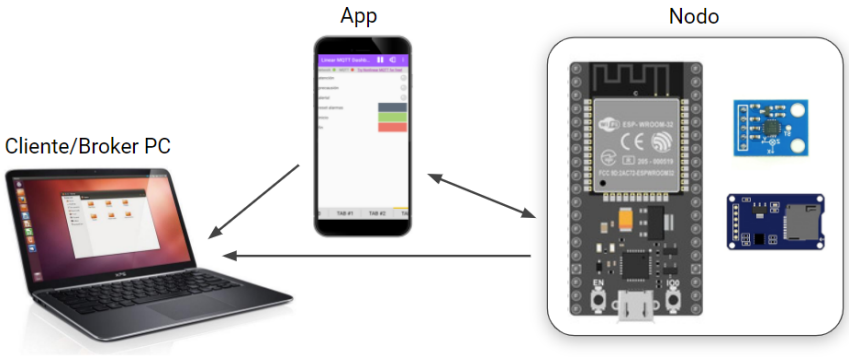
Trabajo práctico integrador

Sistemas Embebidos Distribuidos

1. Introducción

En esta práctica integradora se busca implementar un sistema de monitoreo de vibraciones a partir de las muestras sensadas por un acelerómetro, registrando eventos anómalos (alarmas) y almacenando la información recolectada para su posterior análisis mediante alguna de las herramientas matemáticas recomendadas. En todo este proceso se buscan integrar los contenidos relacionados con: protocolo MQTT, cliente/servidor mosquitto, manejo de sensores, manipulación de archivos en SD, uso de una App cliente MQTT, visualización y procesamiento básico de los señales. En relación al firmware, se tomará como punto de partida el proyecto de referencia el utilizado en la práctica 3. En la siguiente figura se puede ver un esquema simplificado del sistema que deberá implementarse.



Materiales necesarios:

- NodeMcu-32S (ESP32)
- Módulo adaptador SD/SPI
- Acelerómetro
- App cliente MQTT
- PC para cliente/broker MQTT

2. Requerimientos generales

A continuación se definen los requerimientos que deben cumplirse en cada parte del sistema a ser implementado.

Nodo:

- El ESP32 debe estar permanentemente a la espera de un comando de inicio para una nueva medición.
- El comando de inicio que dispara una nueva medición se debe ser enviado desde la App. El mensaje de este comando debe contener la cantidad de segundos que durará la medición.
- La duración especificada en el comando de inicio no debe exceder los 60 s. En caso que se reciba una duración mayor, o un valor no válido (letras, valores negativos, etc), el nodo deberá establecer 60 s como valor predeterminado.
- La medición debe finalizar una vez se haya cumplido el tiempo preestablecido en el inicio o ser interrumpida en cualquier instante a través de un comando enviado desde la App.
- La frecuencia de muestreo para el sensor debe ser de 100 Hz.
- Los capturas del sensor deben almacenarse en la tarjeta SD en forma de tabla en un archivo de formato CSV. Debe incluir una columna para el tiempo en milisegundos, tomando como 0 ms la primer muestra, y otra columna para los datos. También debe agregarse el encabezado correspondiente a cada columna. El nombre del archivo debe seguir el siguiente formato: `A<epoch>.csv` (ejemplo: `A1649413442947.csv`). *Nota: no es necesario escribir este archivo en la SD en simultáneo con el proceso de muestreo. Esto puede hacerse una vez concluida la medición.*
- Durante la medición deben chequearse los umbrales para generar los tres niveles de alarma y publicarlas mediante MQTT cada vez que algún evento de estos ocurra (básicamente lo que se hizo en la práctica 3).
- Todos los eventos de entrada o salida MQTT (inicio, fin y alarmas) deben quedar registrados en un archivo de texto `log.txt`. Cada uno de estos registros debe indicarse con el siguiente formato:

  `<timestamp> MQTT-IN: <topic sub> <mensaje>.`

  `<timestamp> MQTT-OUT: <topic pub> <mensaje>.`

  En cada nueva medición, antes de registrar los nuevos eventos en este archivo, debe agregarse una línea indicando el nombre del nuevo archivo CSV donde se guardarán las muestras tomadas del sensor.

Cliente/Broker PC:

- Debe utilizarse un broker MQTT local con el servidor mosquitto corriendo en la PC.
- Antes de iniciar el muestreo, se debe dejar corriendo un script de bash en una terminal a la espera de recibir datos desde el sensor a través de MQTT. El script debe registrar en un archivo de texto todos los eventos de MQTT publicados por el *nodo* o la *App*.

También debe indicarse qué cliente MQTT envío el mensaje. El formato deberá ser el siguiente:

`<timestamp> MQTT-NODO: <topic> <mensaje>`.

`<timestamp> MQTT-APP: <topic> <mensaje>`.

Nota: En la PC se considera como timestamp el instante de recepción del mensaje MQTT, por lo cual, éste tiempo no necesariamente deberá coincidir con los timestamp del nodo.

App:

Con la App *LinearMQTT* (o similar), configure 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.

3. Mediciones

Una vez implementado el sistema y verificado el cumplimiento de los requerimientos, se deberán realizar distintas mediciones para poder analizarlas posteriormente.

3.1. Medición 1 - Reposo

Coloque el sensor en un lugar donde se garantice que esté completamente quieto y no posea ningún tipo de vibración (el suelo por ejemplo). Una vez hecho esto, realizar 60 segundos de medición sin mover en sensor.

3.2. Medición 2 - Perturbaciones periódicas

Iniciar la medición con la duración máxima (60 s). Una vez iniciada, tome el sensor y realice movimientos oscilantes agitándolo a distintas frecuencias. Sugerencia: realice movimientos de aproximadamente 1 Hz durante al menos 10 s y luego vaya modificando la frecuencia del movimiento hasta completar el tiempo de medición (procure alcanzar la mayor velocidad posible).

3.3. Medición 3 - Opcional

Elija a su criterio cualquier fuente de vibraciones que en base a su disponibilidad considere pueda resultar interesante analizar (sonidos con alto contenido de baja frecuencia, algún equipo o máquina que genere vibraciones, como movimiento objetos pesados, etc.).

4. Procesamiento de los datos

Una vez finalizadas todas las mediciones, debe extraerse la tarjeta SD para introducirla en la PC. Allí levante los datos desde la herramienta de procesamiento que disponga (Matlab, Octave/Octave-online o Python). Luego con los archivos generados en cada medición procese los siguientes datos:

4.1. Medición 1

En esta parte vamos a caracterizar el ruido de fondo propio del sensor:

- Gráfico `plot` de la señal muestreada vs tiempo (el tiempo en ms).
- Observe la señal en el tiempo y determine los umbrales mínimo y máximo de la señal `x` (`xmin` y `xmax`) que considere para no incluir en el histograma muestras esporádicas (por ejemplo impulsos) que no garanticen la estacionariedad del ruido. Luego grafique el histograma de la señal para 10 bins entre los umbrales definidos (ver función `histogram()` en el Apéndice).
- Gráfico del histograma de la señal para 50 bins. ¿Qué diferencias observa con el anterior?
- Estimación de la media y la varianza de la señal.

4.2. Medición 2

Para esta medición vamos a ver el contenido espectral de la señal captada por el sensor y su evolución temporal.

- Gráfico `plot` de la señal muestreada vs tiempo.
- Espectrograma de la señal. Primero quítele a la señal su valor medio. Luego, para definir los parámetros de la función, tenga en cuenta la frecuencia de muestreo utilizada (100 Hz), una ventana temporal de 3 s segundos, con 2.8 s de overlap (en la función del espectrograma, tanto el largo de la ventana como el overlap deben ingresarse en muestras, no en segundos) y 1024 puntos de FFT. Luego ejecute `colormap('jet')` para la configuración de color del espectrograma.

4.3. Medición 3

Igual que en la medición 2, se quiere ver el contenido espectral a lo largo del tiempo, pero para una señal de naturaleza aleatoria y no estacionaria.

- Gráfico `plot` de la señal muestreada vs tiempo.
- Espectrograma de la señal. En principio pruebe con los mismos parámetros de la medición 2, pero modifíquelos en función de la necesidad.

5. Entregable

A continuación se indica una lista de los puntos que deberán presentarse en la entrega del trabajo. Se detallan tanto el contenido del informe y los archivos adjuntos requeridos (que deberán incluirse en un ZIP).

5.1. Informe

1. Breve descripción de la implementación (diagrama de flujo, maquina de estado, diagramas de tiempo, o cualquier otra forma que considere apropiada para describir de forma sintética el diseño del firmware implementado).
2. Debe incluirse también una foto del nodo utilizado con todas las conexiones.
3. Capturas de consola con los mensajes recibidos para los eventos producidos (aprovechando alguna de las mediciones realizadas o generando una medición aparte para que se pueda capturar al menos una alarma de cada tipo).
4. Capturas de la App para diferentes eventos de alarma (al menos una de cada tipo).
5. Gráficos y resultados (sección 4) obtenidos en la etapa de procesamiento para las mediciones solicitadas.
6. Conclusiones indicando inconvenientes encontrados (y cómo se abordaron), una resumen sobre el cumplimiento total o parcial de los requerimientos y un breve análisis de los resultados observados en la etapa de procesamiento que considere relevantes.

5.2. Archivos adjuntos

1. Informe en formato PDF.
2. Carpeta del proyecto con el código fuente utilizando para el ESP32.
3. Script de bash empleado en el cliente PC para el registro de mensajes MQTT.
4. Archivo de texto con mensajes recibidos en el cliente PC.
5. Archivos CSV con los datos registrados del muestreo del Sensor.
6. Archivo de texto `log.txt` con los registros de eventos MQTT en el ESP32.
7. Código de los archivos `.m` (o `.py` en caso de usar python) para el procesamiento posterior de la señal.

6. Apéndice

Funciones recomendadas en matlab/Octave

- `mean()` (matlab/octave): estimador de la media
- `var()`(matlab/octave): estimador de la varianza
- `histogram(x,xbins)`(matlab): grafica el histograma de la secuencia `x` con los bins definidos en el vector xbins. Por ejemplo, dado un umbral mínimo y máximo para `x`, se puede definir `xbins=linspace(xmin,xmax,m)` donde `m` es la cantidad de bins.
- `hist(x,xbins)`(octave): igual `histogram()` pero con otro nombre.

- `spectrogram(x, nwin, overlap, nfft, fs, 'yaxis')` (matlab):

  Grafica el espectrograma de `x` con una ventana de `nwin` puntos, con un solapamiento entre ventanas de `overlap` puntos, una cantidad de puntos de FFT de `nfft` y una frecuencia de muestreo `fs`. Incluya `'yaxis'` para que el eje de frecuencia sea la ordenada.

- `specgram(x, nfft, fs, nwin, overlap)` (octave):

  Los parámetros tienen el mismo significado que en matlab, sólo que se ingresan en otro orden.

Para más detalles, si utiliza matlab u octave, se puede consultar la ayuda con el comando `help` o desde la ayuda en la web.