



Trabajo Práctico Integrador

Alumno

Mauricio Barroso Benavides

Docentes:

Leonardo Carducci

Sebastián García Marra

Tabla de contenido

Registros de cambios	3
1. Descripción de la implementación del sistema embebido	4
2. Aplicación Linear MQTT Dashboard	9
3. Broker MQTT	12
4. Mediciones	13
4.1 Estacionario	13
4.2 Periódico	17
4.3 Aleatorio	18
5. Conclusiones	22

Registros de cambios

Revisión	Detalle de los cambios realizados	Fecha
1.0	Creación del documento	06/05/2022

1. Descripción de la implementación del sistema embebido

El trabajo consiste en la implementación de un sistema para obtener los datos de un acelerómetro, guardarlos en una memoria no volátil y posteriormente analizarlos con ayuda de software especializado. La señalización de inicio, fin y alarmas del sistema se realiza mediante el protocolo MQTT.

La obtención de datos se realiza con ayuda de un sistema embebido compuesto por una tarjeta ESP32-S2-SAOLA-1 y un MPU6050 que intercambian datos a través del protocolo I2C. La tarjeta ESP32-S2-SAOLA-1 posee como núcleo un SoC (System on a Chip) que integra un microcontrolador mono núcleo de 32 bits con arquitectura Xtensa LX7 a 240 MHz, soporta Wi-Fi 802.11 b/g/n, incorpora también tiene diversos periféricos para comunicaciones, bajo consumo y seguridad de datos. El MPU6050 es un IMU (Inertial Measurement Units) de 6 grados de libertad que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes, tiene comunicación I2C y puede generar interrupciones a través de 1 GPIO. En la figura 1 se muestra el diagrama de bloques del sistema para la obtención de datos.

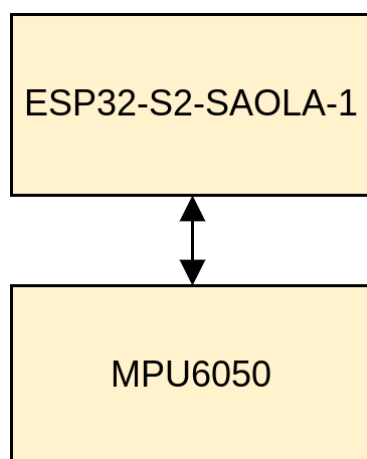


Figura 1. Diagrama en bloques del sistema embebido

El sistema embebido descrito anteriormente se montó sobre una breadboard como se observa en la figura 2.

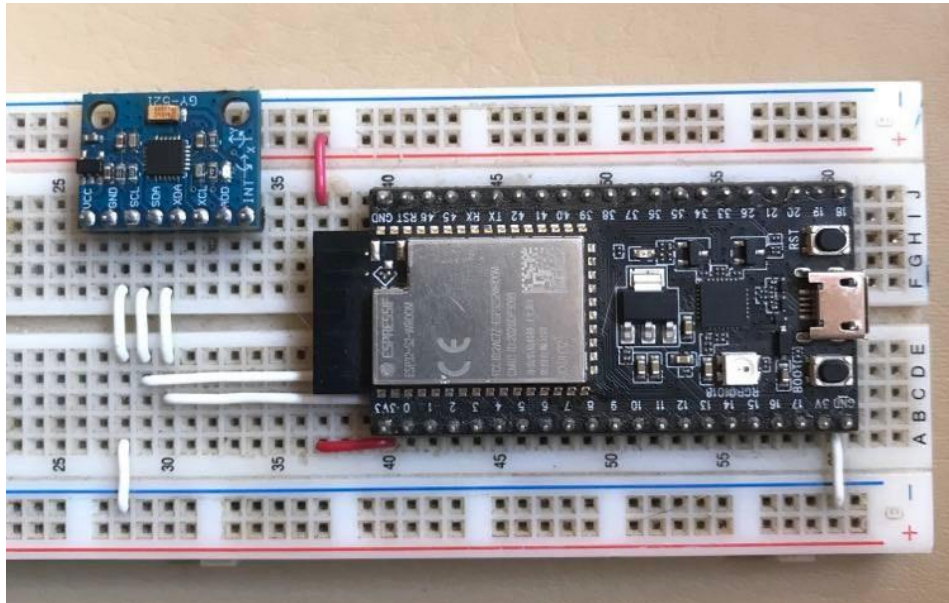


Figura 2. Fotografía del sistema embebido

Para los fines de este trabajo solo se requiere sólo 1 de los 3 datos que puede generar el acelerómetro del MPU6050, en este caso el eje Z. Los datos obtenidos del MPU6050 generan archivos de tipo .CSV que son almacenados en un sistema de archivos que se encuentra implementado en la memoria flash externa que posee el ESP32-S2-SAOLA-1. Este tipo de sistema de archivos se conoce como SPIFFS (SPI Flash File System). Como el ESP32-S2-SAOLA-1 posee una memoria flash externa de 4 MB, el sistema de archivos generado tiene un tamaño de 2 MB y puede almacenar hasta 30 archivos de 60000 muestras cada uno.

Para obtener los archivos generados durante las mediciones, se implementó un servidor HTTP para descargar el archivo de registro `log.txt` y los archivos .CSV de las mediciones. De esta forma se simplifica el proceso para manipular estos archivos y se automatiza un poco el proceso de análisis posterior. Para la lectura y descarga de estos archivos se desarrolló un script nombrado `download_node_files.sh` que se encuentra en la carpeta de `scripts`.

Se utilizó el protocolo MQTT para el intercambio de mensajes entre el sistema embebido y la aplicación que controla las mediciones. Para facilitar el reconocimiento de clientes se establecieron los siguientes tópicos:

- Tópico de recepción: `data/app`
- Tópico de transmisión: `data/node`

De esta manera se establece que el sistema embebido (nodo) debe suscribirse a la información que publica la aplicación y publicar a un tópico con su identificador para que la aplicación y el broker conozcan de dónde vienen los mensajes.

El firmware implementado contempla el desarrollo de un driver propio para el MPU6050 y la utilización del SDK ESP-IDF conjuntamente con FreeRTOS para el manejo de tareas, eventos, colas de mensajes, etc. Los componentes que forman parte del firmware se representan a través de la figura 3 y en la figura 4 el diagrama de tiempos del RTOS.

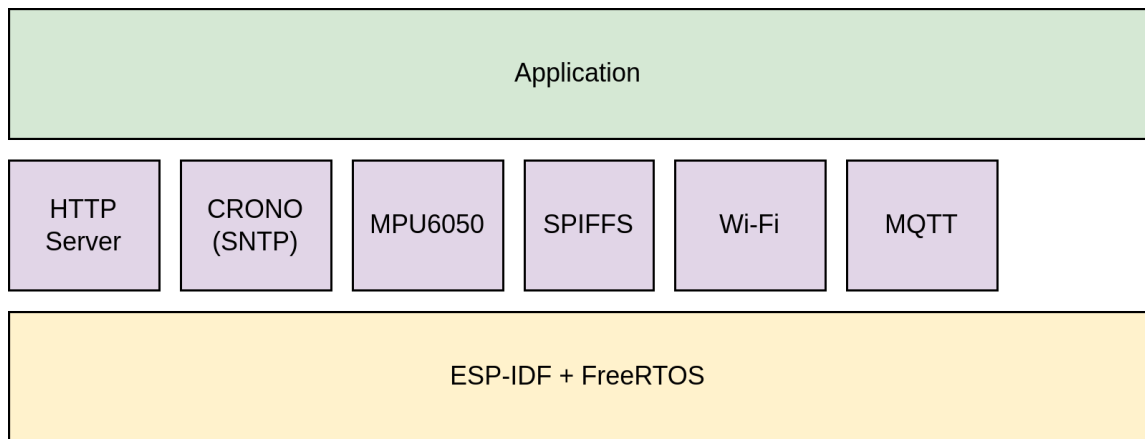


Figura 3. Diagrama de componentes del firmware

Como entorno de desarrollo se utilizó Eclipse en conjunto con el plugin de ESP-IDF provisto por Espressif. Para el análisis y debug del código se utilizó en gran medida el monitor de ESP-IDF, que proporciona mucha información útil a través de texto en una consola. En las figuras 4, 5 y 6 se observan algunas capturas de pantalla del monitor de ESP-IDF.

```
I (21) boot: ESP-IDF v5.0-dev-2349-g4350e6fef8 2nd stage bootloader
I (21) boot: compile time 01:00:31
I (21) boot: chip revision: 0
I (25) boot.esp32s2: SPI Speed      : 80MHz
I (30) boot.esp32s2: SPI Mode      : DIO
I (35) boot.esp32s2: SPI Flash Size : 4MB
I (40) boot: Enabling RNG early entropy source...
I (45) boot: Partition Table:
I (49) boot: ## Label                Usage            Type ST Offset   Length
I (56) boot:  0 nvs                  WiFi data        01 02 00009000 00006000
I (63) boot:  1 phy_init             RF data          01 01 0000f000 00001000
I (71) boot:  2 factory              factory app       00 00 00010000 00100000
I (78) boot:  3 storage              Unknown data      01 82 00110000 00200000
I (86) boot: End of partition table
I (90) esp_image: segment 0: paddr=00010020 vaddr=3f000020 size=271f4h (160244) map
I (130) esp_image: segment 1: paddr=0003721c vaddr=3fffc6890 size=035a8h ( 13736) load
I (134) esp_image: segment 2: paddr=0003a7cc vaddr=40022000 size=0584ch ( 22604) load
I (142) esp_image: segment 3: paddr=00040020 vaddr=40080020 size=9a924h (633124) map
I (271) esp_image: segment 4: paddr=000da94c vaddr=4002784c size=0f03ch ( 61500) load
I (286) esp_image: segment 5: paddr=000e9990 vaddr=50000000 size=00014h (   20) load
I (297) boot: Loaded app from partition at offset 0x10000
I (297) boot: Disabling RNG early entropy source...
I (309) cachefw~000~Ug000cache : size 8KB, 4Ways, cache line size 32Byte
I (309) cpu_start: Pro cpu up.
I (321) cpu_start: Pro cpu start user code
I (321) cpu_start: cpu freq: 240000000 Hz
I (321) cpu_start: Application information:
I (325) cpu_start: Project name:      sed-tpf
I (330) cpu_start: App version:      97ab84e-dirty
I (336) cpu_start: Compile time:     May  6 2022 01:20:43
I (342) cpu_start: ELF file SHA256:  cb6e3f5bd52c56a1...
Warning: checksum mismatch between flashed and built applications. Checksum of built application is 674437b4fd469930453e226ca59e77baffd3b2ef1f11cc6c51d37d6bb490a23f
I (348) cpu_start: ESP-IDF:          v5.0-dev-2349-g4350e6fef8
I (355) heap_init: Initializing. RAM available for dynamic allocation:
I (362) heap_init: At 3FFCDE20 len 0002E1E0 (184 KiB): DRAM
I (368) heap_init: At 3FFFC000 len 00003A10 (14 KiB): DRAM
I (374) heap_init: At 3FF9E000 len 00002000 (8 KiB): RTCRAM
I (381) spi_flash: detected chip: generic
I (385) spi_flash: flash io: dio
I (393) cpu_start: Starting scheduler on PRO CPU.
I (395) app: Initializing I2C...
I (399) mpu6050: Initializing mpu6050 component...
I (410) app: Initializing NVS...
I (415) app: Initializing Wi-Fi...
I (418) wifi:wifi driver task: 3ffd6b78, prio:23, stack:6656, core=0
I (418) system_api: Base MAC address is not set
I (423) system_api: read default base MAC address from EFUSE
```

Figura 4. Captura de pantalla del monitor - Inicialización del sistema

```
I (623) wifi:Init max length of beacon: 752/752
I (628) app: WIFI_EVENT_STA_START
I (632) app: Other Wi-Fi event
I (636) app: Initializing MQTT...
I (3493) wifi:ap channel adjust o:1,1 n:2,1
I (3493) wifi:new:<2,1>, old:<1,1>, ap:<2,1>, sta:<2,1>, prof:1
I (5080) wifi:state: init -> auth (b0)
I (5133) wifi:state: auth -> assoc (0)
I (5183) wifi:state: assoc -> run (10)
I (5231) wifi:connected with test, aid = 1, channel 2, 40U, bssid = b8:05:ab:e6:0a:63
I (5232) wifi:security: WPA2-PSK, phy: bgn, rssi: -47
I (5235) wifi:pm start, type: 1

I (5239) app: WIFI_EVENT_STA_CONNECTED
W (5240) wifi:<ba-add>idx:0 (ifx:0, b8:05:ab:e6:0a:63), tid:0, ssn:0, winSize:64
I (5296) wifi:AP's beacon interval = 102400 us, DTIM period = 1
I (6514) esp_netif_handlers: sta ip: 192.168.1.119, mask: 255.255.255.0, gw: 192.168.1.1
I (6514) app: IP_EVENT_STA_GOT_IP
I (6516) CRONO/SNTP: Boot count: 1
I (6520) CRONO/SNTP: Initializing SNTP
I (6524) CRONO/SNTP: Waiting for system time to be set... (1/10)
I (8531) CRONO/SNTP: Waiting for system time to be set... (2/10)
W (10373) wifi:<ba-add>idx:1 (ifx:0, b8:05:ab:e6:0a:63), tid:2, ssn:0, winSize:64
I (10561) CRONO/SNTP: The current date/time in Buenos Aires is: Fri May 6 12:43:02 2022
I (10561) app: Other MQTT event
I (10617) app: MQTT_EVENT_CONNECTED
I (10619) app: Sent subscribe successful to data/app topic
I (10623) app: Other MQTT event
I (25332) app: Initializing SPIFFS...
I (25373) app: Partition size: total: 1920401, used: 0
I (25374) file server: Starting HTTP Server on port: '80'
I (25376) app: sampling rate in ms is 10
I (25379) app: threshold0 is 1.596667
I (25383) app: threshold1 is 1.940000
I (25388) app: threshold2 is 2.970000
I (1689562) app: MQTT_EVENT_DATA
I (1689563) app: data=1
```

Figura 5. Captura de pantalla del monitor - Evento de Wi-Fi, MQTT y SNTP


```
timestamp:4420, acce_z:-1.83
timestamp:4430, acce_z:-2.39
timestamp:4440, acce_z:-2.73
timestamp:4450, acce_z:-2.83
timestamp:4460, acce_z:-2.85
timestamp:4470, acce_z:-2.83
timestamp:4480, acce_z:-2.78
timestamp:4490, acce_z:-2.67
timestamp:4500, acce_z:-2.40
THRESHOLD2
timestamp:4510, acce_z:-1.96
timestamp:4520, acce_z:-1.41
timestamp:4530, acce_z:-0.77
timestamp:4540, acce_z:-0.15
timestamp:4550, acce_z:0.51
timestamp:4560, acce_z:1.29
timestamp:4570, acce_z:2.14
timestamp:4580, acce_z:2.90
timestamp:4590, acce_z:3.63
timestamp:4600, acce_z:4.00
timestamp:4610, acce_z:4.00
timestamp:4620, acce_z:4.00
timestamp:4630, acce_z:4.00
timestamp:4640, acce_z:4.00
timestamp:4650, acce_z:4.00
timestamp:4660, acce_z:4.00
timestamp:4670, acce_z:3.62
timestamp:4680, acce_z:2.80
```

Figura 6. Captura de pantalla del monitor - Obtención de datos del MPU6050 y alarmas

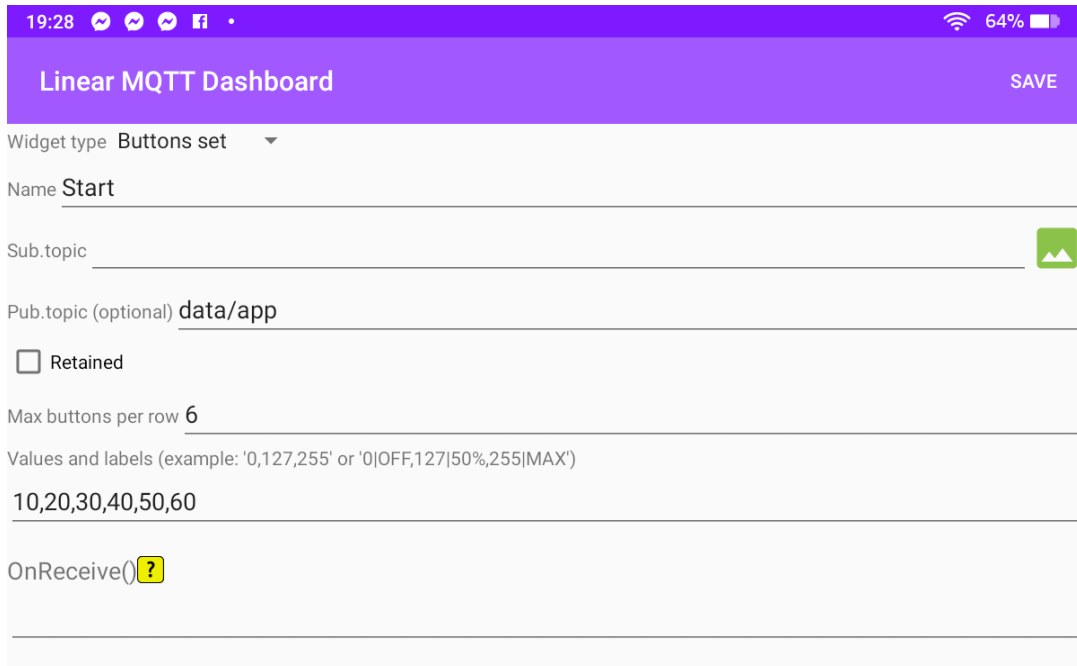
2. Aplicación Linear MQTT Dashboard

Linear MQTT Dashboard es una aplicación para Android que permite desarrollar dashboards compuestos por indicadores visuales y elementos de acción como botones. Los componentes de un dashboard pueden ser configurados para suscribirse y/o publicar a un broker en particular.

Para este trabajo práctico el dashboard se compone de lo siguiente:

- 3 LEDs para indicar los thresholds del sistema
- 6 botones con diferentes valores de tiempo de medición, desde 10 segundos hasta 60 segundos
- 1 botón para parar la medición
- 1 botón para apagar los LEDs indicadores de los thresholds

En las figuras 7, 8, 9, y 10 se observan las configuraciones de los componentes del dashboard.



19:28 64%

Linear MQTT Dashboard SAVE

Widget type **Buttons set**

Name **Start**

Sub.topic

Pub.topic (optional) **data/app**

☐ Retained

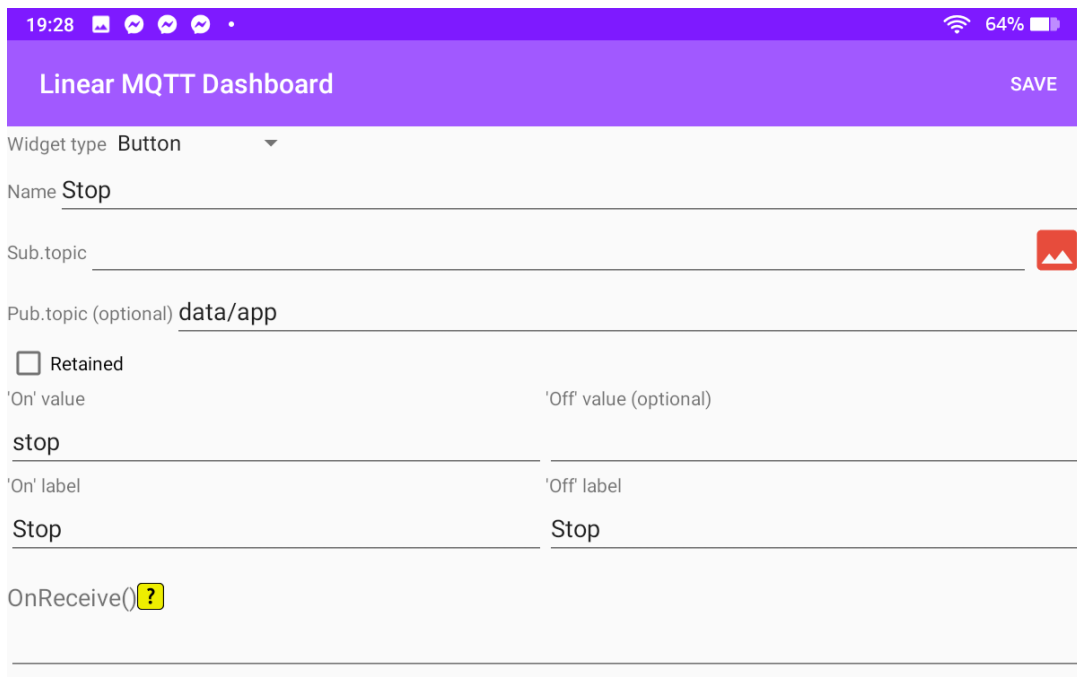
Max buttons per row **6**

Values and labels (example: '0,127,255' or '0|OFF,127|50%,255|MAX')

10,20,30,40,50,60

OnReceive() ?

Figura 7. Captura de pantalla de la aplicación - Configuración de los botones de inicio



19:28 64%

Linear MQTT Dashboard SAVE

Widget type **Button**

Name **Stop**

Sub.topic

Pub.topic (optional) **data/app**

☐ Retained

'On' value **stop**

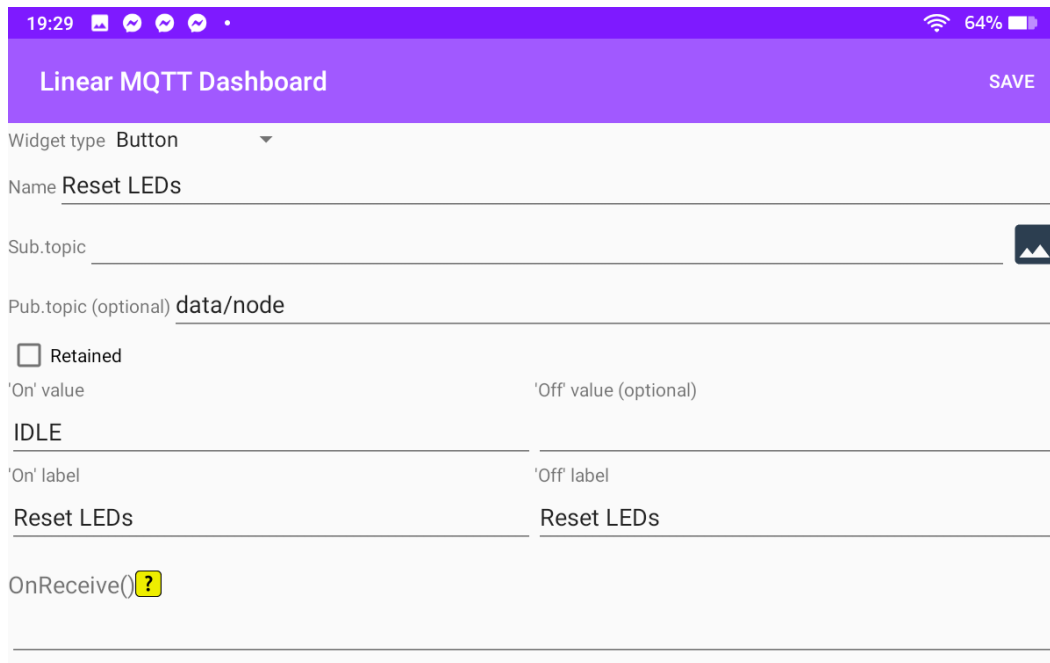
'Off' value (optional)

'On' label **Stop**

'Off' label **Stop**

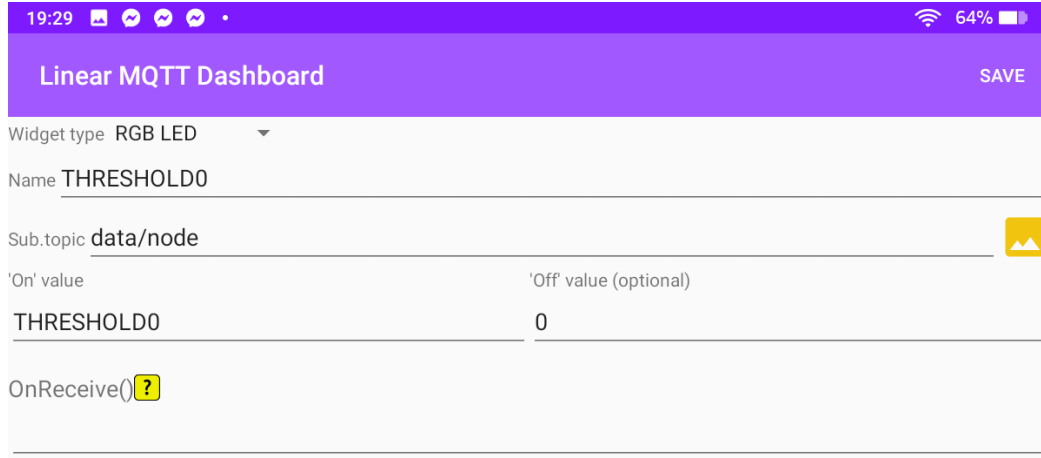
OnReceive() ?

Figura 8. Captura de pantalla de la aplicación - Configuración del botón de parada



The screenshot shows the 'Linear MQTT Dashboard' interface. At the top, there's a purple header with the title and a 'SAVE' button. Below the header, the 'Widget type' is set to 'Button'. The 'Name' field is 'Reset LEDs'. The 'Sub.topic' field is empty, and the 'Pub.topic (optional)' field is 'data/node'. There's a 'Retained' checkbox which is unchecked. The 'On' value is 'IDLE', and the 'Off' value (optional) is empty. The 'On' label is 'Reset LEDs', and the 'Off' label is also 'Reset LEDs'. At the bottom, there's a code editor with 'OnReceive()' and a yellow question mark icon.

Figura 9. Captura de pantalla de la aplicación - Configuración del botón de apagado de los LEDs



The screenshot shows the 'Linear MQTT Dashboard' interface. At the top, there's a purple header with the title and a 'SAVE' button. Below the header, the 'Widget type' is set to 'RGB LED'. The 'Name' field is 'THRESHOLD0'. The 'Sub.topic' field is 'data/node'. The 'On' value is 'THRESHOLD0', and the 'Off' value (optional) is '0'. At the bottom, there's a code editor with 'OnReceive()' and a yellow question mark icon.

Figura 10. Captura de pantalla de la aplicación - Configuración de uno de los LEDs de threshold

Con el sistema embebido funcionando y la aplicación configurada la medición se inicia con uno de los 6 botones de inicio y puede ser detenida con el botón de parada en cualquier momento. Los LEDs indican 3 niveles de threshold definidos en el sistema embebido y tienen el siguiente código de colores:

- THRESHOLD0: Amarillo (bajo)
- THRESHOLD1: Naranja (medio)
- THRESHOLD2: Rojo (alto)

En la figura 11 se observa el dashboard funcionando y con uno de los LEDs encendidos.

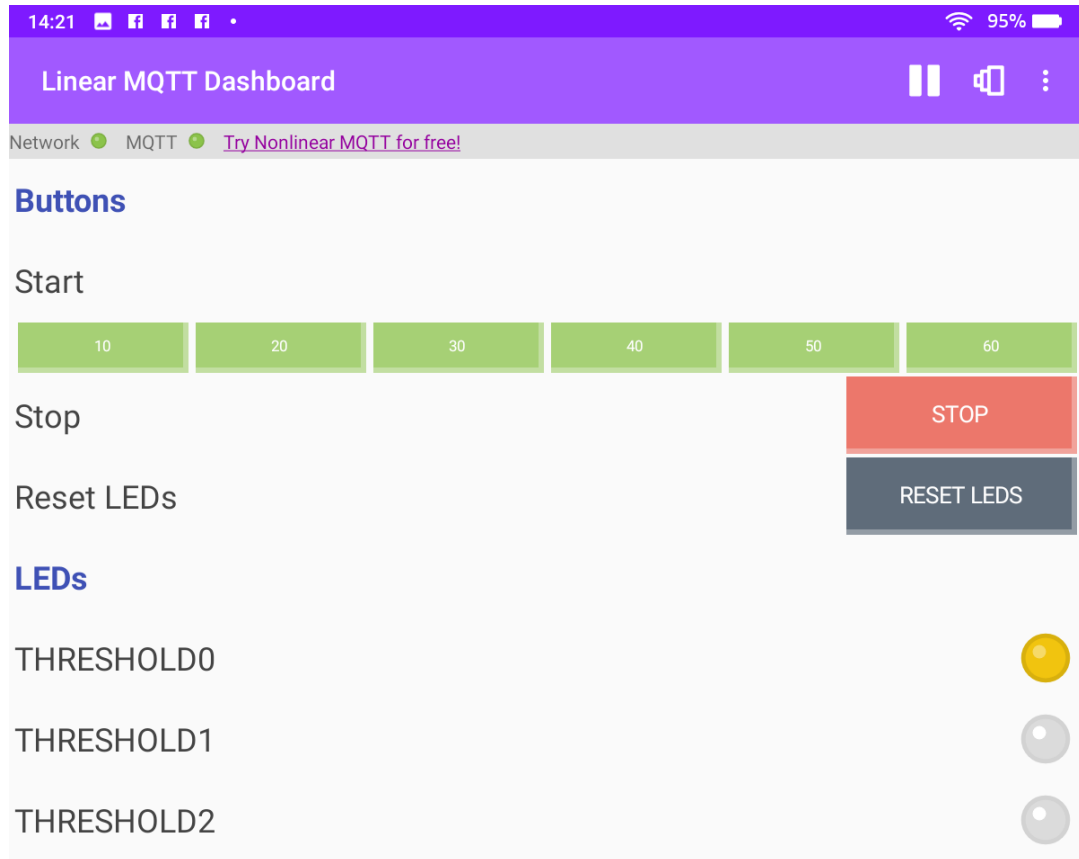


Figura 11. Captura de pantalla de la aplicación - Configuración de uno de los LEDs de threshold

3. Broker MQTT

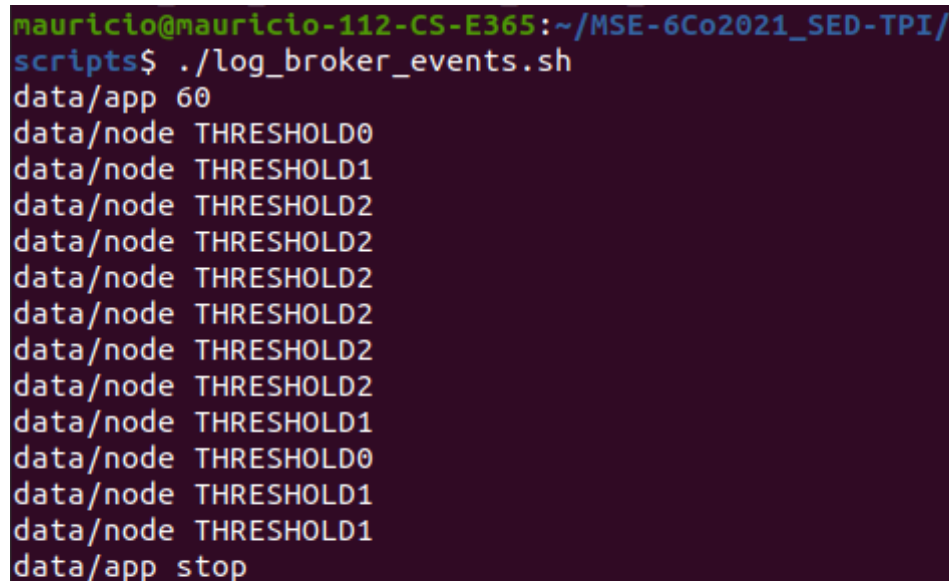
El broker MQTT utilizado en este trabajo fue Mosquitto que se ejecutó de manera local en una PC para el intercambio de mensajes entre el nodo y la aplicación.

Para permitir que clientes MQTT externos puedan utilizar el broker fue necesaria la creación del archivo `listener.conf` de configuración en la ruta `/etc/mosquitto/conf.d`. En la figura 12 se observan las configuraciones dentro del archivo.

```
GNU nano 4.8 listener.conf
listener 1883
allow_anonymous true
```

Figura 12. Captura de pantalla de la PC - Archivo de configuración de Mosquitto

Para llevar un control y registro del tráfico de mensajes MQTT entre el nodo y la aplicación se desarrolló un script que genera un archivo de texto que contiene los mensajes entrantes y salientes, así como su timestamp. El script se encuentra en la carpeta `scripts` y se llama `log_broker_events.sh`. En la figura 13 se observa una captura de pantalla del script siendo ejecutado.



```
mauricio@mauricio-112-CS-E365:~/MSE-6Co2021_SED-TPI/  
scripts$ ./log_broker_events.sh  
data/app 60  
data/node THRESHOLD0  
data/node THRESHOLD1  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD2  
data/node THRESHOLD1  
data/node THRESHOLD0  
data/node THRESHOLD1  
data/node THRESHOLD1  
data/app stop
```

Figura 13. Captura de pantalla de la PC - Script de log de mensajes MQTT

4. Mediciones

Se realizaron 3 tipos de mediciones distintas, cada una de 60 segundos con un tiempo de muestreo de 10 ms. La primera fue con sistema embebido estacionario, la segunda con perturbaciones periódicas y la tercera con perturbaciones aleatorias.

4.1 Estacionario

Para esta medición el sistema embebido no tuvo ningún tipo de perturbación durante el tiempo que se obtuvieron las muestras. Con un programa en Octave llamado `measurement1.m` que se encuentra en la carpeta `octave`, se analizaron los datos y obtuvieron las gráficas solicitadas.

La primera gráfica fue la de muestras vs tiempo que se muestra en la figura 14. En esta se puede observar la variación de los valores del eje Z del acelerómetro durante el tiempo de medición.

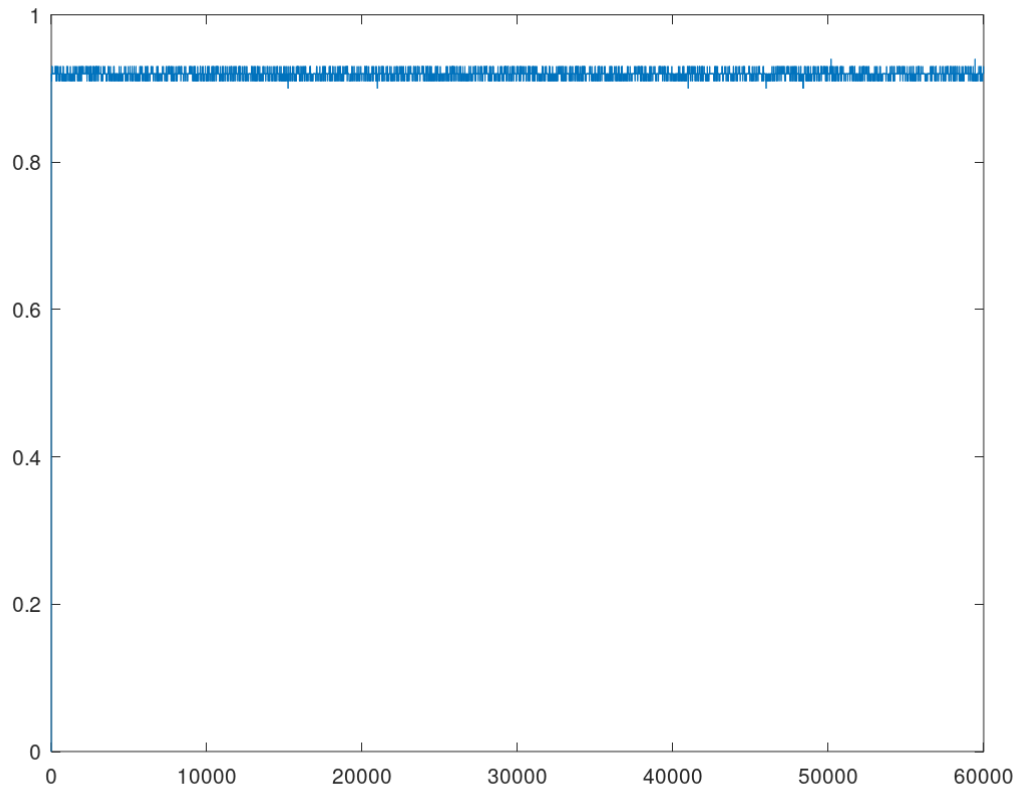


Figura 14. Medición 1 - Muestras vs Tiempo

Haciendo zoom para visualizar de mejor manera la amplitud de la señal se detectaron valores esporádicos que fueron eliminados antes de obtener los histogramas de la señal. En la figura 15 se observan estos valores como picos que aparecen de manera aleatoria.

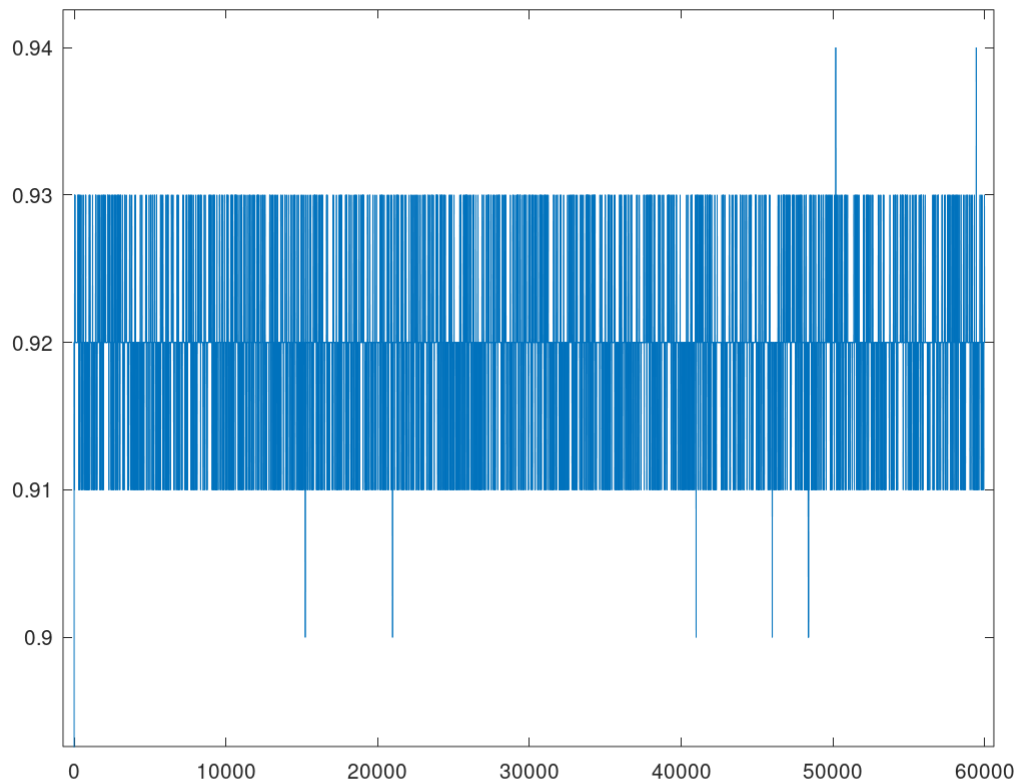


Figura 15. Medición 1 - Zoom a Muestras vs Tiempo

Con ayuda de las funciones `max` y `min` de Octave se filtraron estas muestras y se obtuvieron los histogramas para 10 bins y 50 bins. En las figuras 16 y 17 se muestran los histogramas para 10 y 50 bins, respectivamente.

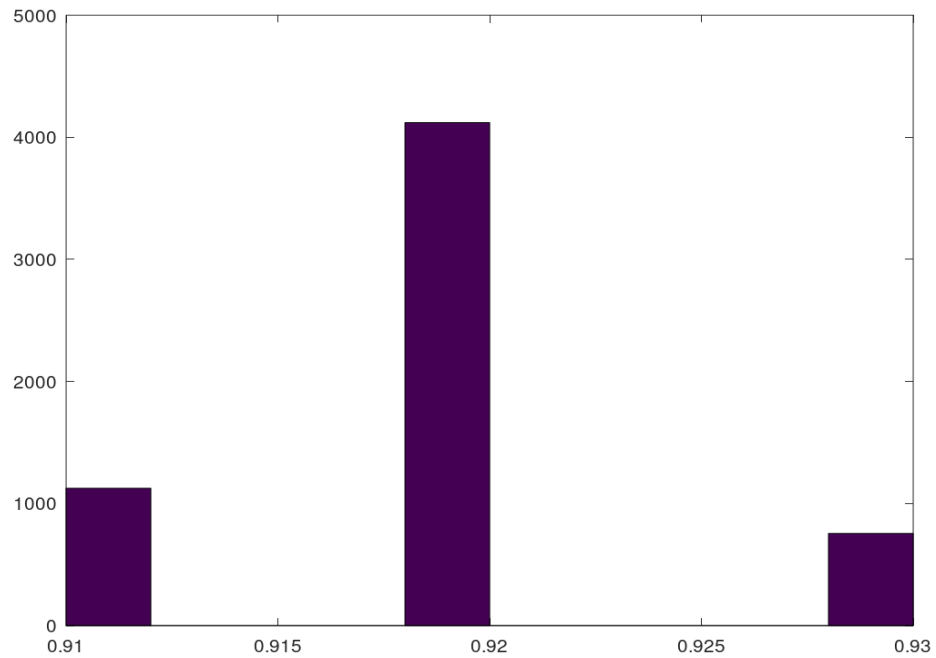


Figura 16. Medición 1 - Histograma 10 bins

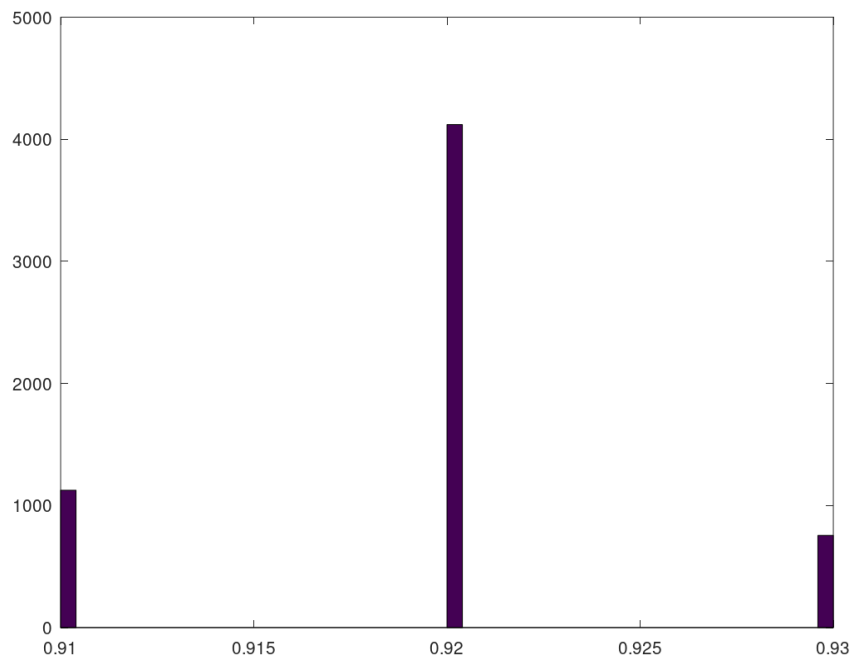


Figura 17. Medición 1 - Histograma 50 bins

No se encuentran diferencias entre ambos histogramas, esto porque la señal después de filtrarla solo posee 3 valores que podían haber sido representados tanto por 10 o 50 bins.

Como último requisito de esta medición se hallaron la media y la varianza de las muestras, con un valor de 0.91938 para la media y 0.000030958 para la varianza. Era de esperar un valor bajo de la varianza debido a la poca variabilidad de la señal.

4.2 Periódico

En esta medición se realizaron movimientos del sistema embebido durante 60 segundos. Los primeros movimientos fueron lentos para simular una frecuencia baja, y cada, aproximadamente 12 segundos, la velocidad de movimiento fue creciendo. Con un programa en Octave llamado `measurement2.m` que se encuentra en la carpeta `octave`, se analizaron los datos y obtuvieron las gráficas solicitadas.

En la figura 18 se muestra el gráfico de muestras vs tiempo, donde se observa claramente como no solo la frecuencia va aumentando si no también la intensidad del movimiento.

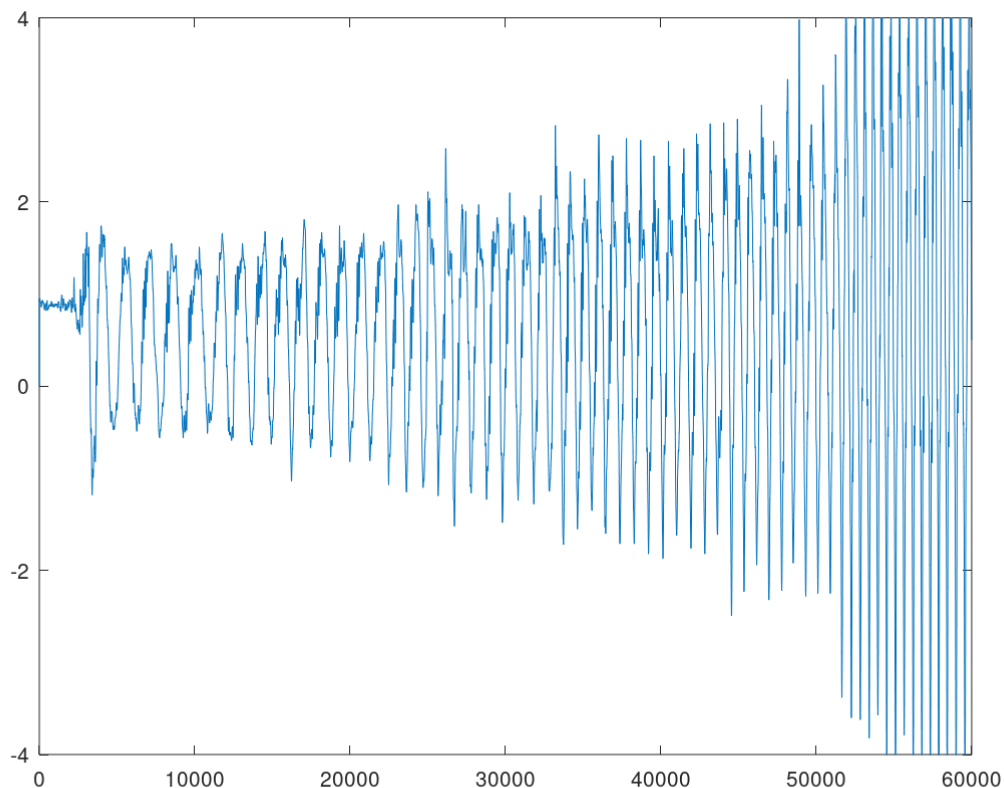


Figura 18. Medición 2 - Muestras vs Tiempo

Otro gráfico de interés fué el espectrograma de la señal anterior, este muestra en el dominio de la frecuencia la variación de la señal. En la figura 19 se observa el espectrograma mencionado.

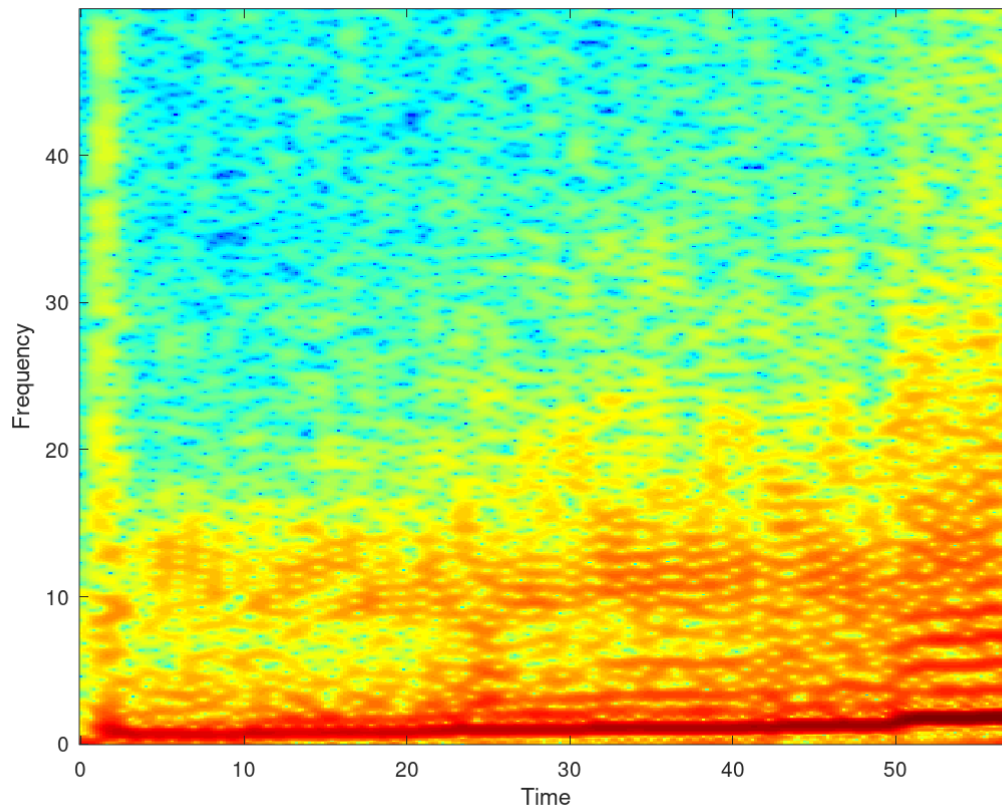


Figura 19. Medición 2 - Espectrograma

En el espectrograma se observa como la frecuencia varía desde aproximadamente 1 Hz hasta 10 Hz y que para los valores más altos de frecuencia alcanzados la intensidad de la señal es mayor.

4.3 Aleatorio

Para esta medición se posicionó el sistema embebido sobre un parlante de 42 W RMS y se reprodujo la canción Emmanuel's Groove de Azuro durante 60 segundos. En la figura 20 se puede observar una fotografía del experimento. Con un programa en Octave llamado `measurement3.m` que se encuentra en la carpeta `octave`, se analizaron los datos y obtuvieron las gráficas solicitadas.



Figura 20. Medición 3 - Fotografía del experimento

El primer gráfico que se obtuvo, como en las mediciones anteriores, fue el de muestras vs tiempo que se puede observar en la figura 21.

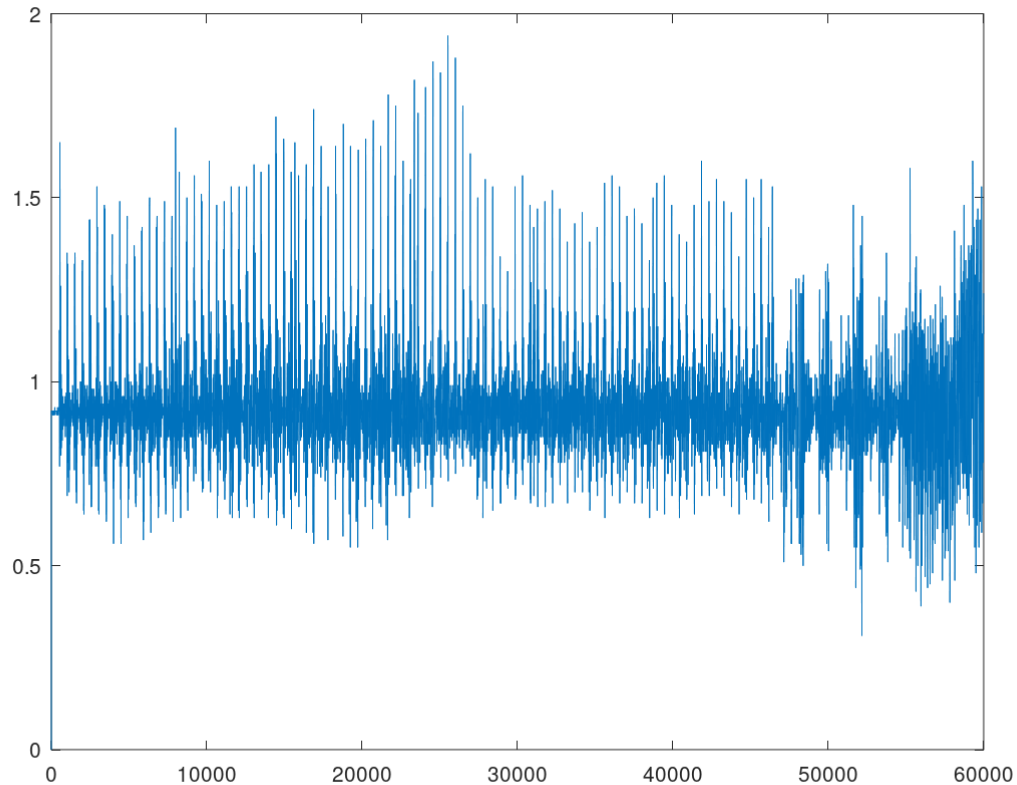


Figura 21. Medición 3 - Muestras vs Tiempo

Una cosa interesante que se puede obtener de esta medición son los BPM (Beats Per Minute) de la canción reproducida. Como los tonos bajos causan mayor vibración que los de frecuencias altas, este parámetro es fácilmente detectable. Para esto se hizo zoom sobre una sección de la señal y con ayuda de los marcadores del gráfico se pudo determinar que los BPM de la canción son aproximadamente 120. En la figura 22 se puede observar el zoom realizado a la señal medida.

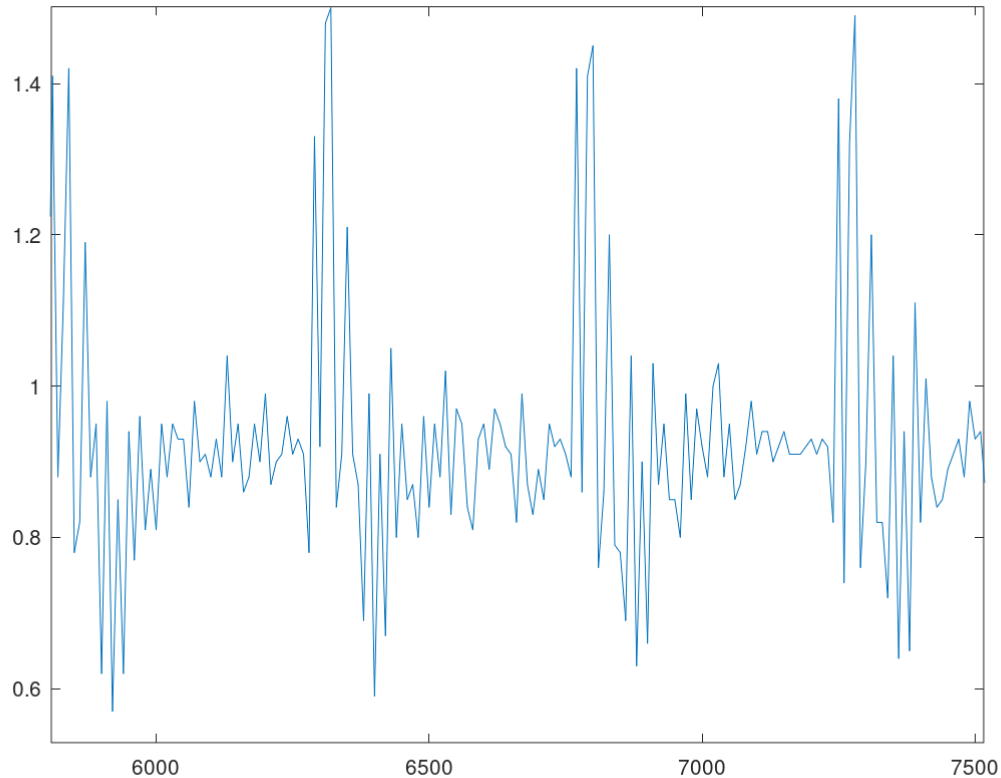


Figura 22. Medición 3 - Zoom a Muestras vs Tiempo

Para verificar los BPMs determinados anteriormente se obtuvo este dato de la página web tunebat.com y en la figura 23 se pueden observar los resultados de la página. Se evidencia que el dato medido con el sistema embebido es muy aproximado al consultado.

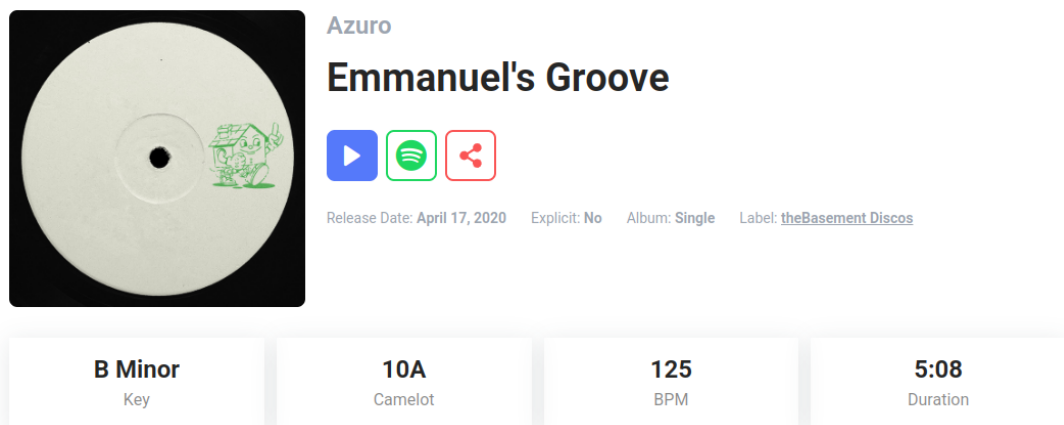


Figura 23. Medición 3 - BPM de la página tunebat.com

Finalmente, se realizó el gráfico del espectrograma de la señal, donde no se puede notar nada realmente sobresaliente. Se puede notar que las frecuencias obtenidas varían entre aproximadamente 2 Hz y 50 Hz. En la figura 24 se puede observar el espectrograma.

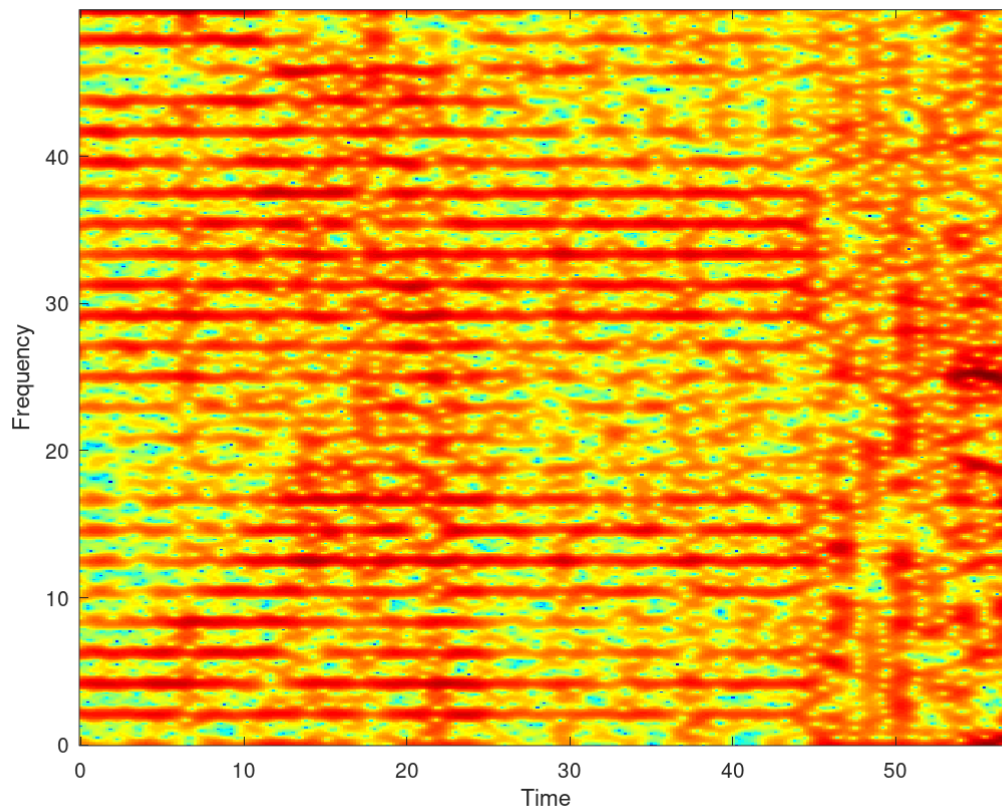


Figura 24. Medición 3 - Espectrograma

5. Conclusiones

- No se logró hacer funcionar el adaptador para tarjetas SD y por lo tanto se determinó utilizar un sistema de archivos SPIFFS para cumplir con la misma función.
- Se realizaron algunos ajustes de los parámetros de funcionamiento del ESP32-S2. Se aumentó la frecuencia de operación de 160 MHz a 240 MHz y también se redujo el tiempo de tick del sistema a 1 ms para poder cumplir con el requerimiento de muestreo a 100 Hz.
- Se tuvo que desarrollar un driver para obtener datos del MPU6050, que al principio causó un gran retraso, pero después facilitó mucho la obtención de datos. También el

tiempo invertido fue de utilidad por que el driver puede ser utilizado más adelante en otros proyectos.

- No se utilizaron los drivers para MQTT y Wi-Fi proporcionados por los profesores por que ya se contaba con código escrito anteriormente para este propósito.
- Se utilizó el driver para SNTP proporcionado por los profesores que funcionaba de gran manera y facilitó la obtención de los timestamps para el archivo `log.txt`.
- Octave simplificó en gran manera el análisis y la graficación de las señales. En un principio se planteó utilizar Python, pero las funciones simplificadas de Octave optimizaron mucho el tiempo de esta tarea.