Adquisición y almacenamiento de señal mediante un sistema de tiempo real

12 de enero de 2022

Autores: Sonia Bartolomé, Gorka Irureta y José María Irizar

Profesor: Jose Antonio Pascual Saiz

Escuela de Informática de Donosti

Curso 2021 - 2022

Índice

Índice de figuras

1. Introducción

En este proyecto se adquieren los valores de varias magnitudes físicas mediante sensores sobre un sistema operativo de tiempo real (FreeRTOS) y envío por puerto serie para poder ser tratados en un pc.

2. Objetivos

El objetivo de este proyecto ha sido el de aprender a utilizar el sistema operativo de tiempo real FreeRTOS, mediante el diseño de una aplicación implementada en la placa de desarrollo NUCLEO-F411RE y programada con el entorno de programación STMCubeIDE, que utilice distintos componentes para poder comunicarse con un host también programado por el equipo.

3. Funcionalidad de cada componente

3.1. NUCLEO Board

Se utiliza una "development board"de STMicroelectronics NUCLEOF411 para hacer la adquisición de datos mediante ADCs y envío por puerto serie al pc. Para facilitar esta tarea se utilizará un sistema operativo de tiempo real.

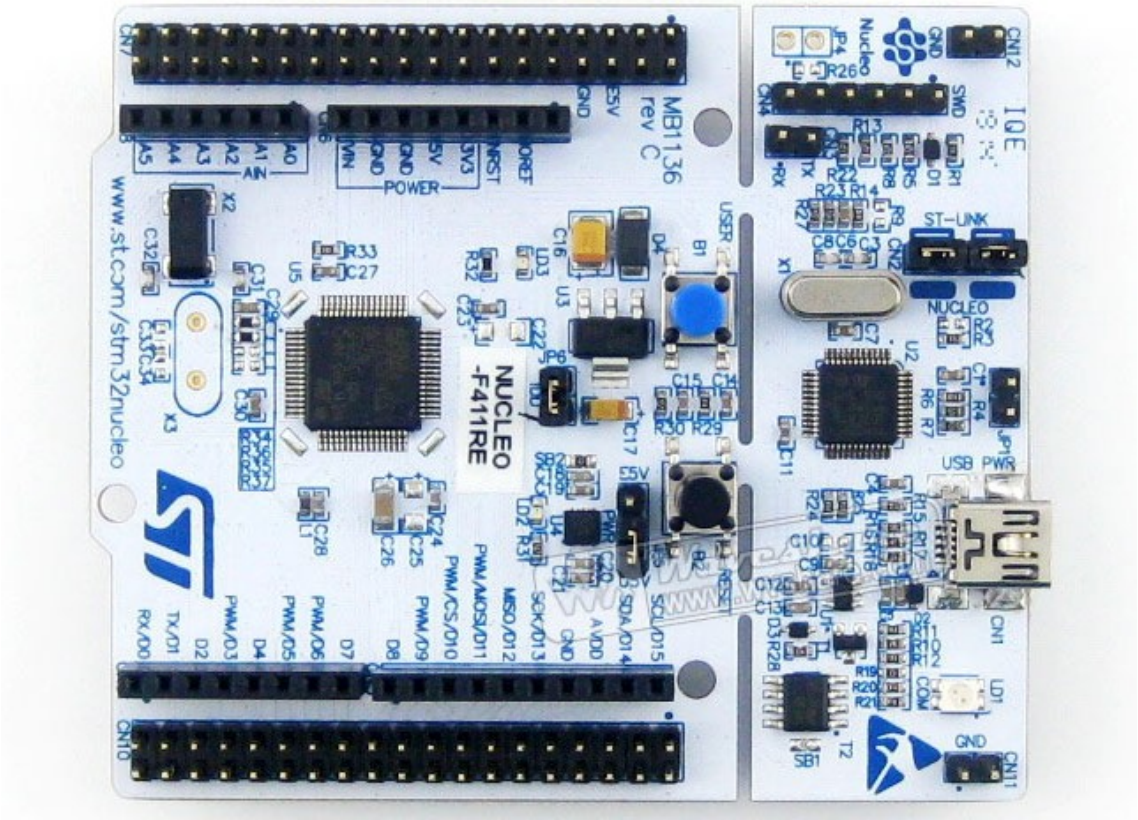

Figura 1: Placa de desarrollo NUCLEOF411RE

3.2. PC

El pc se utiliza para la recepción de datos por puerto serie y para la configuración de las magnitudes físicas. Los datos se muestran por pantalla y se almacenan en un log para que poteriormente puedan ser tratadas para DSP.

3.3. Protoboard

La placa protoboard se usará para poder montar el hardware necesario para que las magnitudes físicas sean acondicionadas para poder ser leídas por los ADCs.

4. Hardware

Se hará la adquisición de temperatura, luz y ángulos. Para la temperatura se utilizará un sensor TMP36. La luz se medirá mediante un divisor de tensión con un LDR genérico. Los ángulos se calcularán con el acelerómetro ADXL335.

El sensor TMP36 tiene tres pines. Alimentación (a 3.3V), tierra y la señal. El pin de señal se puede conectar directamente al pin del ADC del microcontrolador.

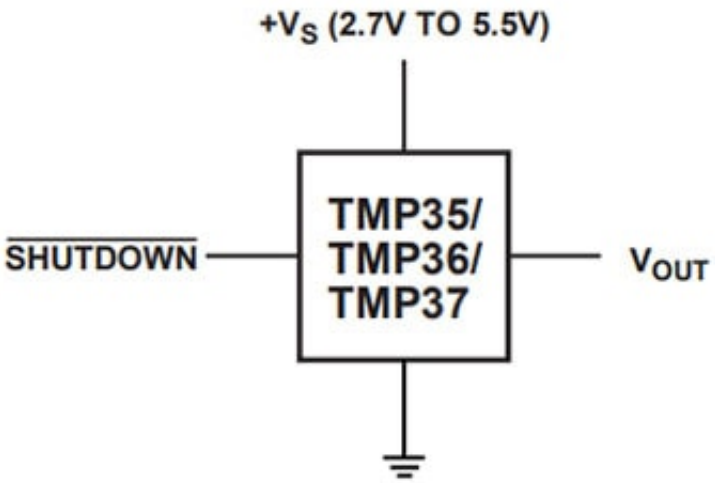

Figura 2: Diagrama del sensor TMP36

El LDR se conecta en la parte superior del divisor de tensión. El divisor se alimenta a 3.3V y el punto común de los dos resistores se conecta al pin del ADC.

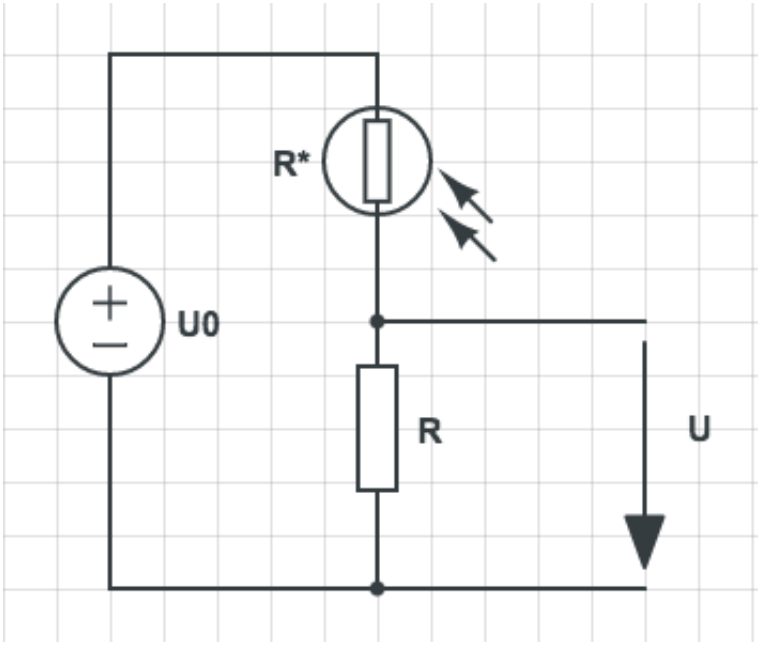

Figura 3: Diagrama del divisor de tensión

El sensor ADXL335 es un acelerómetro de 3 ejes con pines de alimentación a 3.3V y otros 3 pines que se conectan directamente a los pines del ADC.

5. Máquina de estados

5.1. NUCLEO Board

Se ha diseñado un algoritmo en 3 fases principalmente. Durante la primera fase el programa se queda a la espera de recibir los parámetros de configuración de la temperatura y de los grados del acelerómetro. Una vez recibidos estos datos se pasara a recoger los valores del acelerómetro, del LDR y temperatura. Si se recibe algún dato por la UART el sistema se reinicia y vuelve a ponerse a la espera. Podríamos resumir el funcionamiento de la siguiente manera:

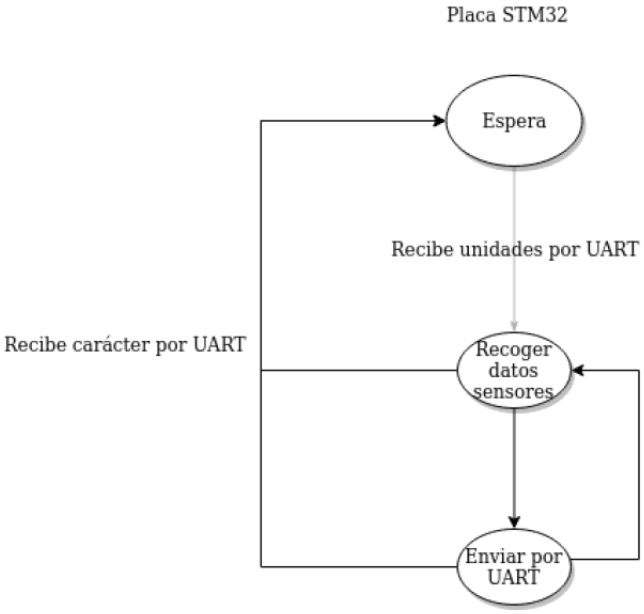

Figura 4: Máquina de estados de la NUCLEO Board

5.2. PC

El programa para la comunicación desarrollado sobre Linux tiene un funcionamiento acorde con la máquina de estados de la placa. Primero se le pedirá al usuario que introduzca las unidades en las que la placa debe mandar las lecturas. Estas se mandarán vía UART y a partir de ese momento se empezarán a recibir los valores por la UART, que se escribirán en un archivo log. Cuando el usuario termine el programa mediante CTRL+C se recogerá la excepción y se mandará un carácter para que la placa se reinicie. Después de esto se cierran los archivos y puertos.

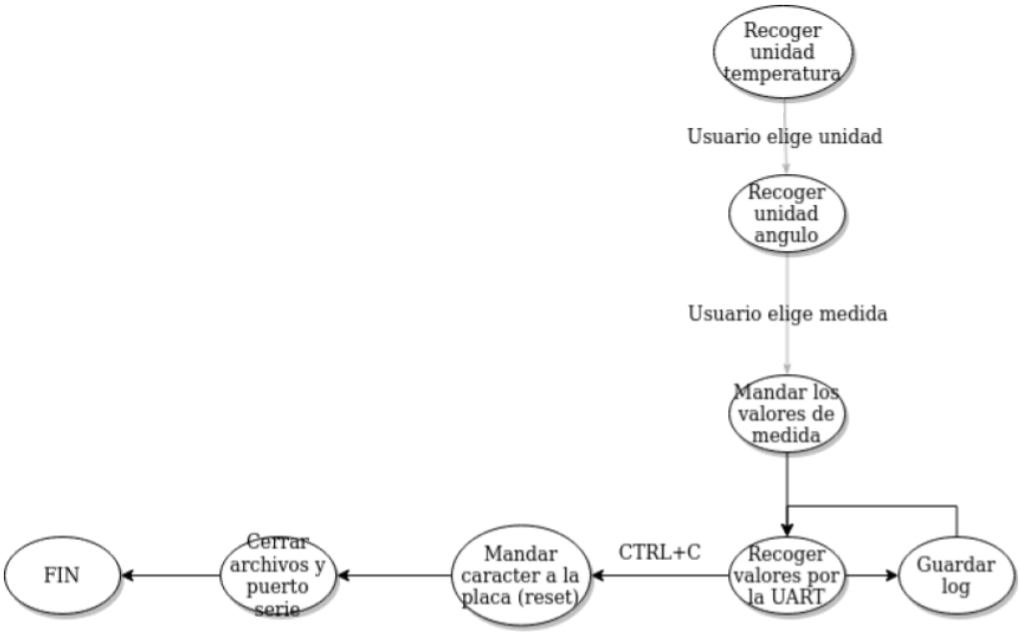

Figura 5: Máquina de estados del Host

6. Definición de las tareas

Hay una tarea para el envío por puerto serie y cinco tareas para adquirir los datos. También hay una tarea para hacer el reset del sistema. La configuración de las medidas de las magnitudes se hace antes de llamar al Scheduler.

6.1. sendSerial

Esta tarea recoge los valores de temperatura, iluminancia e inclinación de las distintas colas y, poniéndolos en un formato que los visualizará en columnas en el host, hace la transmisión serie de los datos. Si la transmisión se ha realizado correctamente, el LED RGB parpadeará en color verde y si ha habido algún error, lo hará en color rojo.

6.2. getTemp

Esta tarea selecciona el canal 0 del ADC (sensor de temperatura) y mediante encuesta espera a que el ADC haga la conversión de la temperatura. Cuando se obtiene el dato se hacen los cambios necesarios de unidades y se pone el resultado obtenido en la cola tempQueue. Si la conversión no ha sido realizada con éxito, el LED RGB parpadeará en color azul.

6.3. getLight

Esta tarea selecciona el canal 1 del ADC, correspondiente al LDR y mediante encuesta espera a que el ADC haga la conversión de la iluminancia. Cuando se obtiene el dato se pone en la cola lightQueue. Si la conversión no ha sido realizada con éxito, el LED RGB parpadeará en color azul.

6.4. getRoll

Esta tarea selecciona el canal 4 del ADC para medir el eje "Roll" del acelerómetro. Mediante encuesta espera a que el ADC haga la conversión de la inclinación y una vez obtenido el dato se hacen los cambios necesarios de unidades y se añade a la cola rollQueue. Si la conversión no ha sido realizada con éxito, el LED RGB parpadeará en color azul.

6.5. getPitch

Esta tarea selecciona el canal 6 del ADC para medir el eje "Pitch" del acelerómetro. Mediante encuesta se espera a que el ADC haga la conversión de la inclinación y una vez obtenido el dato se hacen los cambios necesarios de unidades y se añade a la cola pitchQueue. Si la conversión no ha sido realizada con éxito, el LED RGB parpadeará en color azul.

6.6. getYaw

Esta tarea selecciona el canal 7 del ADC para medir el eje "Yaw" del acelerómetro. Mediante encuesta espera a que el ADC haga la conversión de la inclinación y una vez obtenido el dato se hacen los cambios necesarios de unidades y se añade a la cola yawQueue. Si la conversión no ha sido realizada con éxito, el LED RGB parpadeará en color azul.

6.7. resetTask

Esta tarea se encarga de resetear el programa una vez que se haya recibido un carácter del host. Cabe destacar que no hay problema de interferencia con el primer envío que hace el host con las unidades de los datos porque esa operación se hace antes de que se inicialice el kernel y posteriormente no se vuelve a enviar nada más, salvo el carácter de cierre del host.

7. Mutex

Hay un mutex para el control del acceso de las tareas que adquieren datos para que no accedan al ADC de forma simultanea y se corrompan los datos.

De esta manera se asegura que el ADC no deje ningún proceso de conversión sin acabar, dado que sólo se libera el mutex una vez ha acabado de hacer la conversión.

8. Colas de mensajes

La comunicación entre las distintas tareas se realiza mediante el uso de colas. En este caso, y para no mezclar información, se ha creado una cola para cada dato que se quiera enviar, de manera que el puerto serie pueda escoger los datos que quiera mandar:

- tempQueue (Cola para la temperatura)
- lightQueue (Cola para la iluminancia)
- rollQueue (Cola para el tilt del eje roll)
- pitchQueue (Cola para el tilt del eje pitch)
- yawQueue (Cola para el tilt del eje yaw)

El programa interfaz del host se ocupa de pedir al usuario las unidades en las que quiere los datos de los diferentes sensores y de enviárselos a la placa, para posteriormente recogerlos en un archivo de log además de imprimirlos por pantalla. Cuando el usuario mantiene la combinación de letras CTRL + C para cerrar el programa host se envía un carácter a la placa para informarle de que va a cerrar la conexión.

9. Conclusiones

Las conclusiones que se pueden extraer de este proyecto son las siguientes: No se ha visto necesario el usuario de threads del programa del host para el recibo y envío de datos dado que la comunicación bidireccional no se ha realizado en el mismo momento, si no que se envía o se recibe. Con respecto a la sincronización sólo se ha visto necesario el uso de un Mutex en vez de semáforos dado que solo se ha necesitado restringir el acceso exclusivo al recurso compartido (ADC). No se han utilizado flags para la comunicación entre tareas, únicamente se han utilizado las colas para el envío de datos de una tarea a otra. Por otro lado, nos hubiese gustado hacer alguna funcionalidad más avanzada, como, por ejemplo el control del servo, pero decidimos desarrollar otras funcionalidades en su lugar. Hemos añadido más funcionalidades, como el uso de Logs o errores mediante la señalización en los LEDs. Dicho esto, queremos señalar que el proyecto nos ha parecido completo y de una curva de aprendizaje asumible ya que se basaba en los proyectos realizados en clase.