Tutorial 3: Comunicación Bluetooth

Curso 2017-18

CONTENIDO

INTRODUCCIÓN

En este tutorial explicaremos cómo utilizar una de las tantas librerías disponible en Arduino para mover un servo desde el móvil, gracias al uso de un módulo Bluetooth HC-06. Se supondrá conocido el uso de la librería **Serial** para la comunicación por UART con el HC-06.

CONCEPTOS BÁSICOS

BLUETOOTH

Bluetooth es un estándar de comunicaciones que posibilita el intercambio de datos entre dispositivos a distancias cortas, promovido en principio por *Ericcson*, y más tarde por el consorcio de empresas *Bluetooth* Special Interest Group (SIG). En principio se ideó para eliminar los cables en las conexiones serie RS-323, pero actualmente es una de las tecnologías más utilizadas en redes de área personal (PAN) y una clara competidora en el mercado del *Internet de las Cosas* (IoT). Existen diferentes versiones del estándar, desde la 1.0 a la 5.0, aunque el módulo HC-06 que utilizaremos implementa la versión 2.0.

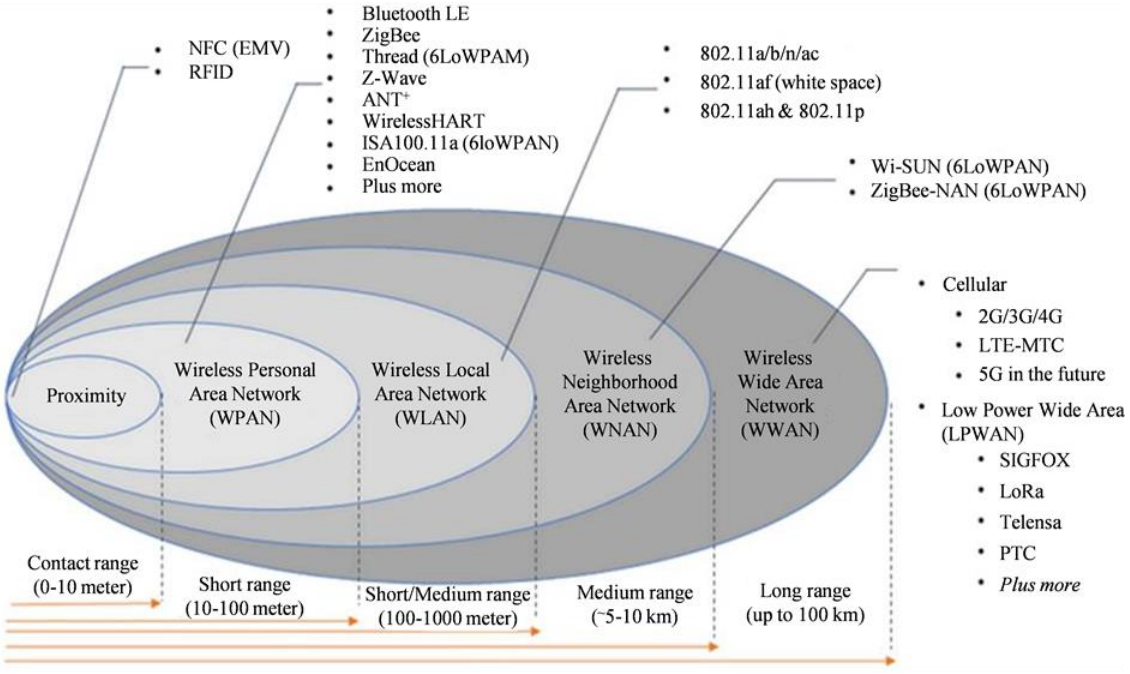


Ilustración 1 – Tecnologías de comunicación inalámbrica y su alcance.
Fuente: Mahmoud, M. and Mohamad, A. (2016) A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT) Applications. *Advances in Internet of Things*, 6, 19-29.

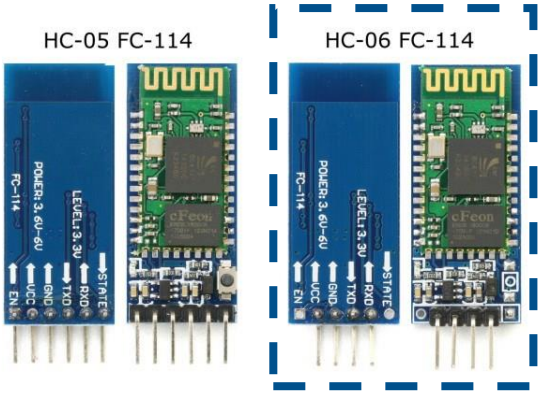
Esta versión utiliza la banda de 2.4GHz (de uso libre), y una combinación de modulaciones GFSK y PSK, además de técnicas de salto en frecuencia. De esta forma, permite formar redes de hasta ocho dispositivos (piconets), llegando a alcanzar velocidades teóricas de transmisión de 2.1Mbps a distancias de hasta 10m con 4dBm de potencia máxima. En estas redes, o piconets, uno de los dispositivos tomará el rol de *master*, mientras que los demás serán *esclavos*, y deberán transmitir cómo y cuándo les diga el *master* para evitar interferencias.

Cada dispositivo Bluetooth tendrá una dirección MAC de 48 bits, que le identificará unívocamente. El método de conexión entre dos dispositivos Bluetooth sigue un procedimiento bien definido que establece un proceso de **descubrimiento de dispositivos y emparejado**, necesitando en la mayoría de los casos de una clave de seguridad conocida por ambos extremos.

COMPONENTES

MÓDULO BLUETOOTH HC-06

Este módulo de bajo coste integra toda la circuitería necesaria para transmitir utilizando Bluetooth 2.0: chip CSR Bluetooth, antena, oscilador, regulador de tensión, memoria flash… Tal y como mostraremos en el ejemplo práctico, puede comunicarse vía UART con un microcontrolador como el MSP430, o Arduino, para pasarle fácilmente la información que se desee transmitir por Bluetooth. También puede configurarse a través de unos comandos AT predefinidos que podéis consultar en el datasheet.

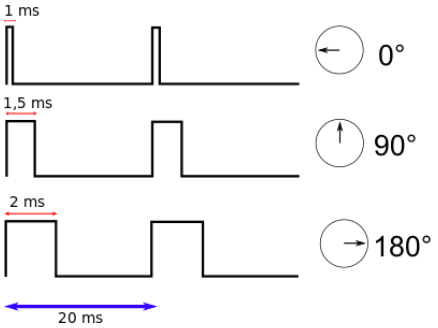


No deben confundirse los modelos HC-05 y HC-06: el que nosotros utilizaremos, el HC-06, tiene 4 pines y sólo es capaz de funcionar como esclavo, mientras que el HC-05 dispone de 6 pines y puede configurarse para operar como master o como esclavo.

SERVOMOTOR RS-2

Un servomotor es básicamente un motor eléctrico en el que, además de la velocidad, podemos controlar la posición exacta de giro dentro del rango de operación. Usualmente está formado por un motor de corriente continua, una caja reductora, un potenciómetro y una circuitería de control. Normalmente, son pocos los servos comerciales que pueden llegar a girar 360°, pero por Internet se pueden encontrar numerosos tutoriales para modificar un servo normal y lograr que gire totalmente de forma continua.

Se utilizan pulsos PWM para controlar el giro, por lo que el ángulo final dependerá del ancho del pulso PWM. Por ello, es importante conocer parámetros del servo que usemos como el rango de rotación y el ancho máximo y mínimo del pulso de control para calcular exactamente la posición. En este tutorial dejaremos que la librería Servo maneje el pulso PWM según el ángulo que le indiquemos, directamente, ya que no requerimos tanta precisión.



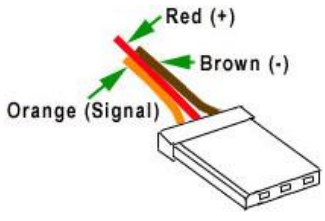


Ilustración 2 - Conector JR

Otra cuestión importante a la hora de conectar nuestro servo es el tipo de conector que tenemos, ya que existen diversos tipos. Esencialmente, siempre dispondremos de dos cables para alimentación y tierra, y un cable para la señal de control, pero los colores y la posición pueden variar. En nuestro caso, el RS-2 utiliza un conector JR, cuyas señales corresponden con las que se pueden observar en la Ilustración de la izquierda.

EJEMPLOS

En este ejemplo controlaremos un servo a través de una app, mediante Bluetooth, haciéndolo girar a la derecha o a la izquierda con comandos simples. Para facilitar el manejo por PWM del servo haremos uso de la librería Servo de Arduino.

En primer lugar, debemos definir un objeto de tipo "Servo", que será algo así como una entidad asociada al pin dónde vayamos a establecer la señal del control de nuestro servo. Esto lo haremos con la siguiente sentencia que irá junto a las definiciones de las variables, fuera de **setup** y **loop**.

```
Servo myservo;
```

Habrá que tener en cuenta que como máximo podrán definirse ocho objetos de tipo Servo. En este ejemplo utilizaremos las siguientes funciones:

- void **myservo**.**attach** (pin): asocia al **pin** elegido el objeto de tipo servo **myservo**, para poder utilizarlo más tarde con las demás funciones de la librería. A la hora de elegir los pines, la única condición es que pueda utilizarse PWM.
- void **myservo**.**write**(***value***): esta función recibe en ***value*** el valor del ángulo deseado en grados y configura un pulso PWM de la anchura necesaria para que el servo se coloque en la posición elegida. Si el valor que le indicamos es mayor de 200°, lo interpreta como el ancho del pulso PWM deseado en milisegundos.

Además, será necesario descargar una app de terminal Bluetooth en nuestro teléfono móvil, y configurarla para que el dispositivo HC-06 se empareje con nuestro móvil. En el siguiente código se leen a través de la UART los comandos que recibimos mediante Bluetooth con el HC-06 y se elige el sentido del giro dependiendo de si se recibe un **+** o un **-**. Es importante que revisemos la configuración de la app del terminal para que no inserte caracteres como **/c** o **/n** al final de cada comando, para que nuestro código pueda identificarlo sin problemas.

```
/*
  Comunicaciones - Ejemplo 1
  Controla la posición de un servo a través de comandos recibidos
  por el módulo Bluetooth.

  ESIBot 2018
*/

#include <Servo.h>

#define SERVO_PIN 9    // Pin dónde nuestro servo recibirá la señal de
control
#define PASO 20        // Paso en grados para mover la posicion del
servo

Servo myservo;  // Para utilizar las funciones de la librería servo
creamos un objeto Servo
                // se pueden crear hasta ocho objetos Servo
```

```
int pos = 0;    // Variable para guardar la posición del servo en grados (0-180)
char c='a';     //Inicializamos a un valor erróneo para que no se mueva el servo por error
void setup(){
  Serial.begin(9600);
  Serial.println("Inserte comandos:");

  myservo.attach(SERVO_PIN);  // Asociamos el pin dónde nuestro servo recibirá la señal de control
}

void loop(){
  if (Serial.available()){
    // Se espera a recibir un comando formado por un sólo caracter
    c = Serial.read();
    Serial.print("Comando: ");
    Serial.println(c);
   //Comandos
    if (c=='+'){
      //Aumentamos la posición del servo en grados
      if(pos < 180){
        pos += PASO;
        myservo.write(pos);
        delay(15);      // Espera 15ms para que al servo le de tiempo a alcanzar la posición
      }
    }
    else if (c=='-'){
      //Disminuimos la posición del servo en grados
      if(pos>=1){
        pos -= PASO;
        myservo.write(pos);
        delay(15);    // Espera 15ms para que al servo le de tiempo a alcanzar la posició
      }
    }
    else{
      Serial.println("Inserte comando correcto");
    }
  }
```

```
}
```

EJEMPLOS RESUELTOS

EJEMPLO RESUELTO BLUETOOTH 1: ENCENDIDO DEL LED RGB

En este caso, utilizaremos lo aprendido en el Tutorial sobre la UART para encender y apagar el led RGB (de ánodo común) con nuestro Arduino con comandos más complejos, de varios caracteres. El montaje utilizado es el mismo que el utilizado en el ejemplo de PWM con el led RGB, incluyendo la conexión con el módulo Bluetooth como en el ejemplo anterior.

```
/*
  Comunicaciones - Ejemplo 2
  Controla el led del Launchpad a través de comandos recibidos
  por el módulo Bluetooth.

  ESIBot 2018*/

#define LED_R 3
#define LED_G 5
#define LED_B 6

// variables que cambiarán:
String inputString = "";         // una cadena que almacenará lo que
recibamos
boolean stringComplete = false;  // indica si ha llegado una línea

void setup() {
  // configuramos los pines:
  pinMode(LED_R, OUTPUT);
  digitalWrite(LED_R, HIGH);
  pinMode(LED_G, OUTPUT);
  digitalWrite(LED_G, HIGH);
  pinMode(LED_B, OUTPUT);
  digitalWrite(LED_B, HIGH);


  // inicializamos el puerto serie a 9600:
  Serial.begin(9600);
  Serial.println("Inserte comandos:");
  // reservamos 50 bytes para inputString:
  inputString.reserve(50);
}
```

```
void loop() {
    // Se espera a recibir un comando
  if (stringComplete) {
    Serial.println("Comando: " + inputString);

    if(inputString.equalsIgnoreCase("LRON")) {
      digitalWrite(LED_R, LOW);
      delay(50);
      } else if (inputString.equalsIgnoreCase("LROFF")) {
      digitalWrite(LED_R, HIGH);
      delay(50);
      } else if (inputString.equalsIgnoreCase("LGOFF")) {
      digitalWrite(LED_G, LOW);
      delay(50);
      } else if (inputString.equalsIgnoreCase("LGOFF")) {
      digitalWrite(LED_G, HIGH);
      delay(50);
    } else {
      Serial.println("Inserte comando correcto");
    }

    // borramos la cadena:
    inputString = "";
    stringComplete = false;
  }
}

/*
  Un SerialEvent tiene lugar cuando llegan nuevos datos
  en el pin RX de la UART. Esta función se ejecuta entre
  cada iteración del bucle loop(), por tanto usar funciones
  de delay() dentro del bucle puede hacer que la respuesta
  sea más lenta. Múltiples bytes de datos pueden estar
  disponibles al ejecutar esta función.
 */
void serialEvent() {
  while (Serial.available()) {
    // leemos el nuevo byte:
    char inChar = (char)Serial.read();

    // si el caracter de entrada es un \n, lo indicamos
```

```
    // en la variable para que el bucle principal pueda
    // hacer algo en consecuencia:
    if (inChar == '\n') {
      stringComplete = true;
    } else {
      // en caso de que sea otro caracter, lo añadimos inputString:
      inputString += inChar;
    }
  }
}
```

MODIFICACIONES Y EJERCICIOS PROPUESTOS

- **Modificaciones propuestas para el Ejemplo Propuesto**:
  - Cambiar los comandos simples de un solo carácter por comandos más complejos, que incluyan por ejemplo el ángulo final deseado.
  - Utilizar las funciones analogWrite() en vez de la librería Servo para manejar los servos.
- **Modificaciones propuestas para el Ejemplo Resuelto 1**:
  - Controlar el led RGB, distinguiendo los tres colores y controlando la intensidad de cada uno de ellos.
- **Ejercicios Propuestos**:
  - Utilizando la comunicación Bluetooth, mostrar por pantalla el resultado de distintos sensores.