Sistemas Embebidos Distribuidos
Informe
Trabajo Práctico Integrador

Campus de la Materia

Docentes: **Leonardo Carducci** <lcarducci@fi.uba.ar>

Sebastián García Marra <sebastianmarra@gmail.com>

Correo grupal: psf_m06@cursoscapse.com

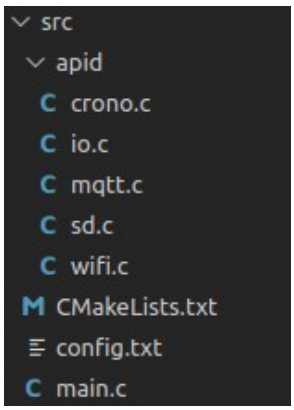
Campus del curso:

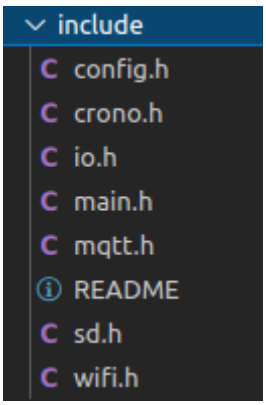
https://campus.fi.uba.ar/enrol/index.php?id=1462

Alumno: Pablo D. Folino pfolino@gmail.com

Breve descrición del proyecto

El programa se realizó en ***Visual Studio Code*** con ***Platformio***. Posee una carpeta **src**(source) en la cual se encuentran los códigos fuentes de los programas, y un directorio ***include*** en donde se encuentran las librerías.





En el archivo **config.h** se encuentran definidas todas las configuraciones principales del sistema:

```
/* configuración WIFI */
#define ESP_WIFI_SSID "ssid"      // CONFIG_ESP_WIFI_SSID  // "SSID wifi"
#define ESP_WIFI_PASS "pass"      // CONFIG_ESP_WIFI_PASS  // "pass wifi"

/*  Configuracion MQTT  */
#define PORT_MQTT 1883            //CONFIG_PORT_MQTT      // default
#define IP_BROKER_MQTT "boker"    //CONFIG_IP_BROKER_MQTT // Broker MQTT

/*  configuración IO */
#define BLINK_GPIO CONFIG_BLINK_GPIO  // port 2 para NodeMcu-23S

/* Configuración SD  */

/* Configuración CRONO  */
#define SENSOR_REP 1909
#define SENSOR_MAX 4095

/* Configuración varias */
#define LOOPS_UMBRAL  5           // Veces de 100mseg en testear los umbrales

#define TIME_MAX 60               // Tiempo máximo de muestreo en segundos
#define FREC_MUESTREO 100
#define PERIODO_MUESTREO  (1000/FREC_MUESTREO)  // Período de muestreo en ms
#define EPOCH_INICIAL 11111111    // Es e archivo A11111111.csv por default
```

```
/* Estructuras generales */
struct micro_t{
  uint8_t inicio;           // Se guarda el tiempo de mustra de 1 a 60 seg
  uint8_t run_stop;         // 0-Stop no se estan tomando muestras
                            // 1-Run  se estan tomando muestras
                            // 2-Paro el muestreo
                            // 3- Abro los el archivo y guardo las muestras
  uint16_t t_muestreo;      // período de muestreo se guarda el número de muestra
  uint16_t muestra;         // es el número de muestra
  uint64_t epoch_init;      // se guarda el tiempo cuando comienza a tomar la muestra
  uint64_t epoch_finish;    // se guarda el tiempo final de muestra

};

#define N_max 6200      // Maxima cantidad de datos a guardar
struct data_t{           //array de datos
  int64_t epochs;
  int16_t valor_adc;
};
```

Las configuraciones del wifi como las de broker MQTT se leen de un archivo config.txt como en el ejercicio 3 de la práctica de la materia.

En éste archivo se evalúan todas las constantes de tiempos.

Además se decleran un par de estructuras **micro_t** y **data_t**, co las cuales se maneja todo el sistema.

En **data_t** se guerdan las muestras tomadas del ADC.

En **micro_t** entre otras cosas hay una variable **run_stop** que se usa como estado del sistema.

Uno de los archivos principales es el **mqtt.c ,** tiene como función principal conectarse a broker MQTT de la PC, instanciar la tarea que atiende las subcripciones (**MQTT_userInit()**), indicar a qué tópicos se conecta (**MQTT_suscripciones()**).

A continuación se muestra el contenido de la función **MQTT_suscripciones():**

```
122 /*****************************************************************************
123 MQTT_suscripciones(): lee el mensaje MQTT recibido cuando se dispara el evento
124 *****************************************************************************/
125 void MQTT_suscripciones(void){
126
127         MQTT_subscribe("test/inicio");
128         MQTT_subscribe("test/stop");
129         MQTT_subscribe("app/inicio");
130         MQTT_subscribe("app/stop");
131
132 };
```

En la función **MQTT_processTopic()** se termina proseando los tòpicos de entrada al sistema. Se evalúan dos fuentes una proveniente de la PC ***test/#*** y otra proveniente de la aplicación del celular ***app/#*** **.**

Ambas fuentes poseen el tópico ***inicio*** para inicaalizar el muestreo, y el tópico ***stop*** para pararalo.

Si la información proviene dela PC para ambos tópicos a demás de realizar sus funciones se llaman a la función ***MQTT_log()*** para registrar el evento.

```
89  /*****************************************************************************
90  MQTT_processTopic(): lee el mensaje MQTT recibido cuando se dispara el evento
91  *****************************************************************************/
92  void MQTT_processTopic(const char * topic, const char * msg){
93
94    /* Acciones a ejecutar para cada topic recibido */
95    if(strcmp("test/inicio", topic)==0 || strcmp("app/inicio", topic)==0) {
96         printf("MQTT: <test/inicio> Mensaje recibido: %s\n", msg);
97         // Registro en el log
98         if(strcmp("test/inicio", topic)==0) MQTT_log(MQTT_IN, "test/inicio", msg);
99         micro.inicio=atoi(msg);
100        if(micro.inicio==0 || micro.inicio>TIME_MAX){
101                printf("Se limitó el tiempo inicio: %d\n", TIME_MAX);
102                micro.inicio=TIME_MAX;
103        }
104        if(micro.run_stop==1){
105                printf("ERROR- espera que se terminar el muestreo anterior\n");
106                }
107           else{
108                micro.muestra=0;
109                micro.run_stop=1;        // Inicio el timer
110        }
111
112    }
113    if( strcmp("test/stop", topic)==0 || strcmp("app/stop", topic)==0 ) {
114         printf("MQTT: <test/stop> Mensaje recibido: %s\n", msg);
115         // Registro en el log
116         if(strcmp("test/inicio", topic)==0) MQTT_log(MQTT_IN, "test/stop", msg);
117         micro.run_stop=3;               // Indica que finalizó el muestreo
118    }
119 }
120
```

Otro de los archivos principales es el **crono.c ,** tiene como función principal leer el conversor analógico digital conectado al acelerómetro. La lectura se lo hace llamando a una tarea mediante in timer `CRONO_timerCallback ()` por intervalos regulares de 10ms(100Hz). Ese timer se puede parar usando la función `CRONO_timerStop()` y arrancar usando la función `CRONO_timerStart()` .

A continuación se muestra la función ***CRONO_timerCallback () :***

```
/*****************************************************************************
 CRONO_timerCallback(void* arg): callback para la interrpción de timer.
*****************************************************************************/
static void CRONO_timerCallback(void* arg)
{
  int64_t epoch_actual;
  char timestamp[64]="";
  char file_name[32];

  switch (micro.run_stop){
    case 2:
      if(micro.muestra==0){
        micro.epoch_init=CRONO_getTime(timestamp, sizeof(timestamp));
        micro.epoch_finish=micro.epoch_init+micro.inicio*1000;
        printf("micro.epoch_init: %lli\n", micro.epoch_init);
        printf("micro.epoch_finish: %lli\n", micro.epoch_finish);

        sprintf(file_name, "A%lld.csv", micro.epoch_init);    // Registro en el log
        MQTT_log("Nuevo evento-->", "Archivo:", file_name);
      }

      (dato+micro.muestra)->valor_adc=IO_readAdc();
      epoch_actual=CRONO_getTime(timestamp, sizeof(timestamp));
      (dato+micro.muestra)->epochs=epoch_actual-micro.epoch_init;

      // Se usa para el chequeo de umbrales
      if (fabs(data_max) < fabs((dato+micro.muestra)->valor_adc)) data_max = (dato+micro.muestra)->valor_adc;

      micro.muestra++;
      if(epoch_actual>=micro.epoch_finish){
        micro.run_stop=3;               // Indica que finalizó el muestreo
      }
      break;
    default:
      break;
  }

}
```

Cuando se encuentra en el estado **run_stop=2**, esta función lee el conversor AD. Si es la primeravez que pasa deja registrado el **micro.epoch_init** y calcula el **micro.epoch_finish**.

Con el **micro.epoch_init** luego se genera el nombre del archivo, y con el **micro.epoch_finish** se calcula hasta cuando se tiene que tomar muestras. Cuando termina de tomar muestras cambia el estado de **run_stop=3**, para indicarle al programa principal, que puede guardar los datos en el archivo.

El programa principal tiene una primara parte en donde se inicaliza el sistema(Wifi,MQTT, SD, ADC, etc), y luego posee un while(1). En ese while es en donde se maneja la máquina de estados con la variable **run_stop**, y por el otro lado cada 500mseg llama a una función ***unbral()*** para actualizar los estados de alerta (ATENCIÓN, PRECAUCIÓN, ALERTA y REPOSO).

Importante: agregé un estado más (REPOSO), con el cual tengo la opciónde apagar los LED's de la aplicación del celular.

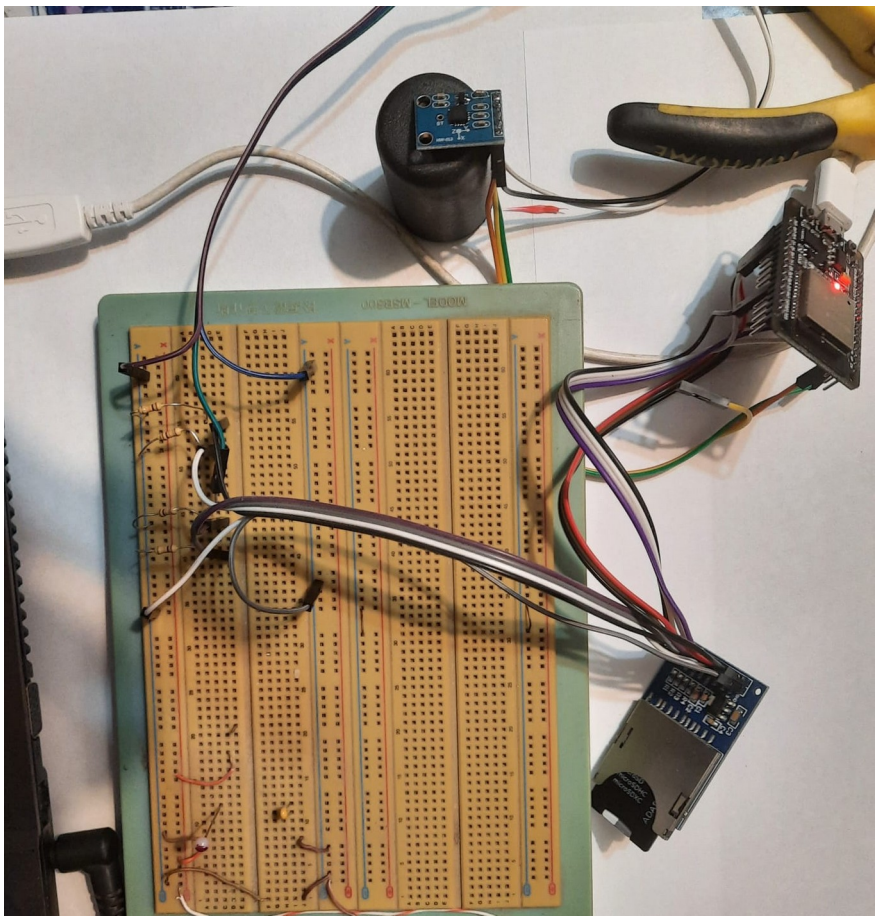
Importante 2: se creo otra función **name_long_to_short()** que convierte el nombre largo del archivo(del tipo **A1651826659407.csv**) a un nombre corto (del tipo **A6659407.csv**), ya que no me permitía crear el archivo.

A continuación se muestra el while del programa principal:

```
135   loops_umbral=0;
136   while(1){
137     CRONO_delayMs(Delay_100MS);
138
139     switch(micro.run_stop){
140       case 1:
141         CRONO_timerStart((uint64_t) micro.t_muestreo);           // Lanzo el contador por default es 100Hz
142         micro.run_stop=2;
143         break;
144       case 2:                                                    // En el estado 2 se está muestreando
145         break;
146       case 3:
147         CRONO_timerStop();                                       // Para el muestreo
148         micro.run_stop=4;
149
150         // Ceo el nombre del archivo
151         sprintf(file_name, "A%lld.csv", micro.epoch_init);
152         //*************************************************************************************************************
153         printf("EL nombre del archivo a crear=%s\n",file_name);         // es del tipo A1651826659407.csv
154         name_long_to_short(file_name_corto, file_name,12);              // no puedo crear archivos mayor a 12 caracteres
155         printf("EL nombre del archivo creado=%s\n",file_name_corto);    // es del tipo A6659407.csv
156         //*************************************************************************************************************
157
158         FILE * f2 = SD_open(file_name_corto, "w");                 // Archivo para escritura
159
160         //Ciclo de lectura-Encabezado
161         SD_printf(f2, "milisegundos,datos\n");
162         for(n=0;n<micro.muestra;n++){
163           SD_printf(f2, "%lli,%i\n", (dato+n)->epochs, (dato+n)->valor_adc);
164         }
165         printf("Se escribió el arcivo=%s\n",file_name_corto);
166
167         SD_close(f2);
168         micro.epoch_init=EPOCH_INICIAL;
169         break;
170     }
171
172     // Miestras mide se chequean los umbrales cada 500mseg, en otra condición espera
173     loops_umbral++;
174     if (loops_umbral>LOOPS_UMBRAL){
175       umbral();
176       loops_umbral=0;
177     }
178
179   }
```

Nota: se puede mejorar el uso de la variable de estado **run_stop.**

Foto del hardware utilizado



En ella se puede visulizar las resistencias de pull-up en el protoboard, el módulo SD, el ESP32S y el acelerómetro. El ESP32 se conecta a la PC con un cable USB (blanco), por el cual se programa, y puede enviarse información de consola. Esa información se utiliza para ver información del sistema y usarlo como un debugger elemental.

A continuación se puede ver un ejemplo de la u¡información de consola:

```
␛[0;32mI (14421) MQTT: MQTT_EVENT_DATA␛[0m
MQTT: <test/inicio> Mensaje recibido: 1
␛[0;32mI (14451) CRONO/TIMER: Started timers, time since boot: 13913631 us␛[0m
micro.epoch_init: 1651890031219
micro.epoch_finish: 1651890032219
␛[0;32mI (14461) SD: Opening file␛[0m
REPOSO ....
␛[0;32mI (14951) SD: Opening file␛[0m
EL nombre del archivo a crear=A1651890031219.csv
EL nombre del archivo creado=A0031219.csv
␛[0;32mI (15561) SD: Opening file␛[0m
Se escribió el arcivo=A0031219.csv
```

Comandos MQTT de la PC

Cargar el servidor MQTT:

```
$ sudo service mosquitto start

$ sudo service mosquitto status

$ sudo service mosquitto stop
```

Los comandos MQTT enviados desde la PC o desde la APP son:

```
$ mosquitto_pub -h 192.168.0.113 -t "test/inicio" -m "20"

$ mosquitto_pub -h 192.168.0.113 -t "test/stop" -m ""
```

Para suscribirse a los tòpicos MQTT recibidos desde el ESP32 o desde la APP son:

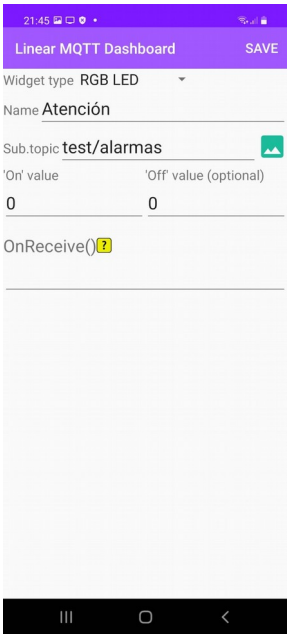
```
$ mosquitto_sub -h 192.168.0.113 -t "test/log" -v

$ mosquitto_sub -h 192.168.0.113 -t "test/rx_data" -v

$ mosquitto_sub -h 192.168.0.113 -t "app/stop" -v

$ mosquitto_sub -h 192.168.0.113 -t "app/inicio" -v
```

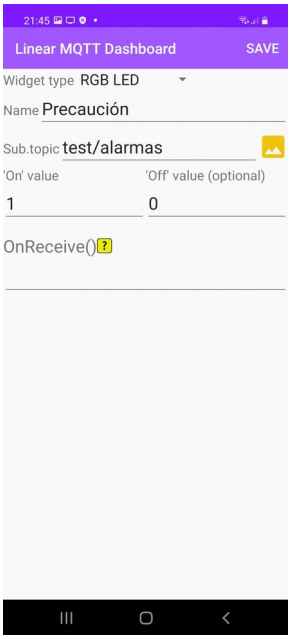
La aplicación del celular

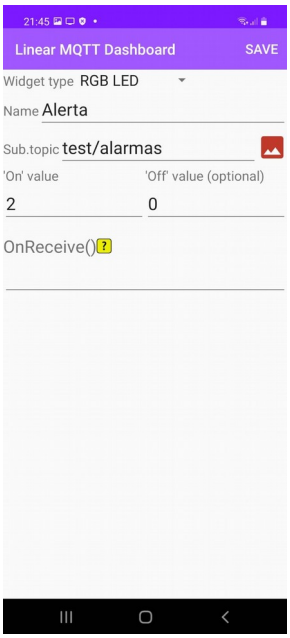
Se intaló el ***Linear MQTT***, y se configuró un panel con Widgets que contenga:

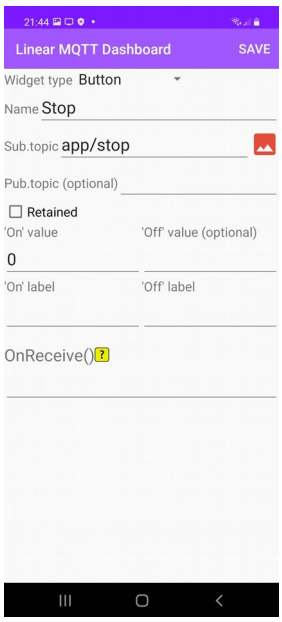
- Pulsador para el inicio de medición (el mensaje que publica debe ser los segundos de duración de la medición).
- Pulsador para finalizar la medición antes de tiempo.
- 3 LEDs RGB (si no se posee la App, emulardo de otra forma), cada uno para indicar tres niveles de alarma (como lo hizo en la práctica 3).
- Un pulsador de reset para poder apagar los LEDs en caso de haber recibido previamente alguna alarma.

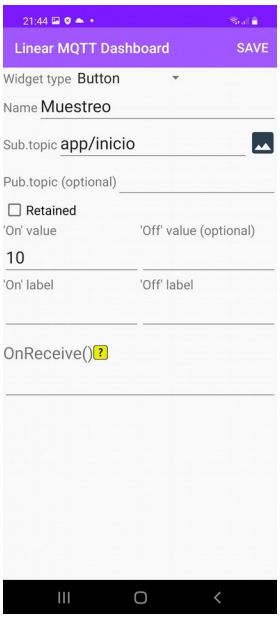
Se muestran las imágenes de lo configurado:

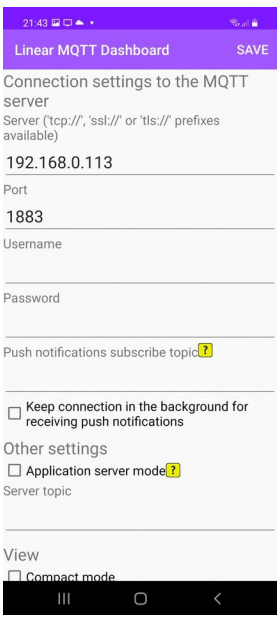


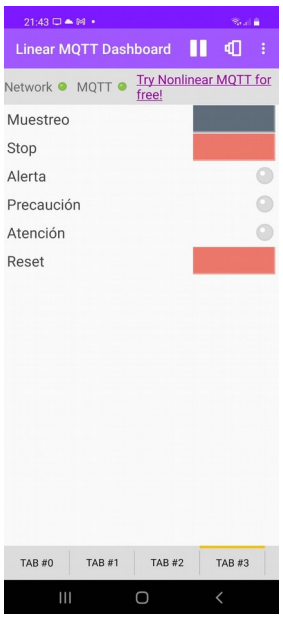












Scrip de la PC

Broker_PC_ver2.sh

```
##!/bin/bash


#topic1="control/logEsp"
#topic2="control/logApp"

broker="192.168.0.113"
port="1883"
topic1="test/logEsp"
topic2="app/inicio"
file="Log.txt"

#------------------------------------------------
if [ -f $file ]; then \
    rm -rf $file
fi

touch $file

mosquitto_sub -t $topic1 -t $topic2 -h $broker -p $port -v | while read topic value
do
#Add timestamp on the log
timestamp=$(date "+%d/%m/%Y %H:%M:%S")

if [ $topic = $topic1 ]
 then
    echo "$timestamp MQTT-NODO: $topic $value." >> $file
    echo "$value"                  # mostramos el resultado por consola
 else
    echo "$timestamp MQTT-APP:  $topic $value." >> $file
    echo "$value"                  # mostramos el resultado por consola
fi

done
```

Este scrip genera un archivo log.txt, como el siguiente:

```
06/05/2022 19:01:52 MQTT-APP:  app/inicio 2.
06/05/2022 19:01:53 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:01:53 MQTT-NODO: test/logEsp Archivo::
<A1651874512713.csv>.
06/05/2022 19:01:53 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:01:54 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:01:54 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:29 MQTT-APP:  app/inicio 20.
06/05/2022 19:02:29 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:29 MQTT-NODO: test/logEsp Archivo::
<A1651874549383.csv>.
06/05/2022 19:02:30 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:30 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:31 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:32 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:32 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:33 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:33 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:34 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:35 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:35 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:36 MQTT-NODO: test/logEsp test/alarmas: <4>.
```

```
06/05/2022 19:02:36 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:37 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:38 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:38 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:39 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:40 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:40 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:41 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:41 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:42 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:43 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:43 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:44 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:44 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:45 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:46 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:46 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:47 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:48 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:48 MQTT-NODO: test/logEsp test/alarmas: <4>.
06/05/2022 19:02:49 MQTT-NODO: test/logEsp test/alarmas: <4>.
```

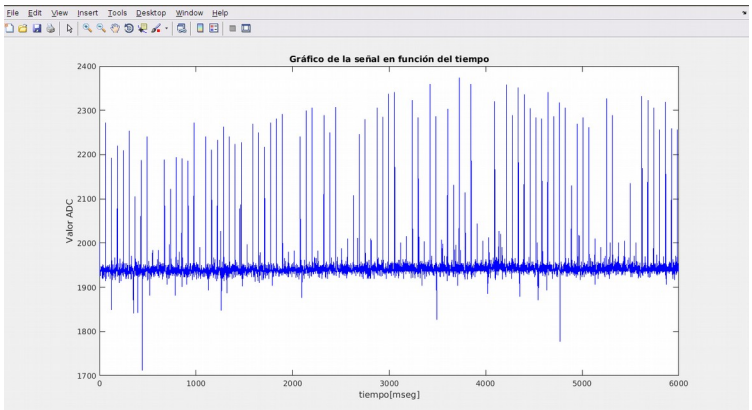
Gráficos de las mediciomes

4.1 Medición 1 con el sensor quieto

Código Matlab

```
medicion1.m
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5068595.CSV', ',', offset_row, offset_col);
6 -    plot(x, 'color', 'b')
7 -    xlim([xmin xmax])
8 -    xlabel('tiempo[mseg]')
9 -    ylabel('Valor ADC')
10 -   title('Gráfico de la señal en función del tiempo')
```

Gráfico temporal



En este grágico se obtuvieron los siguientes datos:

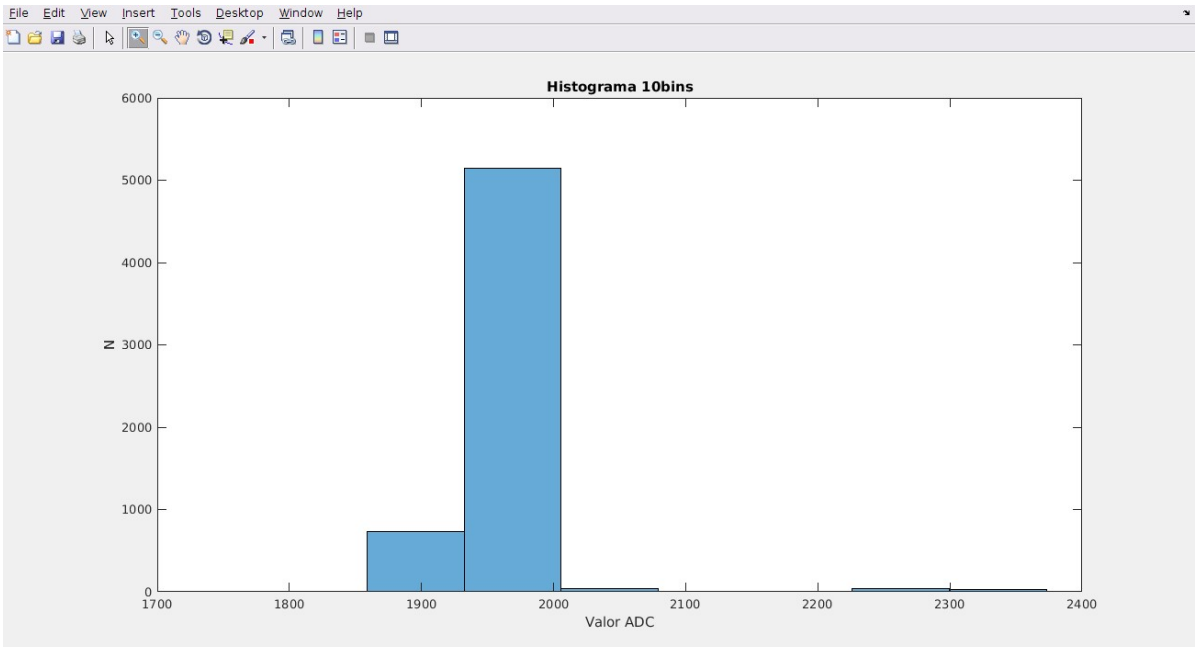
Máximo=2373
Mínimo=1712

El códigoMatlab y el histagrama de 10bins es el siguiente:

Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24_SED_Sistema

medicion1_histograma_10bins.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5068595.CSV', ',', offset_row, offset_col);
6 -    xbins=linspace(1712,2373,10)
7 -    histogram(x,xbins)
8 -    xlabel('Valor ADC')
9 -    ylabel('N')
10 -   title('Histograma 10bins')
```

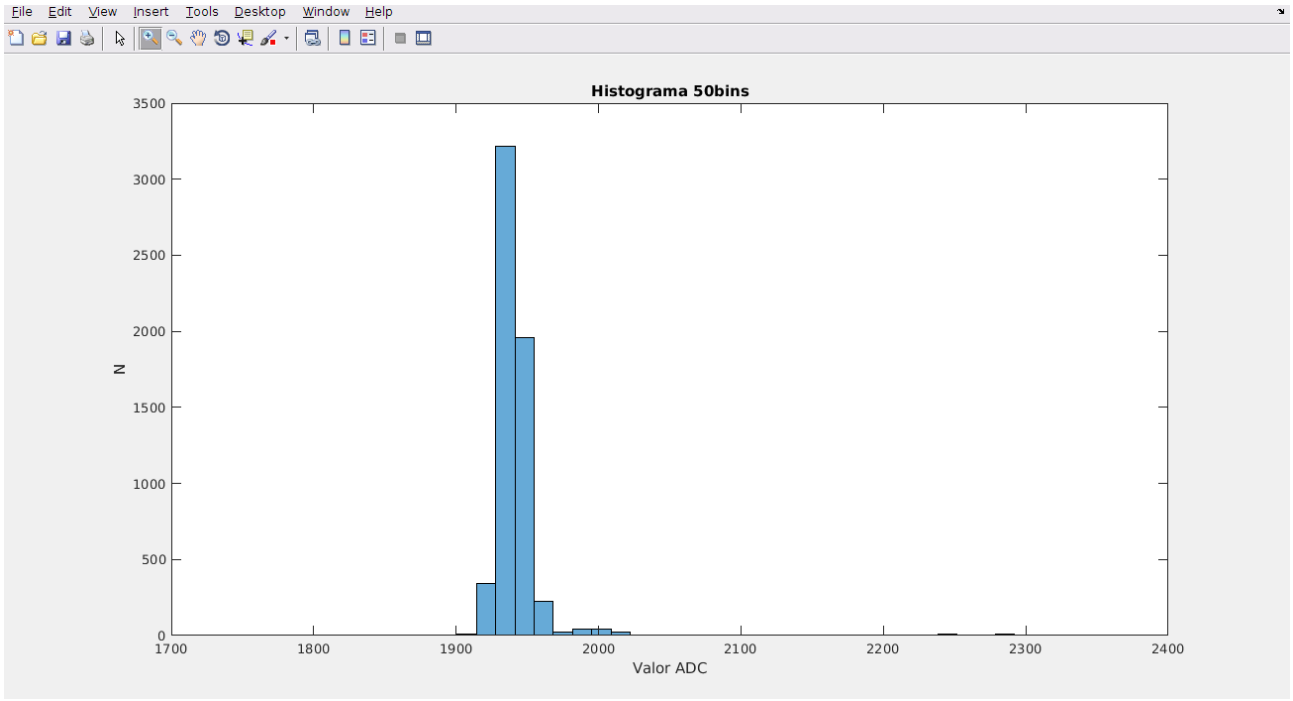


El códigoMatlab y el histagrama de 50bins es el siguiente:

Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24_SED_Siste

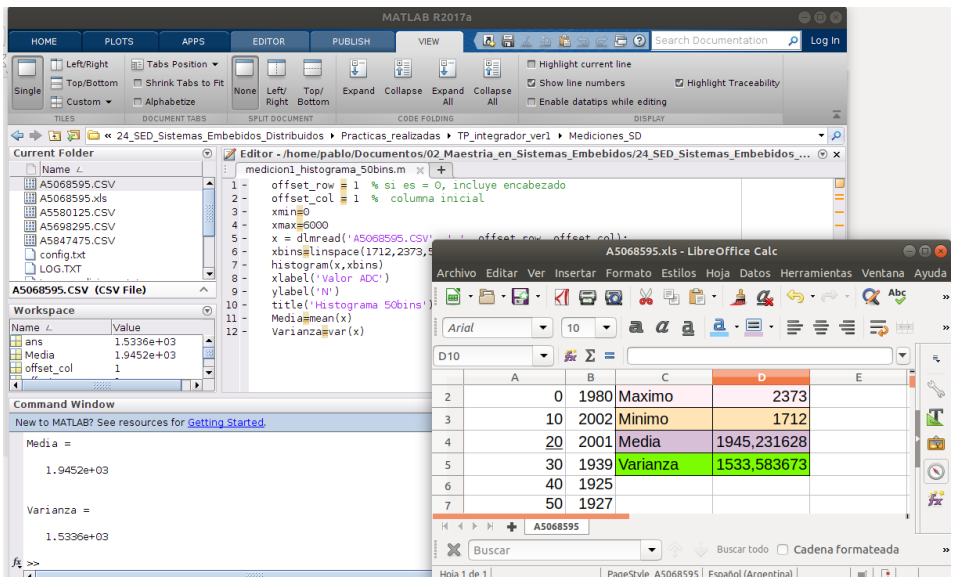
medicion1_histograma_50bins.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5068595.CSV', ',', offset_row, offset_col);
6 -    xbins=linspace(1712,2373,50)
7 -    histogram(x,xbins)
8 -    xlabel('Valor ADC')
9 -    ylabel('N')
10 -   title('Histograma 50bins')
```



En el gráfico de 50bins se observa mayor concentración delos datos.

Se calculó la media y la varianza:

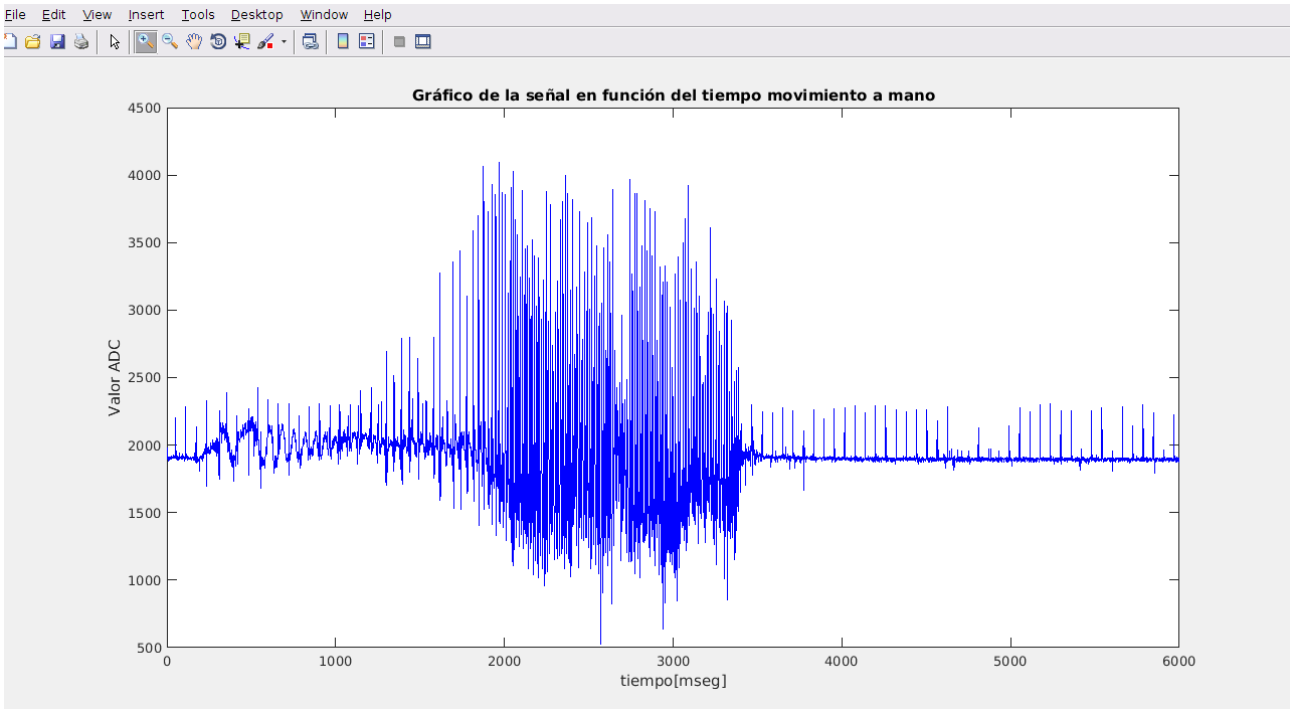


4.2 Medición 2 con el sensor moviéndolo a mano

Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24_SED_Sistemas_

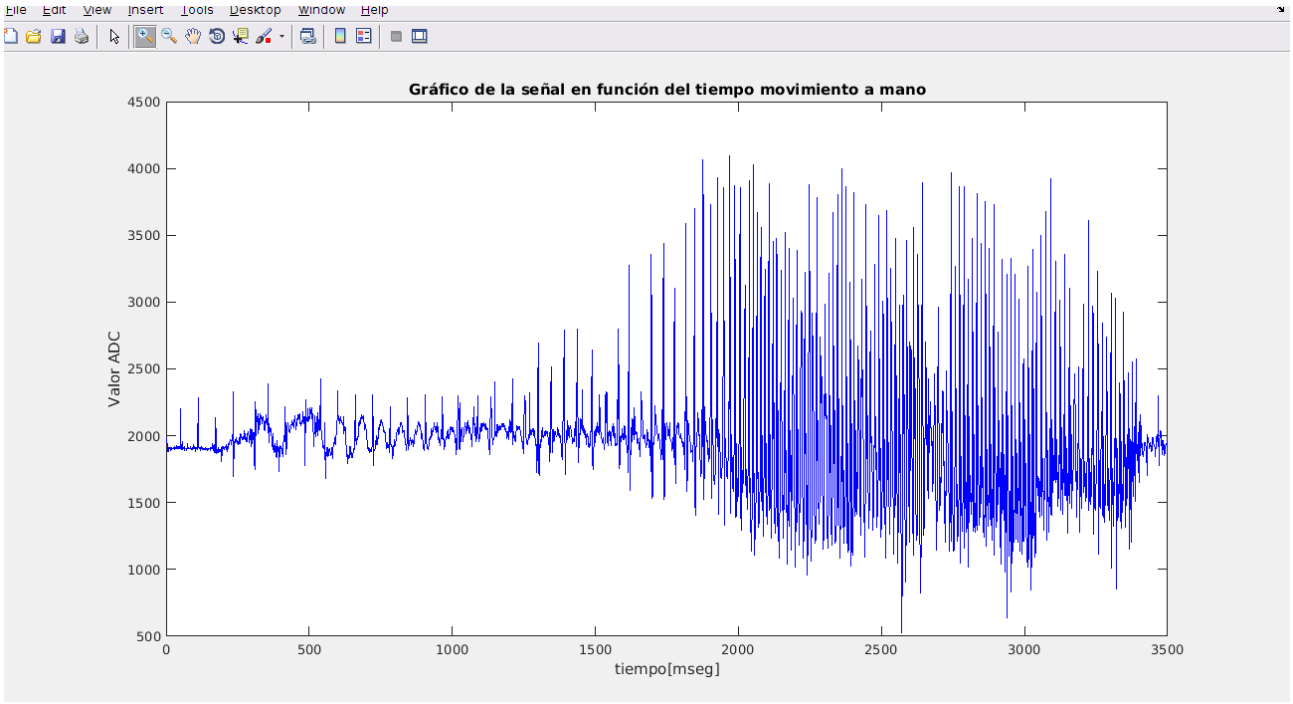
medicion2.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5847475.CSV', ',', offset_row, offset_col);
6 -    plot(x, 'color', 'b')
7 -    xlim([xmin xmax])
8 -    xlabel('tiempo[mseg]')
9 -    ylabel('Valor ADC')
10 -   title('Gráfico de la señal en función del tiempo movimiento a mano')
11
```



Me comenzó a doler la muñeca, por eso termine de mover antes de las 60segundos.

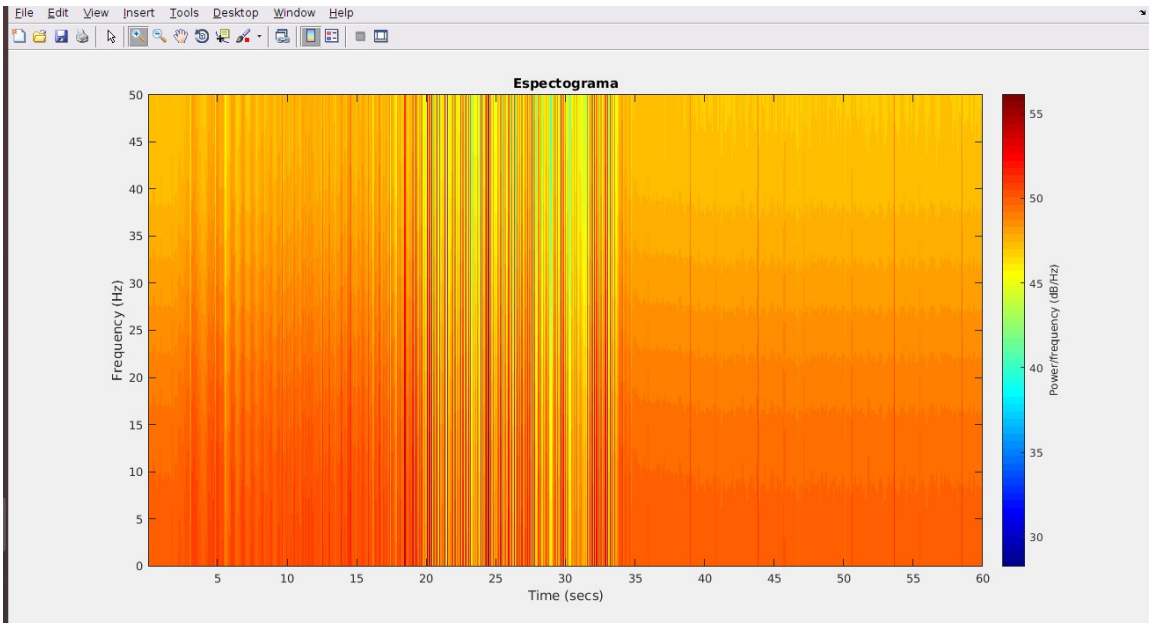
Para la siguiente parte tomo hasta la muestra 3500, que es donde dejñe de mover el acelerómetro.



Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24_S

medicion2_espectrograma.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=3500
5 -    x = dlmread('A5847475.CSV', ',', offset_row, offset_col);
6      %spectrogram(x,3,2.8,1024,100,'yaxis')
7 -    spectrogram(x , 3 , 2 , 1024 , 100, 'yaxis')
8 -    title('Espectograma')
9 -    colormap('jet')
10
```

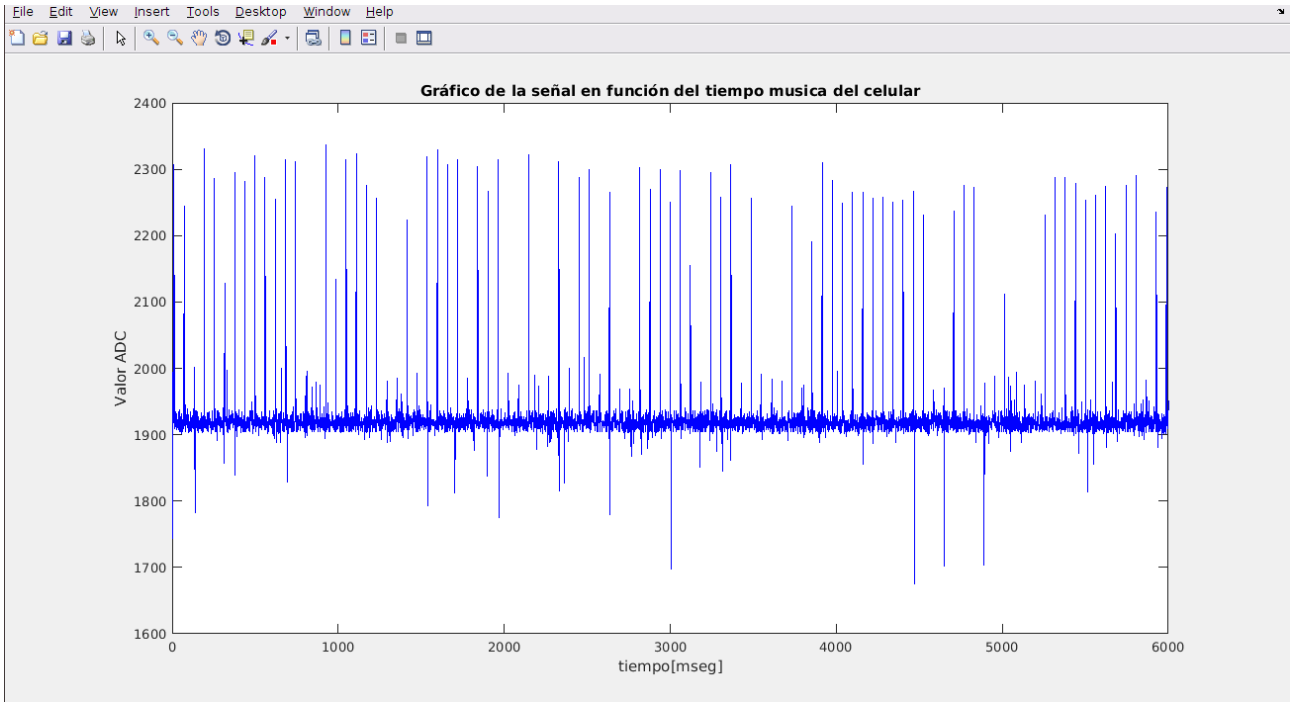


4.3 Medición con el sensor apoyado en el celular y en el con música

Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24_SED_Sistemas_Embebid

medicion2_espectrograma.m | medicion2_musica.m

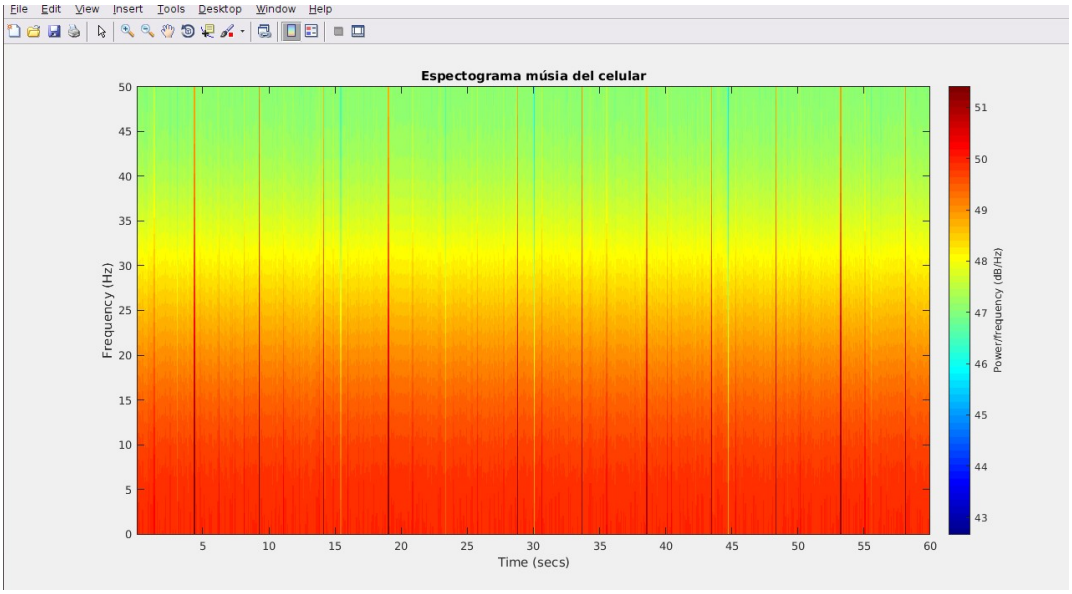
```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5698295.CSV', ',', offset_row, offset_col);
6 -    plot(x, 'color', 'b')
7 -    xlim([xmin xmax])
8 -    xlabel('tiempo[mseg]')
9 -    ylabel('Valor ADC')
10 -   title('Gráfico de la señal en función del tiempo musica del celular')
11
```



Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/24

medicion2_espectrograma_musica.m | medicion2_musica.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=3500
5 -    x = dlmread('A5698295.CSV', ',', offset_row, offset_col);
6      %spectrogram(x,3,2.8,1024,100,'yaxis')
7 -    spectrogram(x , 3 , 2 , 1024 , 100, 'yaxis')
8 -    title('Espectograma músia del celular')
9 -    colormap('jet')
10
```

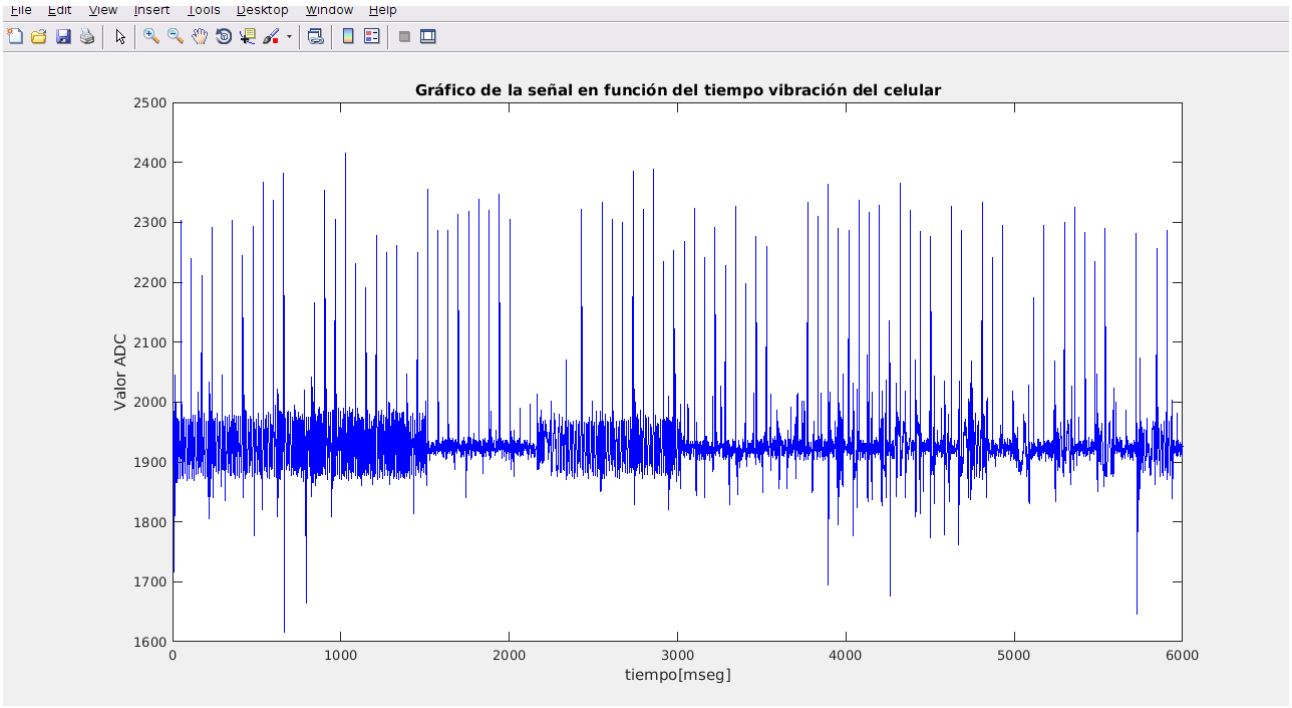


4.3 Medición con el sensor apoyado en el celular y en el en vibración

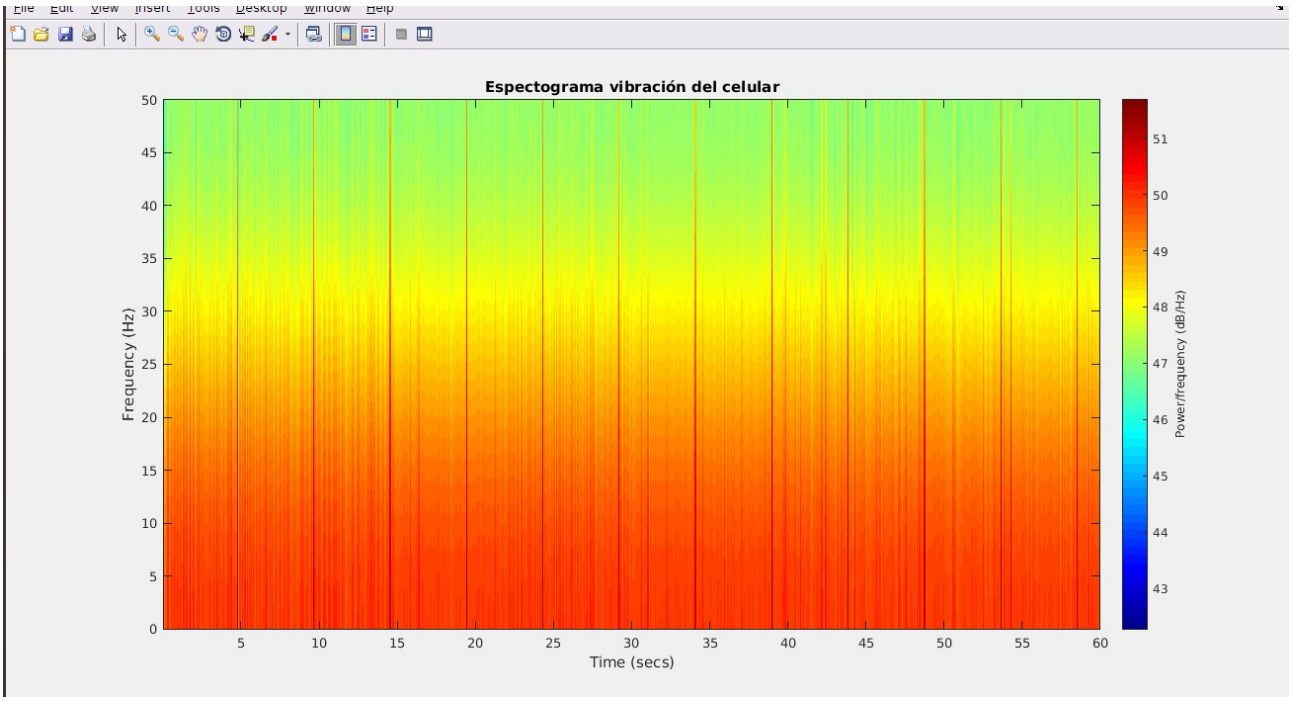
Editor - /home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/

medicion2_espectrograma_vibracion.m | medicion2_vibracion.m

```
1 -    offset_row = 1  % si es = 0, incluye encabezado
2 -    offset_col = 1  %  columna inicial
3 -    xmin=0
4 -    xmax=6000
5 -    x = dlmread('A5580125.CSV', ',', offset_row, offset_col);
6      %spectrogram(x,3,2.8,1024,100,'yaxis')
7 -    spectrogram(x , 3 , 2 , 1024 , 100, 'yaxis')
8 -    title('Espectograma vibración del celular')
9 -    colormap('jet')
10
```




```
medicion2_espectrograma_musica.m | medicion2_vibracion.m
-    offset_row = 1  % si es = 0, incluye encabezado
-    offset_col = 1  %  columna inicial
-    xmin=0
-    xmax=6000
-    x = dlmread('A5580125.CSV', ',', offset_row, offset_col);
-    plot(x, 'color', 'b')
-    xlim([xmin xmax])
-    xlabel('tiempo[mseg]')
-    ylabel('Valor ADC')
-    title('Gráfico de la señal en función del tiempo vibración del celular')
```



Conclusiones

El sensor posee un ruido bastante importante, esto puede ser devido que el lugar eligido como reposo no era tan así.

La aplicación del celular (Linear MQTT), e vez e cuando se bloqueaba u dejaba de transmitir.