

**Universidad Nacional de La Matanza**

**Componentes e Instrumentos de Control**

***Segundo Parcial***

**Alumno:** Kelly Tomás – 40.546.348

**Comisión:** 3900

**Curso:** 01

**Días:** Miércoles 19:00-23:00

**Profesor:** Rodrigo Gómez

**Año:** 2022

Índice:

[1. Introducción 2](#_Toc117196021)

[I. Descripción 2](#_Toc117196022)

[II. Objetivos 2](#_Toc117196023)

[2. Funcionamiento 2](#_Toc117196024)

[I. Diagramas de estados: 3](#_Toc117196025)

[3. Diagrama de bloques 3](#_Toc117196026)

[4. Software 4](#_Toc117196027)

[II. Periféricos utilizados: 4](#_Toc117196028)

[III. Código 4](#_Toc117196029)

[IV. Interfaz 7](#_Toc117196030)

[5. Hardware 9](#_Toc117196031)

[V. Componentes 9](#_Toc117196032)

[VI. Circuito en Protoboard 11](#_Toc117196033)

[VII. Montaje 12](#_Toc117196034)

[6. Desarrollo 12](#_Toc117196035)

[VIII. Medidores de Distancia Ultrasónicos: 12](#_Toc117196036)

[IX. Comunicación Bluetooth: 12](#_Toc117196037)

[7. Conclusiones 13](#_Toc117196038)

[8. Referencias 13](#_Toc117196039)

[9. Repositorio 13](#_Toc117196040)

# Introducción

## Descripción

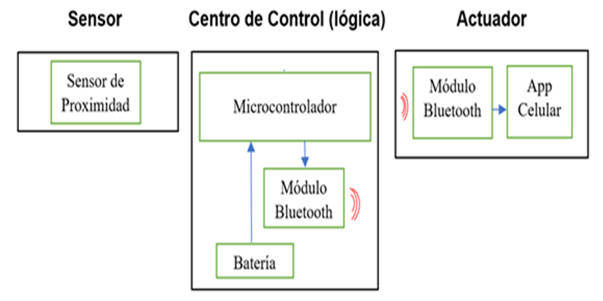
Lo que se busca en este proyecto es lograr medir alguna magnitud física. En este caso se eligió medir distancia. En este proyecto se pusieron a prueba todos los conocimientos adquiridos hasta ahora en electrónica y programación.

## Objetivos

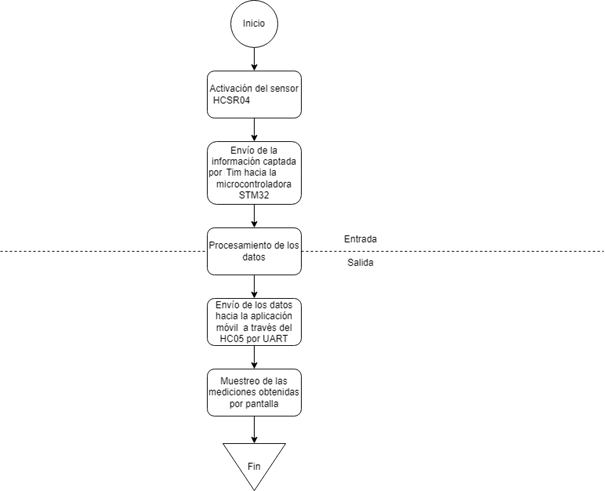
* + Medir una magnitud física y mostrarla a través de una terminal.

# Funcionamiento

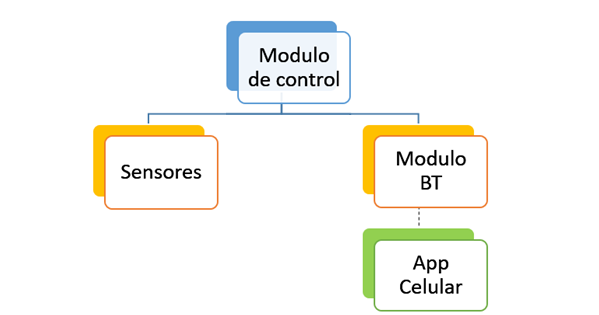
Por medio del Microcontrolador STM32F411 manejaremos el sensor de proximidad. Dicha información será procesada y enviada mediante un módulo Bluetooth al Smartphone con la aplicación correspondiente descargada.



### Diagramas de estados:



# Diagrama de bloques



# Software

### Periféricos utilizados:

Las comunicaciones a efectuar entre los distintos módulos:

* La comunicación entre el sensor HC-SR04 y el microcontrolador será manejada por el timer con los modos de operación correspondientes.
* La comunicación entre el módulo bluetooth (HC-05) y el celular del empleado será a través de UART.

### Código

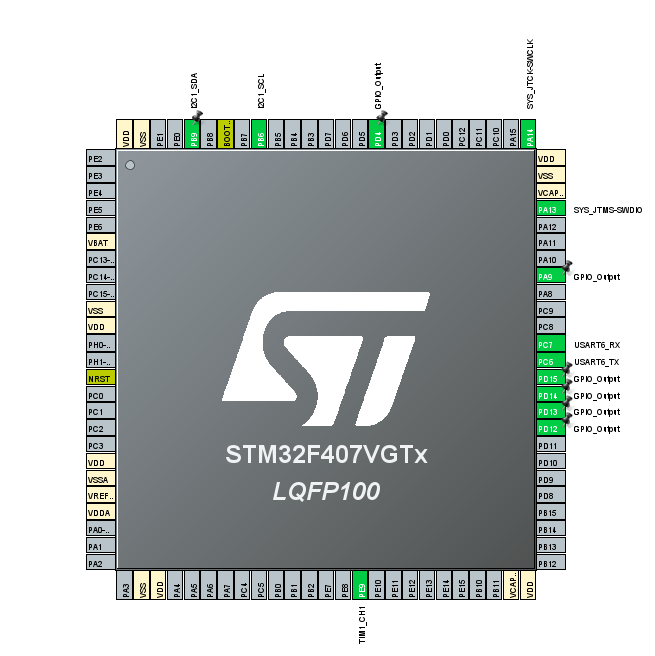
La interfaz que utilizamos para realizar el código y posteriormente debugearlo en la placa microcontroladora fue el STM32CubeIDE(v 1.6.1).

**Enumeración de rutinas**

En esta sección simplemente se van a enumerar todas las rutinas incluidas en el programa y la función que cumplen en el mismo en forma resumida.

* Rutina “main”: < Es la rutina principal y la que posee dentro todas las inicializaciones como el bucle principal en donde se encuentran todas las subrutinas >.
* Subrutina “HAL\_Init”: < Inicialización y configuración de las funciones de la HAL (Hardware Abstraction Layer) >.
* Subrutina “SystemClock\_Config”: < Configuración RCC >.
* Subrutina “MX\_GPIO\_Init”: < Inicialización GPIO >.
* Subrutina “MX\_TIM1\_Init”: < Inicialización del TIMER1 >.
* Subrutina “MX\_TIM2\_Init”: < Inicialización del TIMER2 >.
* Subrutina “MX\_USART6\_UART\_Init”: < Inicialización de la UART6 >.
* Subrutina “HAL\_TIM\_IC\_CaptureCallback”: < Interrupción de captura TIM >.
* Subrutina “HAL\_TIM\_PeriodElapsedCallback”: <Interrupción de periodo TIM>.
* Subrutina “HAL\_UART\_TxCpltCallback”: <Interrupción de transmisión UART>.

La etapa previa a la generación de código es lo que se conoce como la “selección de periféricos”. Esto nos da como resultado un archivo .ico, el cual, luego de realizar la asignación de todos los pines y periféricos a usar, nos permite, gracias el STM32CubeIDE, generar el código en el que se inicializan todos estos periféricos que seleccionamos. El resultado fue el siguiente:



**IMAGEN TOMADA DE EJEMPLO**

Decidí explicar y mostrar los fragmentos de código por categoría, con el fin de facilitar su entendimiento por parte del lector. Quedándonos de la siguiente manera:

* **While(1):** es el bucle en el que se ejecutan todas las funciones que engloban el funcionamiento final del proyecto. Dentro de este está cada función correspondiente a cada componente utilizado.

**while** (1){

HCSR04\_Read();

UART\_Task();

}

* **HCSR04-Read():** es la función correspondiente al sensor ultrasónico.

**void** **HCSR04\_Read** (**void**)//Inicio el HCSR04

{

**if**(counter\_HC >= 999)

{

counter\_HC = 0;

HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, *GPIO\_PIN\_SET*); // mando la señal de trigger

HAL\_Delay(0.00001);//tengo que poner este delay para que genere una señal de trigger de 10 us de duracion

HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, *GPIO\_PIN\_RESET*); // apago el trigger

\_\_HAL\_TIM\_ENABLE\_IT(&htim1, TIM\_IT\_CC1);

}

}

**void** **HAL\_TIM\_IC\_CaptureCallback**(TIM\_HandleTypeDef \*htim)//esta funcion se llama cuando se campture un flanco ascendente o descendente

{

**if** (htim->Channel == *HAL\_TIM\_ACTIVE\_CHANNEL\_1*) // Pregunto si la interrupcion es en el channel1

{

**if** (Valor\_Capturado==0) // si no capturo el primer valor...

{

IC\_Val1 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); // si detecto un flanco ascendente, la marca de tiempo se almacena en IC\_Val1

Valor\_Capturado = 1;

\_\_HAL\_TIM\_SET\_CAPTUREPOLARITY(htim, TIM\_CHANNEL\_1, TIM\_INPUTCHANNELPOLARITY\_FALLING);// Cambio la polaridad para que capture flancos descendentes

}

**else** **if** (Valor\_Capturado==1) // Si el primer valor ya fue capturado...

{

IC\_Val2 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); // si detecto un flanco descendente, la marca de tiempo se almacena en IC\_Val2

\_\_HAL\_TIM\_SET\_COUNTER(htim, 0); // Reseteo el contador

//calculamos la diferencia entre las dos marcas de tiempo

**if** (IC\_Val2 > IC\_Val1)

{

Diferencia = IC\_Val2-IC\_Val1;

}

**else** **if** (IC\_Val1 > IC\_Val2)

{

Diferencia = (0xffff - IC\_Val1) + IC\_Val2;//0xffff se refiere al valor maximo de conteo del timer

}

Distancia = Diferencia /58;//formula que me da el datasheet para pasar a cm

Valor\_Capturado = 0;

\_\_HAL\_TIM\_SET\_CAPTUREPOLARITY(htim, TIM\_CHANNEL\_1, TIM\_INPUTCHANNELPOLARITY\_RISING);//al terminar todos los calculos, cambio la polaridad para capturar flanco ascendente nuevamente

\_\_HAL\_TIM\_DISABLE\_IT(&htim1, TIM\_IT\_CC1);

// Valor\_Capturado = 0;

// int a;

**strcpy**(tx\_buffer\_HC, "");//limpio el buffer

//

// strcpy(svalue, "");

**sprintf**(tx\_buffer\_HC, "Distancia = %d cm \n", Distancia);

// HAL\_UART\_Transmit\_IT(&huart6, (uint8\_t \*)tx\_buffer\_HC, strlen(tx\_buffer\_HC));

}

}

* **UART\_Tx\_Rx():** es la función que se encarga de transmitir y recibir los datos entre la microcontroladora y la app móvil.

**void** **UART\_Task**(**void**){

**if**(counter\_Uart>=999){

counter\_Uart=0;

**strcpy**(tx\_buffer\_Uart, "");//limpio el buffer

**strcat**(tx\_buffer\_Uart,tx\_buffer\_HC);//concateno lo que tiene tx\_buffer\_HC al tx\_buffer\_Uart

HAL\_UART\_Transmit\_IT(&huart6, (uint8\_t \*)tx\_buffer\_Uart, **strlen**(tx\_buffer\_Uart) );

}

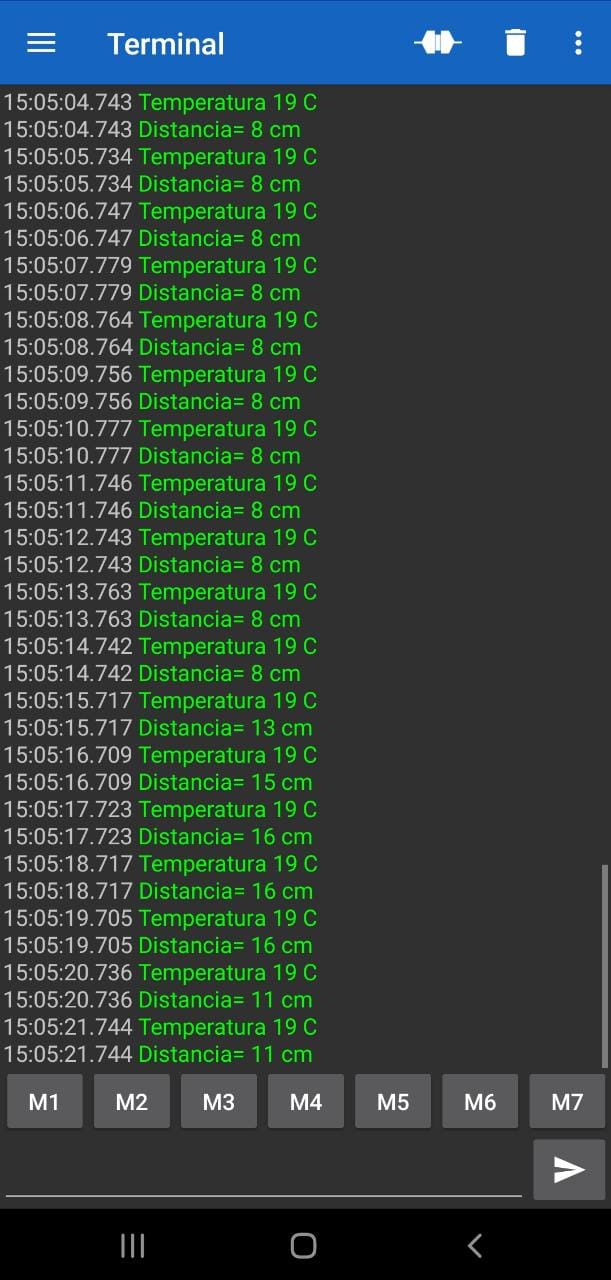
}

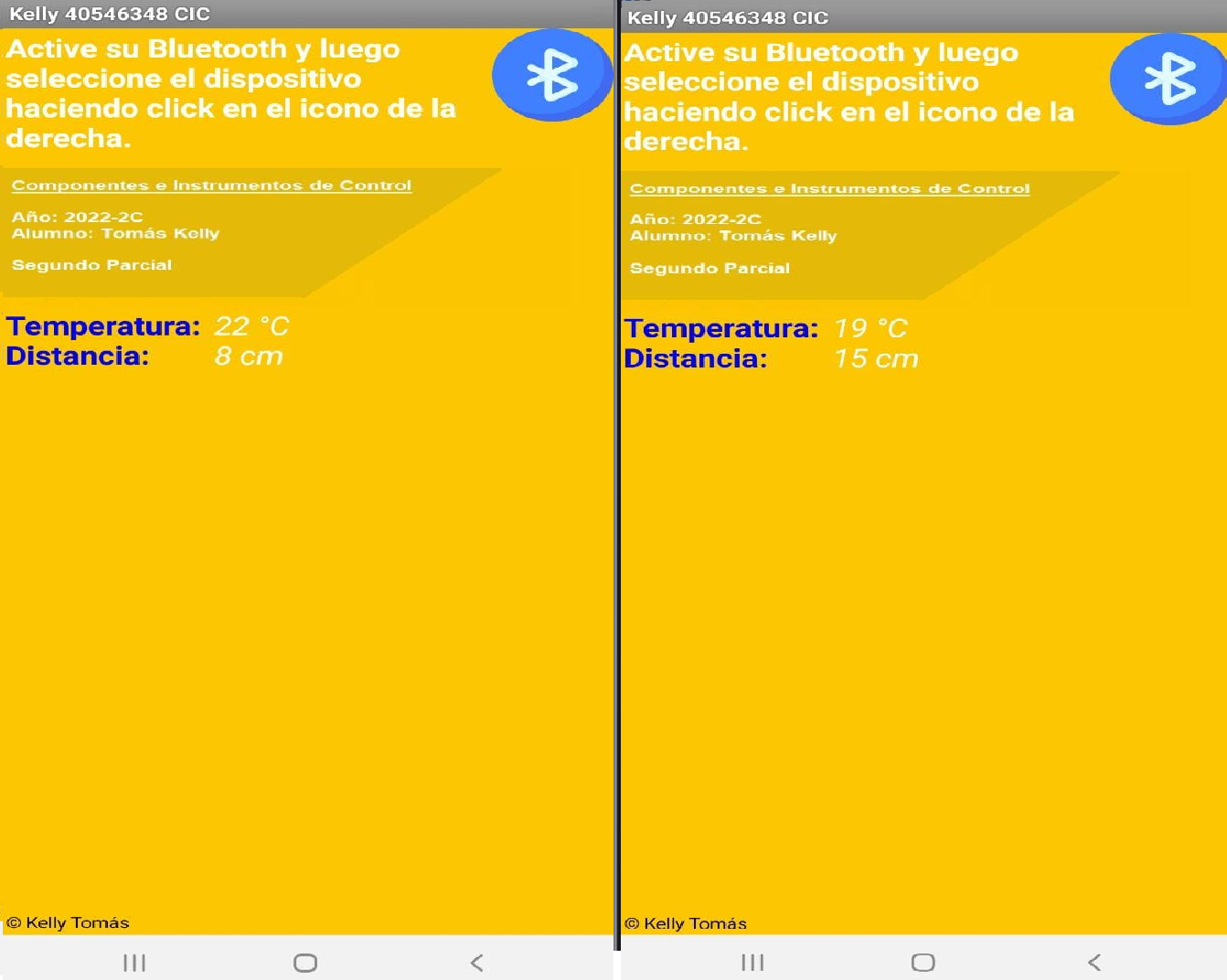
### Interfaz

La interfaz que va a utilizar desde el celular el encargado/ supervisor fue realizada con el programa appInventor. Está disponible para dispositivos Android descargándola desde el siguiente link: [KellyCIC.apk](https://drive.google.com/file/d/1_kjm2rdEA2qGD0AHUSFCmO9E0FUD4v3j/view?usp=sharing). La aplicación cuenta con la información de la distancia que capta el sensor.

Previamente se utilizó la aplicación móvil “Serial Bluetooth Terminal” como terminal de pruebas.

La primera imagen se trata de la aplicación “Serial Bluetooth Terminal” y la segunda corresponde a la aplicación móvil que fue desarrollada.





**IMAGEN TOMADA DE EJEMPLO**

**IMAGEN TOMADA DE EJEMPLO**

# Hardware

### Componentes

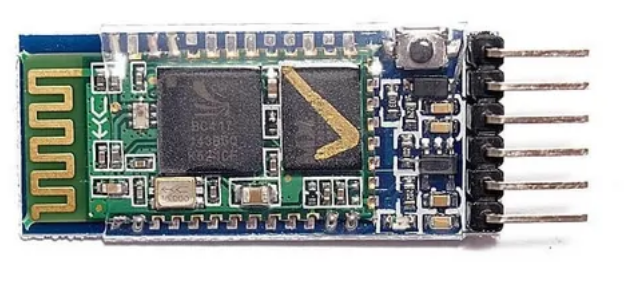
Microcontrolador (STM32F411E-DISCO Discovery Kit): es el cerebro del sistema, hacia él llega la información proveniente de distintos tipos de sensores y se encarga de procesar esos datos para generar una respuesta acorde al problema que debemos resolver, esta respuesta se ve reflejada en distintos dispositivos de salida. Para grabar las órdenes que luego ejecutará el microcontrolador, programamos en lenguaje C/C++. El IDE que utilizamos es el que nos proporciona la marca de la microcontroladora STM32CubeIDE(v. 1.6.1). Decidimos utilizar esta microcontroladora debido a su gran alcance hoy en día y a la gran cantidad de periféricos que nos otorga.



Características:

* Microcontrolador STM32F411VET6, 512KB de memoria flash, 128KB de RAM en 100 pines LQFP.
* ST-LINK/V2 integrado con selector de modo de selección para usar el kit como un ST-LINK/V2 independiente.
* Diseñado para ser alimentado por USB o un suministro externo de 5V.
* Puede suministrar aplicaciones con 3V y 5V.
* L3GD20: Giroscopio de salida digital de tres ejes y sensor de movimiento ST MEMS
* LSM303DLHC: Sistema en paquete ST MEMS que incluye un sensor de aceleración lineal digital 3D y un sensor magnético digital en 3D.
* MP45DT02: Sensor de audio ST MEMS, micrófono digital omnidireccional.
* CS43L22: audio DAC con controlador de altavoz clase D integrado.
* Cuatro LED de usuario.
* Dos botos pulsadores (usuario y reset).
* USB OTG con conector micro-AB (STMicroelectronics, 2017).

**Módulo Bluetooth (HC-05):** permite una conexión inalámbrica entre dispositivos alejados, lo cual facilita en términos de practicidad la comunicación. Además, les permite a los operadores una mayor movilidad por el establecimiento. La limitante es una distancia máxima de aproximadamente 10 metros.



**Sensor de Proximidad (HCSR-04):** mide la distancia entre el sensor y un objeto calculando el tiempo que tarda la onda de ultrasonido en salir del emisor rebotar con el objeto e incidir en el receptor.

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

**Alimentación:** El módulo que abarca el microcontrolador, los sensores y el módulo bluetooth estarán conectados a una fuente de 5V 10A conectada a la red de distribución. Se tomó en cuenta que todos los componentes (contando sensores, módulo bluetooth y placa controladora) operan en 5V.

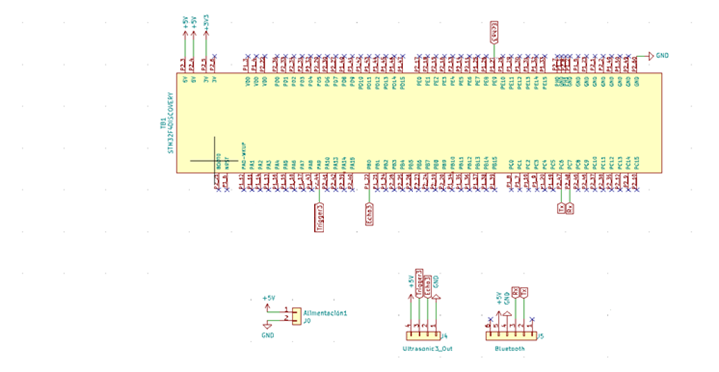
### Circuito en Protoboard

El circuito prototipo fue realizado sobre una protoboard. Fue diseñado de forma tal que la microcontroladora pueda ser reutilizable para otros usos independientes al proyecto, ó en caso de que esta cuente con una falla solo haya que reemplazarla por una que funcione correctamente, sin tener que cambiar componentes.

Pinout adoptado:

HC-05: PC6-Tx  
 PC7-Rx

HC-SR04: PA9-Trigger  
 PE9-Echo

Esquemático:

### Montaje

El montaje del proyecto fue realizado en una protoboard. A continuación, se adjuntan las imágenes de las distintas etapas en el desarrollo del mismo:

Adjuntar imágenes del proyecto en la protoboard. La parte del sensor de proximidad. Y el modulo bluetooth.

# Desarrollo

### Medidores de Distancia Ultrasónicos:

En el proyecto se utilizó un sensor ultrasónico HC-SR04. El periférico utilizado para poder controlar dicho componente es mediante la configuración adecuada de un timer.

Como su nombre lo indica, el sensor ultrasónico mide distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción. Dicha distancia nos la devuelve en una variable la cual, mediante una fórmula matemática que nos provee el fabricante en la hoja de datos, la convertimos a una distancia en centímetros.

### Comunicación Bluetooth:

El proyecto consta de comunicación bluetooth para poder realizar la recepción y transmisión de datos con la app del Smartphone.

Para realizar dicha comunicación se utilizó un módulo de bluetooth HC-05 que cumple con las especificaciones del estándar Bluetooth 2.0 a 2.4 GHz que es perfectamente compatible con celulares o Smartphone Android. El mismo fue configurado por UART. Además de utilizar la UART para la configuración de dicho componente, se utilizó también un Timer de propósito general con el fin de ser utilizado para sincronizar los datos enviados al celular.

Los módulos Bluetooth se pueden comportar como esclavo o maestro, los cuales sirven para escuchar peticiones de conexión y otros para generar peticiones de conexión. Es por este motivo que para poder utilizar el HC-05 como esclavo en la microcontroladora STM32F411 previamente tuvimos que configurar el módulo bluetooth en un Arduino UNO por parámetros AT para que el mismo opere como esclavo y de esta forma poder realizar la comunicación deseada con la aplicación móvil.

# Conclusiones

Lo que se puede destacar del proyecto es que se pudo poner en funcionamiento la mayoría de los periféricos que nos brinda la stm32f411(UART, Timer, GPIO).

# Referencias

* [STM32F411-DATASHEET](https://www.st.com/resource/en/user_manual/dm00148985-discovery-kit-with-stm32f411ve-mcu-stmicroelectronics.pdf)
* [HC-05-DATASHEET](http://www.electronica60norte.com/mwfls/pdf/newBluetooth.pdf)
* [HC-SR04-DATASHEET](https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf)
* [USERMANUALSTM32F4](https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf)

# Repositorio

Todo el contenido del proyecto, ya sea código, imágenes, pruebas, datasheets y hasta este mismo informe se puede encontrar en el siguiente repositorio de GitHub: [click aquí](https://github.com/toomy97/CIC_Kelly).