# Informe practica de laboratorio 7

**Universidad Sergio Arboleda**
**Sistemas Embebidos**

**Estudiantes:**
- Sharay Valentina Galvis Prieto.
- Juan Diego Moreno Cardenas.

## Introducción:

El presente informe describe el desarrollo e implementación de un sistema embebido utilizando el microcontrolador STM32F411. El objetivo principal del laboratorio fue diseñar un sistema capaz de ubicar y mantenerse a una distancia determinada de una pared; por medio del desarrollo del sistema de medición implementado en el laboratorio anterior, se utilizaron los LEDs IR para conocer la distancia a la que se encuentra la pared, de esta forma se desarrolló una función encargada de hacer mover los motores hasta que los sensores detectan la distancia esperada. Se implementó una función que permite recibir los datos e información enviada desde una aplicación de ventana diseñada en el software QT Creator que permite ajustar la distancia a la cual se espera que se mantenga el sistema de la pared.

Entre los diferentes objetivos que se plantearon se encuentra, el establecer una comunicación USB mediante el puerto COM, realizar la conexión del segundo motor del sistema, ubicar la protoboard (con el circuito implementado) encima de la base en madera proporcionada por el profesor, de forma en que se encuentre estable, adicionar un pulsador a la configuración del circuito que se encargue de enviar una señal digital al microcontrolador y permitir ejecutar el procedimiento de mantener la distancia requerida, poder ejecutar el sistema de manera independiente, estando conectado a una fuente de alimentación externa; y por último, transmitir tanto el valor de la velocidad (mm/s) como de la distancia a la que se espera mantener el sistema de una pared (mm), desde la interfaz gráfica desarrollada en QT Creator, recibir los valores y almacenarlos en la memoria del microcontrolador. Además, todo el proyecto se reestructuró con el uso de librerías en donde se encuentran todas las funciones utilizadas.

Para cumplir con estos objetivos, se trabajó en la integración de todos los sistemas desarrollados en laboratorios anteriores, se utilizó el mismo circuito implementado para la lectura de distancias en los pines analogicos, se utilizo la misma configuracion del reloj, de la comunicación USB y de los diferentes periféricos y módulos utilizados (Timers, ADC, DMA). Además, todo el desarrollo de este laboratorio está anexo al laboratorio anterior, de esta manera cuenta con todas las funcionalidades que se desarrollaron tanto en el laboratorio #6 como en el laboratorio #7. Este laboratorio permitio reforzar conceptos sobre uso de librerías, manejo en el almacenamiento de datos, tipos de variables y funcionamiento de la transmisión de datos.

**Materiales utilizados:**

- Microcontrolador STM32F411.
- ST-Link V2.
- Cable USB.
- Base ROBOT.
- 2 motorreductores con encoders de cuadratura.
- 2 llantas.
- puente H TB6612FNG.
- Protoboard.
- Capacitores.
- Resistencias.
- MOSFET 2n7000.
- Led IR emisor.
- Led IR receptor.
- pulsador.
- Cables de conexión.

**Softwares utilizados:**

- Stmcube IDE.
- QT Creator.

**Procedimiento:**

El procedimiento seguido para la implementación del sistema fue:

1. Inicialmente, se realizó el montaje de los dos motorreductores en la base del robot brindada por el profesor, se utilizaron los soportes diseñados en 3D de cada motor para poder fijarlos a la base y se ajustaron con el uso de tornillos 2m y su respectiva tuerca; además, se realizó la conexión de los pines pertenecientes al segundo motor, conectandolos tanto a su respectiva alimentación, al puente h, y a los pines gpio asignados como modo encoder.

2. luego, se llevó a cabo el montaje de la protoboard (con el circuito utilizado en el laboratorio #6) sobre la superficie de la base del robot brindada por el profesor, se colocó el tornillo frontal de la base que es el encargado de próximamente asegurar la posición de la pcb; además, se dispuso una base de madera de la misma altura que el tornillo y se pegó sobre la base del robot, de esta manera se colocó la protoboard con el circuito encima de estas dos bases y se aseguró su estabilidad fijando la protoboard a estas mismas. Este montaje es temporal y posibilita la opción de posteriormente retirar dichos elementos para próximos desarrollos del robot.

3. Para el desarrollo de este laboratorio utilizamos como base la configuración del proyecto desarrollado anteriormente (laboratorio #6), con lo cual comenzamos a configurar los parámetros del proyecto desde STM32IDE. Como se mencionó anteriormente, el proyecto debe contar con las funcionalidades del laboratorio anterior, con lo cual, utilizamos el proyecto creado para ese caso y a partir de este modificamos ciertas configuraciones. Se mantienen tanto las entradas analógicas (ADC)(Analog-to-Digital Converter) como la conexión del DMA (Direct Memory Access), se estableció la conectividad USB_OTG_FS en modo Device_Only, se configuró el USB_DEVICE en la clase Communication Device Class. Al utilizar este medio de transmisión, era necesario establecer el reloj interno del microcontrolador a una velocidad de 96MHz. Además, se mantuvo activó el periférico ADC1, para poder obtener los datos de los 4 leds IR, específicamente se utilizó IN4, IN5, IN6, IN7 para recibir los datos. Se mantuvo la configuración del DMA2 en modo circular, con baja prioridad y con dirección desde el periférico hacia la memoria del microcontrolador; además, para la recepción de los datos, permaneció activa la opción “scan conversion” para que el ADC recorra todos los canales que utilizamos y realice la conversión una tras otra, además se mantuvo una resolución de 12 bits y se dejó el orden de lectura de los canales desde el cuarto hasta el séptimo.

4. Una vez configurados los periféricos del microcontrolador, se configuró el siguiente pin en la STM32 (además de los ya configurados en los laboratorios anteriores):

- **PIN B10**: se configuró un pin GPIO en modo INPUT, el cual será el encargado de recibir la señal emitida por acción del pulsador. Este pulsador se conectó con una configuración Pull-Up, con un voltaje de 3.3v y una resistencia de 10kΩ.

5. En este punto, se realizó el desarrollo de una función dentro del programa de la stm32, que se encarga de mantener el robot a una distancia determinada de una pared. Se elaboró una función que recibe como parámetro una distancia en mm, la cual es transmitida vía usb desde la interfaz gráfica en QT y recibida desde el microcontrolador (por funciones desarrolladas en laboratorios anteriores); este valor se almacena y se integra en esta función. Además, se estructuró toda la función dentro de un bucle *while* para que se ejecute indefinidamente hasta obtener la distancia esperada, se controla la salida del ciclo con una “bandera”. Dentro de la función, se toma los valores de lectura de los 4 LEDs IR (convertido a mm), promediándolos para obtener un aproximado de la distancia real a la que se encuentra la pared, luego se realiza una resta entre el valor de distancia esperado y el detectado por los sensores, a partir de esto conocemos el error o la diferencia que se presenta entre la distancia del robot y la que se desea, como se observa en la *Ec.1*. Al obtener este valor, se utiliza un condicional *if* para comparar si este dato está dentro de la tolerancia permitida que se dispuso (± 2mm), en tal caso detendrá los motores y modificará la “bandera” utilizada para romper el bucle y salir de la función (siempre y cuando también se oprima el pulsador del circuito cuando el sistema está detenido para verificar la instrucción de terminar la ejecución del seguimiento). En caso de que el error no se encuentre dentro de la tolerancia definida, se compara aquel valor con otro condicional *if* para comprobar si el error es mayor o menor que 0, de esta manera se ejecuta una instrucción para mover los motores hacia adelante o hacia atrás dependiendo del caso.

Dentro de esta misma función se realizó el llamado a las demás funciones implementadas en el programa, ya que al ser un bucle sólo se ejecutaría esta parte del programa y no se realizarian las otras operaciones del sistema (calcular rpm, recibir datos, transmitir datos, etc…), de esta manera no se ve afectado el correcto funcionamiento del programa aunque se quede dentro de esta función.

$$error\ distancia\ = distancia\ esperada\ -\ (promedio\ distancia\ sensada\ por\ IR)$$

*Ec1. Ecuación cálculo error distancia esperada.*

6. Dentro de la función main(), se realiza el llamado a las funciones anteriormente mencionadas, sin embargo, para acceder a la función que realiza el procedimiento de mantener el robot a cierta distancia, se utilizó un condicional *if* para permitir su ejecución solamente cuando se oprima el pulsador implementado al inicio del laboratorio y cuando se haya recibido un valor de "distancia esperada" desde la comunicación USB. Por medio de la función HAL_GPIO_ReadPin() se obtiene el valor recibido por el pin conectado al pulsador y permite verificar el valor de entrada al microcontrolador.

7. Para que el código fuera más ordenado y fácil de manejar, se pasaron todas las funciones del archivo " main.c" a librerías separadas con archivos .c y .h. Esto nos permitió dividir el programa en partes según las funciones que contienen. Primero, se identificaron las funciones principales del proyecto, como las que manejan la comunicación USB, el cálculo de las RPM, el control de los motores, la lectura de los sensores IR y el ajuste de la distancia. Cada grupo de funciones se asignó a su propia librería, por ejemplo, una para comunicación, otra para velocidad, otra para distancia, etc. Luego, se crearon dos tipos de archivos para cada librería (.h)(.c).

   En el .h se declararon las funciones y las variables globales que otros archivos requieren, se utilizó el modificador "extern" para especificar que esas variables están definidas en otro archivo y poder utilizarlas en varias de estas librerías sin necesidad de redefinir las variables. También se incluyeron las estructuras creadas, como el "PAQUETE" para la librería de comunicación.

   El .c contiene el código que hace ejecutar las funciones declaradas en el .h, con todos los cálculos y la lógica. En estos archivos se definieron las variables globales sin "extern", inicializando sus valores. También se añadieron los #include necesarios, como el del .h correspondiente y archivos del STM32, para que todo se conecte correctamente.

   Para componer cada librería, se utilizaron las funciones que se encontraban en el archivo "main.c", se colocaron en los .c de cada librería, y se ajustó el archivo "main.c" para que solo inicialice el microcontrolador y llame a esas funciones desde el while principal del programa. Se utiliza #include para llamar los archivos .h de cada librería creada, así conoce qué funciones y variables se pueden usar. El modificador "extern" evita duplicados o redefiniciones de variables y posibilita que todos los archivos usen las mismas.

8. Posteriormente, se actualizó el proyecto creado en el laboratorio anterior en la aplicación QT Creator, en donde se generó un botón adicional a los ya creados, con el cual se toma el valor diligenciado en el cuadro de texto (correspondiente a la distancia en mm a mantener por el robot), y al oprimir el botón se realiza la transmisión de este dato con su respectivo identificador, hacia el microcontrolador.

9. Finalmente, se realizaron pruebas de funcionamiento del sistema en donde se mandaron diferentes distancias desde la interfaz en QT hacia el microcontrolador, se oprimió el pulsador que activa el procedimiento de seguimiento y se realizaban mediciones respecto a la distancia que mantenía el robot a la que se esperaba inicialmente.

**Dificultades presentadas:**

Durante el desarrollo del laboratorio, enfrentamos varios retos que complicaron el avance del proyecto, pero logramos resolverlos con análisis y apoyo del profesor, de la siguiente manera:

- **Mediciones inestables de los LEDs IR**: Cuando se realizaban las mediciones de las distancias por medio de los LEDs IR y se hacían las respectivas pruebas, se presentaron problemas con las lecturas, ya que en la distancia que mantenía el robot de la pared no correspondía con la enviada desde la interfaz. Cuando la tabla estaba lejos, tres de los cuatro sensores marcaban distancias correctas, pero uno mostraba un valor muy alejado, con lo cual al obtenerse el promedio de todos los LEDs IR en la función de seguimiento, este afectaba la correcta interpretación de la distancia por parte del microcontrolador. Pensamos que podía ser ruido en el circuito, que los LEDs no estaban bien puestos, o un mal contacto en la protoboard, así que tuvimos que ajustar el montaje y repetir las pruebas varias veces hasta que los datos daban correctamente.
- **Alineación de los LEDs:** Otra dificultad fue lograr que los LEDs emisores y receptores estuvieran bien alineados. Cómo usamos cinta aislante para evitar señales intrusas, cualquier pequeño movimiento en el montaje hacía que las lecturas cambiaran. No fue tan sencillo buscar la posición exacta entre emisor y receptor ya que era clave, y tuvimos que fijarlos mejor en la protoboard para que las mediciones fueran consistentes.
- **Alimentación externa:** Al no tener certeza de cómo realizar la alimentación del microcontrolador por medio de una fuente externa, sin incurrir en un cortocircuito o una sobrecarga sobre el mismo, esperamos a consultar con el profesor acerca de la correcta conexión que debería implementarse.
- **Implementación librerías:** Tener poca experiencia en el uso y elaboración de librerías dificulto mucho pasar las funciones desde el archivo “main.c” hacia cada una de las librerías elaboradas, por lo cual nos tomó demasiado tiempo comprender cómo pasar las funciones y conocer con que debía contar cada uno de los archivos (.c) y (.h).

**Diagramas:**

- Diagrama de flujo (*Archivo anexo ¨Link diagrama de flujo¨*).

- Diagrama esquemático (*Archivo anexo ¨Esquematico laboratorio 7¨*).

**Conclusiones:**

1. La integración de ambos programas correspondientes al laboratorio #6 y al programa actual se lograron de manera exitosa, sin presentar fallas en su funcionamiento. Esta unión permitió una comunicación eficiente entre las herramientas, asegurando la correcta transmisión y procesamiento de los datos. Además, mantener un completo funcionamiento del sistema, incluso en funciones como la del seguimiento (con bucle indefinido) permitió un funcionamiento fluido y sin cortes.
2. Pasar las funciones a librerías .c y .h nos enseñó que dividir el código en partes claras hace todo más ordenado y fácil de manejar. Aunque al principio fue complicado entender cómo armarlas, el resultado fue un programa más organizado donde cada librería tiene su función, y eso ayuda a encontrar y solucionar problemas más rápido.
3. Las dificultades con los LEDs IR nos mostraron que alinear bien los emisores y receptores es clave para que las mediciones sean confiables. Cualquier pequeño movimiento afectaba las lecturas, así que fijarlos correctamente en la protoboard fue esencial para que el robot mantuviera la distancia esperada.
4. Aprender a conectar una fuente externa correctamente nos permitió hacer que el robot funcione independiente del USB, lo cual es importante para próximos desarrollos del robot. Además, es algo importante porque haber realizado una mala conexión de la tarjeta podría haber significado el daño de la misma.
5. Integrar la interfaz de QT Creator con el STM32 por USB nos permitió enviar y recibir datos como la distancia esperada sin complicaciones. Esto, junto con el pulsador, nos permitió controlar cuándo empezar o parar el seguimiento de la distancia. Esto nos demostró que una señal simple, puede hacer el sistema más fácil de usar y evitar que se ejecute cuando no debe.