio.
Lo mejor que he conseguido cambiando en la biblioteca de los sensores, en LSM9DS1.cpp en todas los sensores:

```
//            Original            //      Orientacion cambiada
x = data[0] * 4.0 / 32768.0; // z = -data[0] * 4.0 / 32768.0;
y = data[1] * 4.0 / 32768.0; // y =  data[1] * 4.0 / 32768.0;
z = data[2] * 4.0 / 32768.0; // x =  data[2] * 4.0 / 32768.0;
```

Creo que habría que cambiar algo de deep_pen.ino(a partir de línea 408)

```
1    const float gy = current_gravity[1];
2    const float gz = current_gravity[2];
3    float gmag = sqrtf((gy * gy) + (gz * gz));
4    if (gmag < 0.0001f) {
5      gmag = 0.0001f;
6    }
7    // ...
```

LA OBTENCIÓN DE DATOS TIENE UN FALLO AL BORRAR MUESTRAS

Este fallo aparentemente se produce al borrar instancias por debajo de la última. En ocasiones, se produce un pequeño error en las comprobaciones de índices de muestras, por lo que se borran varias muestras y estas terminan con índices desordenados.
Para paliar este problema, he creado un pequeño script que toma el número de muestras y ordena los índices de las mismas.

MODELO

Se trata de un modelo de aprendizaje supervisado, es decir, que está basado en etiquetas(*labels*) estas etiquetas representarán las distintas soluciones a las que hará frente el modelo; en nuestro caso, letras. La forma que emplearemos para representar dichas letras como input para el modelo, serán imágenes. Aunque para tomar dichas imágenes, hacemos uso del giroscopio y el acelerómetro de la placa.

LEER DEL PUERTO SERIE EN QT

Hay que añadir al *.pro del proyecto QT:

```
1  QT += serialport
```

Y tras esto, añadir la biblioteca con normalidad y acceder al puerto con el nombre, en nuestro caso, "ttyACM0".

FALLO RECONOCIMIENTO DE IMÁGENES QT

Al importar imágenes en QT tras haber exportado el programa a Win y Linux para probar que todo fuera correctamente, las imágenes dejaron de mostrarse.
Esto se debe a que cambió la ruta del proyecto al directorio en el que se exportó. Por tanto las rutas especificadas para las imágenes, dejaron de ser válidas. Para corregir este problema, basta con cambiar el 'Build directory' del proyecto(Desde 'Projects' en el panel de la derecha del QT creator).

HEBRAS QT

Para la lectura del puerto serie desde el programa, necesitamos imporat

VISOR WEB QT

Para hacer empotrar html en qt, importamos las bibliotecas de QWeb. Para ello, necesi-

tamos aantes instlar todo el QtWebKit siguiendo las instrucciones: https://github.com/OpenBoard-org/OpenBoard/wiki/Build-OpenBoard-on-Ubuntu-20.04

VINCULAR PUERTO SERIE A PLACA

Para ello, haré uso de las reglas de udev del sistema linux. Lo primero es conocer la información de nuestra placa:

```
1  ~$ lsusb
2  [...]
3  Bus 003 Device 004: ID 2341:805a Arduino SA Nano 33 BLE
4  [...]
```

De aquí podemos extraer el fabricante(o idVendor) y el producto de este fabricante (o idProduct). Necesitaremos ambos.

Ahora necesitamos el serial del puerto al que vincularemos la placa:

```
1  ~$ udevadm info -a -n /dev/ttyACM0 | grep serial
2  ATTRS{serial}=="185F25FD3EF48040"
```

Con esto, ya podemos crear la regla para vincular univocamente el puerto a la placa y evitar así problemas con la detección en el UI.
En /etc/udev/rules.d/99-ftdi.rules (en mi caso, varía dependiendo del sistema), crearemos la regla
SUBSYSTEM=="tty", ATTRSidVendor=="2341", ATTRSidProduct=="805a", ATTRSserial=="185F25FD3EF48040", SYMLINK+="ttySLAB0"
por los datos obtenidos.

CAMBIOS MODELO

Cambiar el tamaño del kernel de las capas Conv2D de 3 a 4, resulta muy efectivo, a costa, evidentemente, de aumentar el tamaño que ocupa el modelo.
Al pasar de 4 a 5, el modelo arroja unos datos de efectividad teóricos extremadamente buenos, alcanzando cifras de precisión mucho más altas con menos epochs. En general, podemos extrapolar que, a mayor tamaño del kernel, mejor precisión pero mayor tamaño del modelo. En nuestro caso no llega a ser un problema, ya que, aunque estamos usando un dispositivo de memoria limitada, no llega a ocuparse toda la memoria del mismo, al menos por ahora.

PRUEBAS BLUETOOTH

He tenido que instalar qtconnectivity5-dev para probar el QT project que estoy probando.

POR QUÉ ESTA PLACA?

Aduino Nano Sense 33 BLE
Por qué arduino: Documentación, respaldo comunidad, IDE facilita trabajo, etc.
Por qué Nano: Queremos integrarla en un 'lápiz', debe ser un dispositivo pequeño.
Por qué Sense: Necesitamos los sensores para el reconocimiento de movimiento.
BLE: Por pura utilidad, es mucho más cómodo utilizar el lápiz de forma inalámbrica. Además

no tiene mucho sentido integrar el procesamiento en un dispositivo pequeño si va a depender de un ordenador.

DESCONEXIÓN DEL PROGRAMA A BT AL ESCRIBIR PLACA EN RX

Para el control de flujo de la comunicación bluetooth(asegurarse de que recibimos bien y solo una vez cada letra), implemento un sistema de señales. Cuando recibimos una letra desde la placa(mediante canal tx), el programa la almacena y envía una señal de que ha recibido la letra(mediante canal rx) y cuando la placa recibe esta señal, borra del canal tx la letra para que no vuelva a leerse desde el programa. El problema descrito, viene, creo, al escribir en rx la señal de recibo. Por algún motivo, el programa se desconecta de la placa.

MÉTODO DE ENVÍO DE BUFFER DE LETRAS PARA CONEXIÓN BT CUANDO HAYA UNA CADENA PREVIA A CONEXIÓN

Documentación Bluetooth LE: https://doc.qt.io/qt-6/qtbluetooth-le-overview.html

Visualizar red neuronal : http://alexlenail.me/NN-SVG/index.html

Documentación sobre Keras: https://keras.io/api/

Capítulo 5

Herramientas

1. **QT Designer**(Licencia de Codigo Abierto): Bocetado de la interfaz de usuario.
2. **QT Creator**(Licencia de Codigo Abierto): Implementación de la interfaz de usuario. (v6.2.4-Linux | v6.2.2-Windows)
3. **Arduino IDE**: Desarrollo del software para el DeepPen.
4. **Visual Studio Code + Latex**: Creación de la memoria.
5. **Bibliotecas**:
   TensorFlowLite(Arduino).
   Arduino_LSM9DS1(Arduino).
   libglu1-mesa-dev (Para QT linux).

Capítulo 6

Introducción

Este proyecto es software libre, y está liberado con la licencia [5].

Capítulo 7

Descripción del problema

Capítulo 8

Planificación

8.1. Metodología utilizada

8.2. Temporización

8.3. Seguimiento del desarrollo

Capítulo 9

Análisis del problema

Capítulo 10

Implementación

La implementación del software se ha dividido en hitos. Estos, han sido definidos en Github y cada uno de ellos contiene un grupo de *issues* que se corresponden con las distintas mejoras que se han ido incorporando al software a lo largo de su desarrollo.

Capítulo 11

Conclusiones y trabajos futuros

Bibliografía

[1] Arduino. Biblioteca arduino_apds9960, que contiene herramientas para sensores extras. https://github.com/arduino-libraries/Arduino_APDS9960.

[2] Arduino. Biblioteca arduino_cmsis-dsp, que provee de la biblioteca arm_math.h. https://github.com/arduino-libraries/Arduino_CMSIS-DSP.

[3] Arduino. Biblioteca arduino_hts221, herramientas para el sensor de temperatura y humedad. Arduino_HTS221.

[4] Arduino. Biblioteca arduino_lps22hb herramientas para el sensor de presión. https://github.com/arduino-libraries/Arduino_LPS22HB.

[5] Free Software Foundation. GNU General Public License. http://www.gnu.org/licenses/gpl.html.

[6] TensorFlow. Github ejemplo magic_wand de tensorflow. https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/magic_wand.

[7] TensorFlow. Introducción a los microcontroladores por tensorflow. https://www.tensorflow.org/lite/microcontrollers/get_started_low_level.