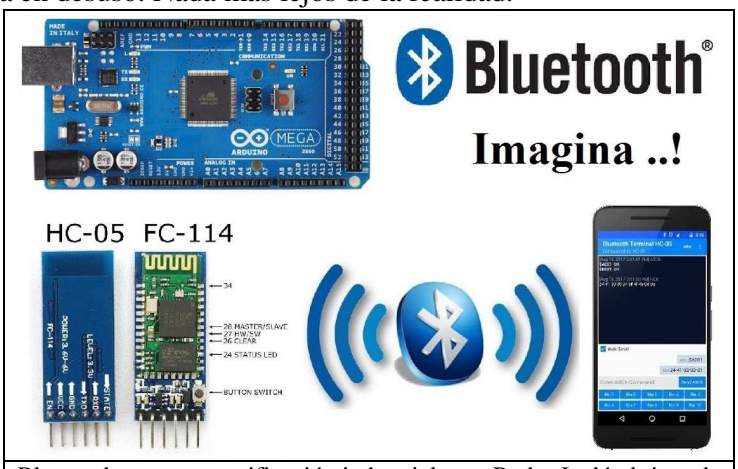
MODULO BLUETOOTH

Uno de los módulos más interesantes que podemos integrar con Arduino, es sin duda el de comunicación Bluetooth. Es común escuchar que el Bluetooth es una tecnología anticuada, que se usaba para trasmitir datos entre dispositivos, y que actualmente está en desuso. Nada más lejos de la realidad.

Bluetooth tiene la enorme ventaja de estar ya integrado de fábrica en la mayoría de dispositivos Portátiles, Tablets, y Smartphones llevan integrado Bluetooth. Además, su uso es independiente del sistema operativo (Windows, Linux, Mac o Android).

Esto convierte a la tecnología Bluetooth en uno de los mejores medios para comunicarnos de forma inalámbrica con Arduino. Y adicionalmente el modulo Bluetooth, es muy económico y ampliamente distribuido en los comercios, por ende, es muy fácil adquirir. Además es sumamente sencillo de usar.



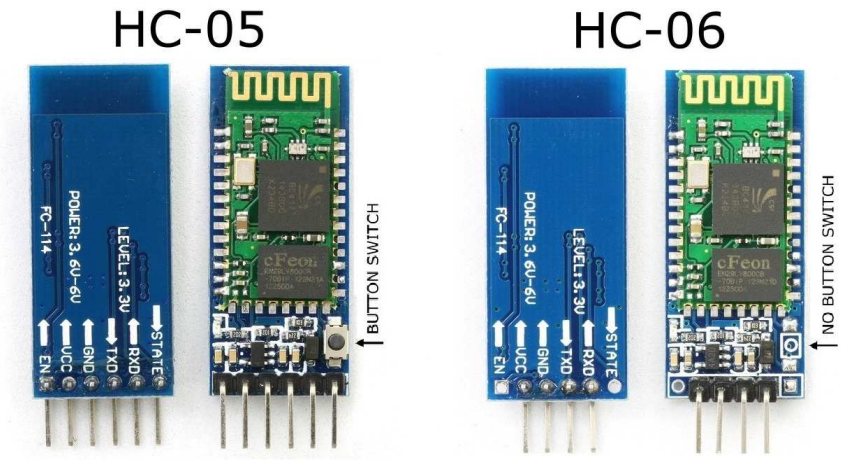
Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) creado por "**Bluetooth Special Interest Group, Inc.**" que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz.

Los usos son muy variados, Por ejemplo, podemos emplearlo para controlar un robot desde un celular o Tablet, o para comunicar dos dispositivos controlados por Arduino. Incluso es posible programar Arduino de forma inalámbrica a través de Bluetooth.

Algo que hace un poco más interesante el Bluetooth, es la cantidad de aplicaciones que se han desarrollado para los celulares, que nos facilitan la comunicación. Entre otros y quizás los más emocionantes podemos nombrar a los Joystick digitales y aplicaciones de reconocimiento de Voz, que podemos instalar en nuestro celular y configurar para que envíe comando a nuestro dispositivo Arduino.

Modelos de Módulos Bluetooth (que usaremos)

Para Arduino, en los comercios, podremos encontrar varios modelos de módulos Bluetooth. En la presente guía usaremos el modulo HC-05 y el modulo HC-06. Ambos modelos nos servirán para conectar con Arduino, pero hay diferencias entre ellos.



Modelos de Módulos Bluetooth compatibles con Arduino

La diferencia entre ambos módulos es que el HC-06 sólo permite recibir comunicaciones (**slave**) mientras que el HC-05 puede recibirlas e iniciarlas (**master and server**). Por tanto el módulo HC-05 es superior en características técnicas.

Estos módulos, se pueden diferenciar a simple vista, ya que el modelo HC-06 tiene 4 patas o pines y el HC-05 tiene 6, además el modelo HC-05, tiene un botón switch, que debemos pulsar para poner el módulo en modo configuración (comandos AT - que veremos un poco más adelante).

IMPORTANTE: Existen múltiples fabricantes de módulos Bluetooth, por lo que es posible encontrar algunas diferencias en las características que a continuación expongo.

Características Técnicas (módulos HC-06 y HC-05)

Características compartidas:

- ✓ Compatible con el protocolo Bluetooth V2.0.
- ✓ Voltaje de alimentación: 3.3VDC – 6VDC.
- ✓ Voltaje de operación: 3.3VDC.
- ✓ Velocidad configurable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- ✓ Velocidad de comunicación por defecto 960 Baudios
- ✓ Tamaño del módulo: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)
- ✓ Corriente modo sleep: < 1mA
- ✓ Puede configurarse mediante comandos AT (Deben escribirse en mayúscula)
- ✓ Chip de radio: CSR BC417143
- ✓ Frecuencia: 2.4 GHz, banda ISM
- ✓ Modulación: GFSK (Gaussian Frequency Shift Keying)
- ✓ Antena de PCB incorporada
- ✓ Potencia de emisión: ≤ 6 dBm, Clase 2
- ✓ Alcance 5 m a 10 m
- ✓ Sensibilidad: ≤ -80 dBm a 0.1% BER
- ✓ Velocidad: Asincrónica: 2.1 Mbps (max.)/160 kbps, sincrónica: 1 Mbps/1 Mbps
- ✓ Seguridad: Autenticación y encriptación (Password por defecto: 1234)
- ✓ Perfiles: Puertoserial Bluetooth
- ✓ Temperatura de operación: -20 ºC a +75 ºC

HC-06	HC-05
✓ **Modo esclavo (Solo puede operar en este modo)** ✓ Módulo montado en tarjeta con regulador de voltaje y 4 pines suministrando acceso a VCC, GND, TXD, y RXD **IMPORTANTE:** Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.	✓ **Puede configurarse como maestro, esclavo, y esclavo con autoconexión (Loopback) mediante comandos AT** ✓ Módulo montado en tarjeta con regulador de voltaje y 6 pines suministrando acceso a VCC, GND, TXD, RXD, KEY y status LED (STATE) ✓ Consumo de corriente: 30 a 50 mA ✓ El pin RX del módulo requiere resistencia de pull-up a 3.3 V (4.7 k a 10 k). Si el microcontrolador no tiene resistencia de pull-up interna en el pin Tx se debe poner externamente. ✓ Niveles lógicos: 3.3 V. Conectarlos a señales con voltajes mayores, como por ej. 5 V, puede dañar el módulo

Ambos módulos cuestan prácticamente lo mismo. Entonces si el HC-05 es superior en características, lo normal sería que elijamos comprar el módulo HC-05.

Conectando el Modulo Bluetooth

(Modelos HC-05 y HC-06)

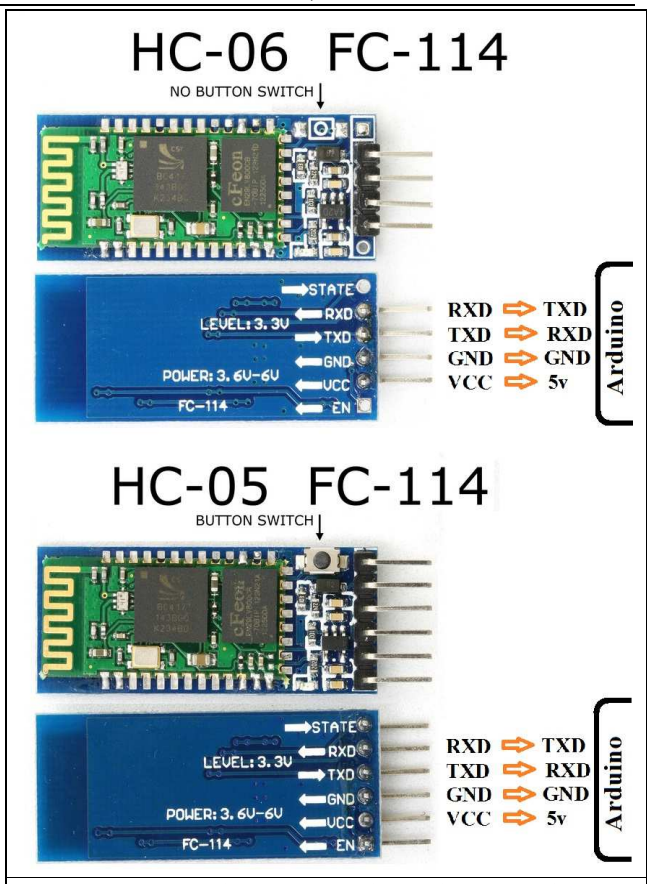
Utilizar el módulo de Bluetooth requiere el uso de un puerto serie de nuestra placa Arduino. Por tanto, mientras usemos el módulo de Bluetooth no podremos usar el puerto serie en las placas modelo **Uno, Mini, y Nano**. En el modelo **Mega** no tiene este problema, ya que incorpora 4 puertos de serie (Esto lo veremos en detalle).

Mientras estemos cargando un nuevo programa en la placa Arduino tenemos que desconectar el módulo Bluetooth, dado que la programación se realiza a través del puerto serie.
Si realmente necesitamos ambas comunicaciones podemos emplear la librería SoftSerial para establecer una comunicación de puerto serie por cualquier pareja de pins digitales, aunque ello supondrá un costo adicional de tiempo de proceso en Arduino.

La conexión es sencilla. Alimentamos mediante Vcc (5v) y GND. Posteriormente conectamos el TXD (pin de transmisión) y RXD (pin de recepción) a los opuestos de la placa Arduino (cada TXD a un RXD). Así quedarían las conexiones del módulo, con los pines de Arduino.

Por defecto suele venir configurado para trabajar a 9600 baudios, El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

Luego, el nombre del dispositivo, contraseña y otras configuraciones, podrán ser cambiadas mediante comandos AT ya sea desde el propio Arduino o desde el ordenador. Los llamados comandos **AT** los veremos detenidamente un poco más adelante.



Como conectar los Módulos Bluetooth con Arduino.
Los dos pines libres en el modelo HC-05 los usaremos más adelante para la parte de configuración (Comandos AT)

Consumo y alimentación de un Modulo Bluetooth

El consumo de ambos módulos, HC-05 y HC-06, son mínimos y nuestro Arduino es capaz de alimentar a cualquiera de los dos sin problemas. Esto luego nos facilitara configurarlos, en especial al HC-05, que requiere que unos de los pines este prendido (HIGH) al iniciar el modulo.

Para saber más sobre reconocimiento de texto y las funciones que se usan, debe consultar la guía "Anexo_01_Procesamiento_de_Texto".

1) Enviar un número por Bluetooth y Arduino hará parpadear un Led conectado a la placa, el número de veces que indique el número enviado (entre 1 y 9).

Enviando comandos desde nuestro celular, Arduino lo recibe y Hacemos parpadear un LED. El comando enviado desde el celular es un número que deberá estar previamente configurado: Cuando apretemos una tecla en el celular, se enviará el número.

(Programa "050_Bluetooth_01_Led_01")

```
1  const int PinLed = 10; // Pin de control para el LED
2  int Comando, i;
-
3  void setup( ){
4    pinMode(PinLed, OUTPUT);
5    BT.begin(9600);  //Velocidad puerto módulo Bluetooth
6    Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
7  }
```

Este programa funcionará con cualquiera de los modelos de Modulo Bluetooth que se conecten con Arduino: HC-05 o HC-06

```
-
8   void loop( ){
9     if(BT.available( )){  // Si hay algo pendiente de ser leído en Modulo Bluetooth
10      Comando = BT.read( );  // Leo el modulo Bluetooth
11      if (Comando >= '1' && Comando <= '9'){ //si el Comando esta entre '1' y '9'
12        Comando = Comando - '0';      //restamos el valor '0' para obtener el número enviado
13        for( i=0; i< Comando; i++){
14          digitalWrite(PinLed, HIGH);
15          delay(500);
16          digitalWrite(PinLed, LOW);
17          delay(500);
18        }
19      }
20    }
21  }
```

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

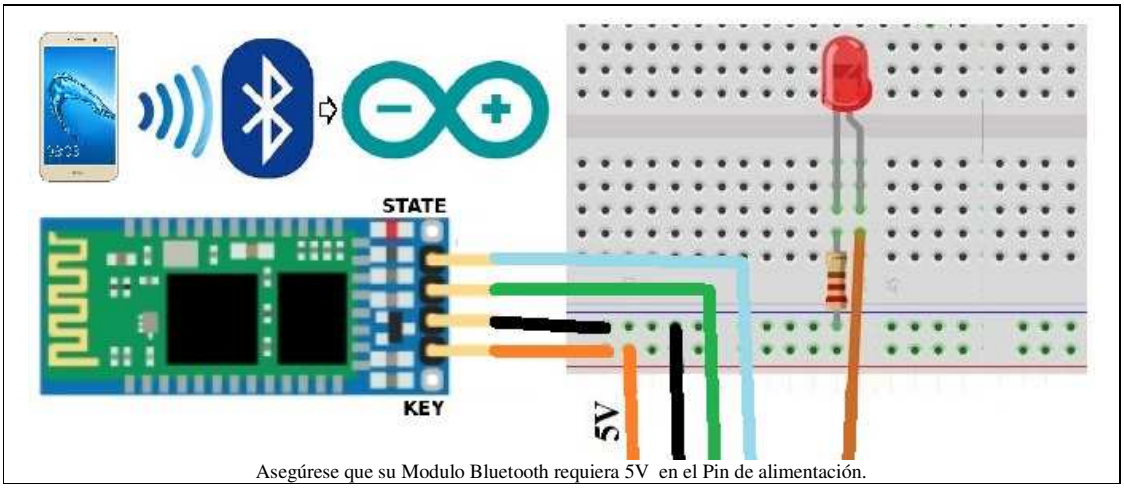
Para Seleccionar e instalar aplicación que permita enviar comando desde el celular, Ver **Apéndice A - Comandos Bluetooth desde un Celular**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Led - 1 Resistencia de 470Ω. - 2 Cables Macho/Macho - 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).

Importante: que para poder enviar el programa a nuestra placa Arduino, el modulo Bluetooth debe estar apagado (La luz roja del modulo debe estar apagada) - Simplemente desconecta el cable de 5v que alimenta el modulo, luego enviar el programa. Recuerda conectar el cable de 5v cuando ya hayas subido el programa. :-)

Los cables deberán ser conectados: Abajo, de Izquierda a Derecha, Cable Naranja: a 5v de Arduino. Cable Negro a GND. Cable Verde: Corresponde en el Modulo Bluetooth a TX y se debe conectar a RX en Arduino. Cable Celeste, Corresponde a del Modulo Bluetooth a RX y se debe conectar a TX de Arduino. Cable Marrón (Ultimo de la derecha), se debe conectar al pin que configuramos para el Led (Línea 01 del programa). **Si te queda alguna dudo como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**



Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

A tener en cuenta: deberemos conectar el modulo Bluetooth, en los pines RX y TX, que son los puertos por donde Arduino se conecta con el monitor del puerto serie. Lamentablemente por ahora debemos elegir si conectamos el modulo Bluetooth o enviamos y recibimos datos desde y hacia el monitor. A así que a prestar mucha atención. Luego solucionaremos este problema.

USO DE LOS PINES TX Y RX

Si nos fijamos con detenimiento, en nuestra placa Arduino, podremos ver que el PIN 0, que está identificado como RX y el PIN 1, que está identificado como TX. Esto también podríamos verlo en el Plano o PinOut de nuestra placa. En algunos modelos de Arduino, pueden verse como RXD y TXD. Tal es el caso de la placa modelo MEGA.

Segmento imagen PinOut Arduino UNO

Y para que sirven estos PINES (RX y TX)?

Bueno es muy simple: La función de estos Pines, es la de comunicación de nuestra placa (modulo) con otros dispositivos. Y si hacemos una analogía con el cuerpo humano, TX seria nuestra **boca**, por donde hablamos y RX nuestro **oído**, por donde escuchamos.

Pin	Nombre	Descripción
0	RX	Es el Pin usado para la Recepción de datos enviados por el otro dispositivo.
1	TX	Es el Pin usado para Transmisión de datos (Envío), al otro dispositivo.

En las placas Arduino, los pines 0 y 1 se usan para comunicarse con una PC, Aunque también sirven para comunicarse con otros dispositivos, por ejemplo, con el Modulo Bluetooth. **El único inconveniente, es que no se pueden usar para dos cosas simultáneamente**. Nos comunicamos con la PC o Nos comunicamos con el Módulo Bluetooth (u otro módulo que utilice estos pines) y perdemos la conexión con la PC.

Para Solucionar este inconveniente, Arduino Provee una librería llamada **SoftwareSerial**, que se encuentra en el IDE y nos permite definir otros pines con idéntica funcionalidad a los pines RX y TX, para comunicarnos con otros dispositivos simultáneamente. (**Ver Apéndice B - Librería SoftwareSerial**)

Usando la Librería **SoftwareSerial,** solucionamos el problema que teníamos al momento de subir el programa a nuestro Arduino, mientras el modulo Bluetooth esta conectado y funcionando.

Ahora que ya podemos leer el Modulo Bluetooth, y escribir en el monitor del puerto serie (simultáneamente), haremos un programa que nos facilita las cosas, ya que nos mostrará por el monitor del puerto serie los caracteres (comandos) que ingresan desde el modulo Bluetooth.

Para saber más sobre reconocimiento de texto y las funciones que podríamos usar, debe consultar en el "Anexo_01_Procesamiento_de_Texto".

2) Mostrar en el monitor del puerto serie, los caracteres (comandos) que ingresan por el modulo Bluetooth.

Con este programa podremos analizar lo que estamos leyendo del modulo Bluetooth, cada vez que necesitemos comprender. Luego podremos programar el que hacer ante tal o cual comando (**Ver Apéndice B - Librería SoftwareSerial**). El circuito es el mismo para ambos programas.

(Programa "050_Bluetooth_02_Led_01")

```
1   #include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pins
-   //Configuración Pines Adicionales  RXD,  TDX (En Arduino).
2   SoftwareSerial  BT(10,11);  //10- RX, 11- TX.
-
3   char Comando;
-
4   void setup( ){
5     BT.begin(9600);      //Velocidad del puerto del módulo Bluetooth
6     Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
7   }
-
8   void loop( ){
9     if(BT.available( )){  // Si hay algo pendiente de ser leído en Modulo Bluetooth
10      Comando = BT.read( );  // Leo el modulo Bluetooth
11      Serial.print("Comando Recibido: ");
12      Serial.println(Comando);
13      // Aca reconocer comandos y
14      // hacer según corresponda
15    }
16  } // Fin del loop( )
```

Este programa funcionará con cualquiera de los modelos de Modulo Bluetooth que usamos con Arduino: HC-05 o HC-06

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

Programa Alternativo - Ver en **Apéndice B - Librería SoftwareSerial - Función** readBytes

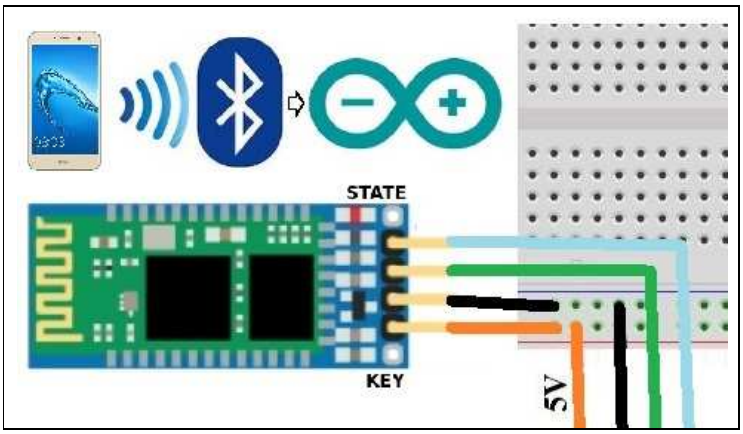
```
1   #include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones
-
2   SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11
-
3   byte bytesLeidos;
4   char buffer[100];
-
5   void setup( ){
6      Serial.begin(9600);
7      PuertoUno.begin(9600);   // Inicia PuertoUno  (puertos software)
8   }
-
9   void loop( ){
10     bytesLeidos = PuertoUno.readBytes ( buffer,  sizeof(buffer));
11     Serial.print( "Leídos: ");
12     Serial.print(bytesLeidos);
13     Serial.print( " - ");
14     Serial.println( buffer );
15  }
```

Estos programas fueron probados en el modelo Arduino Mega, por lo tanto, si quieres que funcionen en otro modelo, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino.**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).

Para Seleccionar e instalar aplicación que permita enviar comando desde el celular, Ver **Apéndice A - Comandos Bluetooth desde un Celular**



Los cables deberán ser conectados: Abajo, de Izquierda a Derecha, Cable Naranja: a 5v de Arduino. Cable Negro a GND. Cable Verde: Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Cable Celeste, del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). **Si te queda alguna duda como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**

Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

3) Enviar un número por Bluetooth y Arduino hará parpadear un Led conectado a la placa, el número de veces que indique el número enviado (entre 1 y 9).

La única diferencia con el primer ejemplo, es que gracias a la librería **SoftwareSerial** nos podemos comunicar con la PC y con el modulo Bluetooth simultáneamente. (**Ver Apéndice B - Librería SoftwareSerial**).

(Programa "050_Bluetooth_02_Led_02")

```
1   #include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pins
-   //Configuración Pines Adicionales  RXD,  TDX (En Arduino).
2   SoftwareSerial  BT(10,11);  //10- RX, 11- TX.
-
3   const int PinLed = 12; // Pin de control para el LED
4   int Comando, i;
-
5   void setup( ){
6    BT.begin(9600); //Velocidad del puerto del módulo Bluetooth
7    Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
8    pinMode(PinLed, OUTPUT);
9   }
10  void loop( ){
11   if(BT.available( )) {  // Si queda algo pendiente de ser leído en Modulo Bluetooth
12     Comando = BT.read( );  // Leo el modulo Bluetooth
13     if(Comando >= '1' && Comando <= '9'){ //si el Comando esta entre '1' y '9'
14       Comando = Comando - '0'; // Restamos el valor '0' para obtener el número enviado
15       Serial.println(Comando);  // Envío al monitor puerto serie, lo leído en el modulo Bluetooth
16       for( i=0; i< Comando; i++){
17         digitalWrite(PinLed, HIGH);
18         delay(500);
19         digitalWrite(PinLed, LOW);
20         delay(500);
21       }
22     } else { // Si lo leído en modulo Bluetooth NO ES un numero entre 1 y 9
23       Serial.print(Comando);
24       Serial.println(" - Comando Invalido");
25     }
26   }
27  }
```

Este programa funcionará con cualquiera de los modelos de Modulo Bluetooth que se conecten con Arduino: HC-05 o HC-06

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

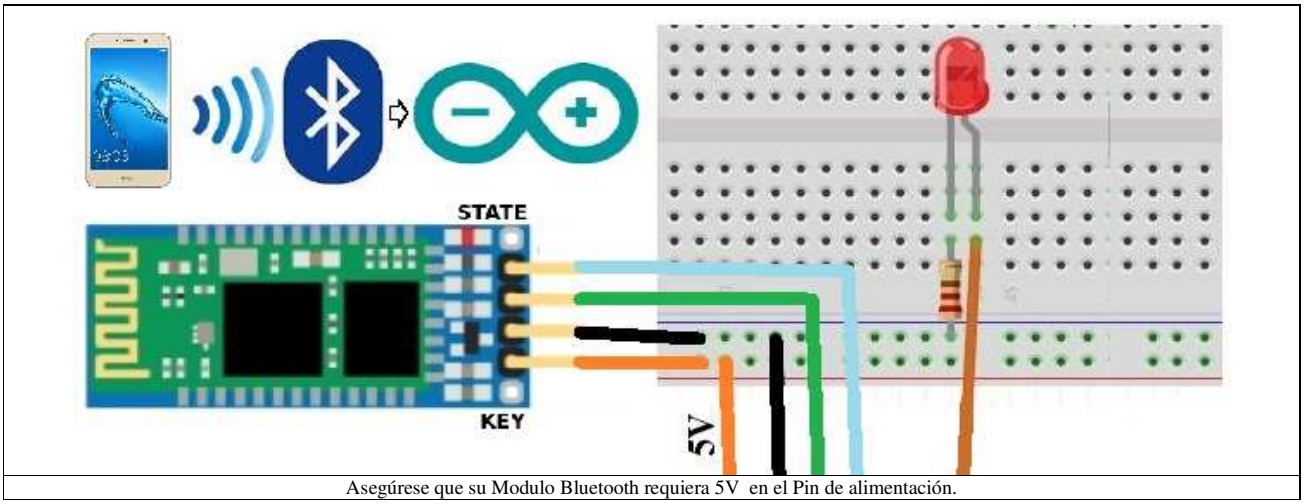
Estos programas fueron probados en el modelo Arduino Mega, por lo tanto, si quieres que funcionen en otro modelo, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Led - 1 Resistencia de 470Ω. - 2 Cables Macho/Macho - 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).

Para Seleccionar e instalar aplicación que permita enviar comando desde el celular, Ver **Apéndice A - Comandos Bluetooth desde un Celular**

Los cables deberán ser conectados: Abajo, de Izquierda a Derecha, Cable Naranja: a 5v de Arduino. Cable Negro a GND. Cable Verde: Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Cable Celeste, del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). Cable Marrón (Ultimo de la derecha), se debe conectar al pin que configuramos para el Led (Línea 01 del programa). **Si te queda alguna duda como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**



Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

Para saber más sobre reconocimiento de texto y las funciones que se usan, debe consultar "Anexo_01_Procesamiento_de_Texto".

4) Comandos: Prender y/o apagar un Led, enviando el comando através de Bluetooth: 1 Prende y 0 Apaga.

Esta forma de trabajar permitirá reconocer infinidad de comando y realizar acciones asociadas a los comandos, manteniendo comunicación con el monitor del puerto serie simultáneamente.

(Programa "050_Bluetooth_02_Led_02")

```
1  #include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pines
-  //Configuración Pines Adicionales RXD, TDX (En Arduino).
2  SoftwareSerial BT(10,11);  //10- RX, 11- TX.
3
3  const int PinLed = 12; // Pin de control para el LED
4  char comando;  // Comando recibido desde Modulo Bluetooth
5
5  void setup( ){
6    BT.begin(9600); //Velocidad del puerto del módulo Bluetooth
7    Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
8    pinMode(PinLed, OUTPUT);
```

Este programa funcionará con cualquiera de los modelos de Modulo Bluetooth que se conecten con Arduino: HC-05 o HC-06

```
 9  }
10  void loop( ){
11   if(BT.available( )){  // Si queda algo pendiente de ser leído en Modulo Bluetooth
12      comando = BT.read( );  // Leo el modulo Bluetooth
13      switch(comando){
14      case 'a':    // Esto es por la dudas no reconozca bien  comando por voz
15      case 'A':    // Esto es por la dudas no reconozca bien  comando por voz
16      case '0':{   // Esto es por la dudas no reconozca bien  comando por voz
17          digitalWrite(PinLed,LOW);
18          Serial.println("Led Apagado");
19        }break;
20      case 'p':    // Esto es por la dudas no reconozca bien  comando por voz
21      case 'P':    // Esto es por la dudas no reconozca bien  comando por voz
22      case '1': { // Esto es por la dudas no reconozca bien  comando por voz
23          digitalWrite(PinLed,HIGH);
24          Serial.println("Led Prendido");
25         }break;
26      default:
          Serial.println("Comando Invalido. Introduzca 0 o 1.");
        } // Fin del switch
    } // Fin del if
  } // Fin del loop
```

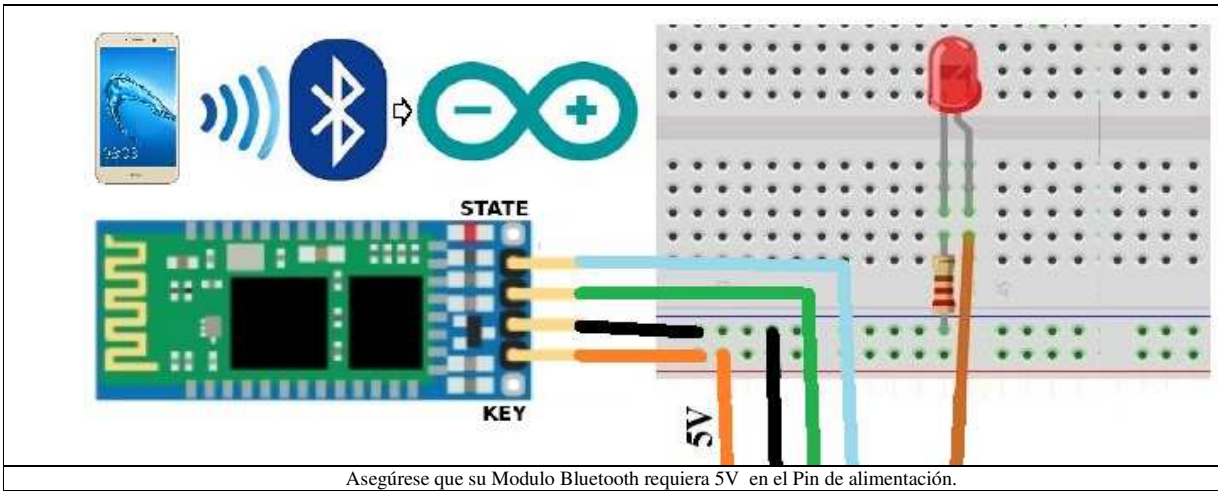
Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

Estos programas fueron probados en el modelo Arduino Mega, por lo tanto, si quieres que funcionen en otro modelo, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino.**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Led - 1 Resistencia de 470Ω. - 2 Cables Macho/Macho - 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).

Los cables deberán ser conectados: Abajo, de Izquierda a Derecha, Cable Naranja: a 5v de Arduino. Cable Negro a GND. Cable Verde: Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Cable Celeste, del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). Cable Marrón (Ultimo de la derecha), se debe conectar al pin que configuramos para el Led (Línea 01 del programa). **Si te queda alguna duda como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**



Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

5) Mostrar en el monitor del puerto serie, las cadenas de caracteres o palabras (comandos) que ingresan por el modulo Bluetooth.

Con este programa podremos analizar lo que estamos leyendo del modulo Bluetooth, cada vez que necesitemos comprender. Luego podremos programar el que hacer ante tal o cual comando (**Ver Apéndice B - Librería SoftwareSerial**). El circuito es el mismo para ambos programas.

(Programa "050_Bluetooth_03_Led_01")

```
1  #include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones
2  SoftwareSerial BT(10,11); //TX = pin digital 10, RX = pin digital 11
3  char cadena[255];  //Creamos un array de caracteres de 256 posiciones
4  char comando;
5  int i = 0; //Tamaño actual del array
6  void setup( ) {
7    BT.begin(9600);
8    Serial.begin(9600);
9    Serial.println("Los Comandos Bluetooth deben terminar con un Punto");
10 }
11 void loop( ){
12   if(BT.available( )){ //Cuando haya comandos disponibles
13     comando = BT.read( ); //Guarda los comandos carácter a carácter en la variable "comando"
14     cadena[i] = comando; //Vamos colocando cada carácter recibido en el array "cadena"
15     i++;
16     if(comando=='.'){ // El punto marca el final del comando
17       Serial.print("Comando Recibido: "); //Visualizamos el comando recibido en el Monitor Serial
18       Serial.println(cadena);              //Visualizamos el comando recibido en el Monitor Serial
19       // Aca reconocer los comandos y hacer según corresponda
20      clean( ); //Ejecutamos la función clean( ) para limpiar el array
21      }
22   }
23 }
24 void clean( ){ //Limpia el array
25   int cl;
26   for ( cl=0; cl<=i; cl++){
27      cadena[cl]=0;
28   }
29   i=0;
30 }
```

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

Programa Alternativo (050_Bluetooth_03) - Ver **Apéndice B - Librería SoftwareSerial - Función** readBytes

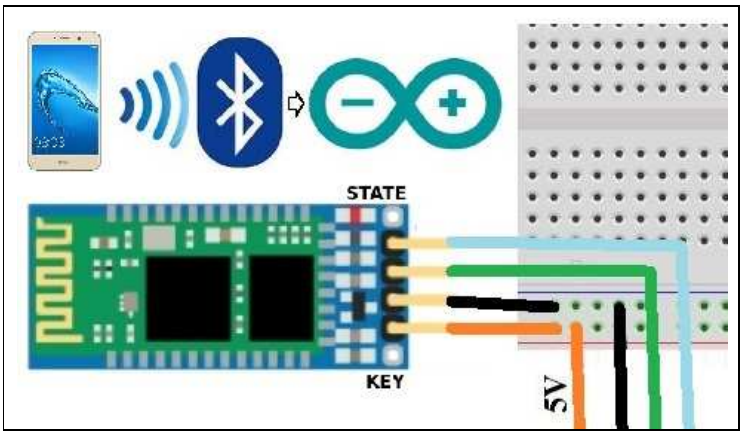
```
1  #include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones
2  SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11
3  byte bytesLeidos;
4  char buffer[100];
5  void setup( ){
6     Serial.begin(9600);
7     PuertoUno.begin(9600);   // Inicia PuertoUno  (puertos software)
```

```
8   }
-
9   void loop( ){
10      bytesLeidos = PuertoUno.readBytes ( buffer,  sizeof(buffer));
11      Serial.print( "Leídos: ");
12      Serial.print(bytesLeidos);
13      Serial.print( " - ");
14      Serial.println( buffer );
15  }
```

Estos programas fueron probados en el modelo Arduino Mega, por lo tanto, si quieres que funcionen en otro modelo, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).



Los cables deberán ser conectados: Abajo, de Izquierda a Derecha, Cable Naranja: a 5v de Arduino. Cable Negro a GND. Cable Verde: Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Cable Celeste, del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). **Si te queda alguna duda como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**

Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

Para saber más sobre reconocimiento de texto y las funciones que se usan, debe consultar "Anexo_01_Procesamiento_de_Texto".

6) Mostrar en el monitor del puerto serie, las cadenas de caracteres o palabras (comandos) que ingresan por el modulo Bluetooth. Incorporar una función de ayuda al usuario.

Con este programa podremos analizar lo que estamos leyendo del modulo Bluetooth, cada vez que necesitemos comprender. Luego podremos programar el que hacer ante tal o cual comando (**Ver Apéndice B - Librería SoftwareSerial**). El circuito es el mismo para ambos programas.

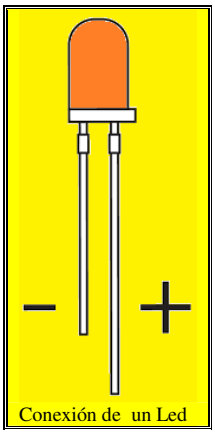
(Programa "050_Bluetooth_03_Led_02")

```
1   #include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones
-
2   SoftwareSerial BT(10,11); //TX = pin digital 10, RX = pin digital 11
3   int verde     = 6;
4   int amarillo = 7;
5   int rojo      = 8;
6   char cadena[100]; //Creamos un array de caracteres de 100 posiciones (Tamaño máximo)
```

```
-
7   char comando;      // Variable que usaremos para leer caractér a caractér el comando
8   int i = 0; //Tamaño actual del array - Donde debo guardar el caractér leído
-
9   void setup( ){
10   BT.begin(9600);
11   Serial.begin(9600);
-
12   pinMode(verde,OUTPUT);
13   pinMode(amarillo,OUTPUT);
14   pinMode(rojo,OUTPUT);
15  }
-
16  void loop( ){
17    if(BT.available( )){   // Detecta que hay caracteres que leer - Hay comandos disponibles
-
18        comando = BT.read( ); //Lee y Guarda carácter a carácter en variable "comando"
19        cadena[i] = comando; //Vamos colocando cada carácter recibido en el array "cadena"
20        i++;
-
21        if(comando == '.'){ // El punto da por terminado el comando.
-
22           Serial.println(cadena); //Visualizamos el comando recibido en el Monitor Serial
-
23           //Led Verde
24           if(strstr(cadena,"verde +")!=0){ //Buscar subcadena dentro del string.
25              digitalWrite(verde,HIGH);
26           }
27           if(strstr(cadena,"verde -")!=0){ //Buscar subcadena dentro del string.
28               digitalWrite(verde,LOW);
29           }
-
30           //Led Amarillo
31           if(strstr(cadena,"amarillo +")!=0){ //Buscar subcadena dentro del string.
32               digitalWrite(amarillo,HIGH);
33           }
34           if(strstr(cadena,"amarillo -")!=0){ //Buscar subcadena dentro del string.
35               digitalWrite(amarillo,LOW);
36           }
-
37           //Led Rojo
38           if(strstr(cadena,"rojo +")!=0){ //Buscar subcadena dentro del string.
39               digitalWrite(rojo,HIGH);
40           }
41           if(strstr(cadena,"rojo -")!=0){ //Buscar subcadena dentro del string.
42               digitalWrite(rojo,LOW);
43           }
-
44           // Prende Todos los Leds
45           if(strstr(cadena,"todo +")!=0){ //Buscar subcadena dentro del string.
46               digitalWrite(verde,HIGH);
47               digitalWrite(amarillo,HIGH);
48               digitalWrite(rojo,HIGH);
49           }
-
```

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

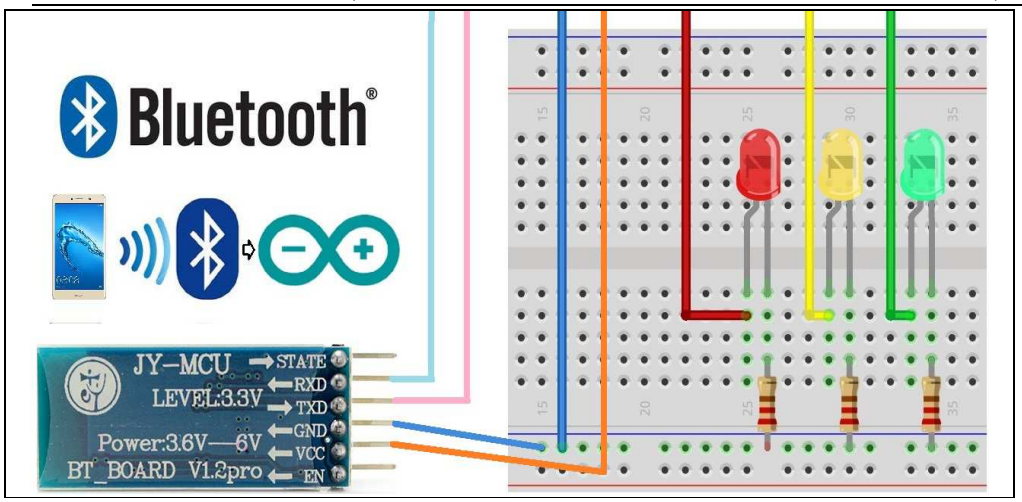
```
50          // Apaga Todos los Leds
51          if(strstr(cadena,"todo -")!=0){ //Buscar subcadena dentro del string.
52              digitalWrite(verde,LOW);
53              digitalWrite(amarillo,LOW);
54              digitalWrite(rojo,LOW);
55          }
-
56          //  Ayuda para el Usuario
57          if(strstr(cadena,"?")!=0){
58              ayuda( );
59          }
60          clean( ); //Ejecutamos la función clean( ) para limpiar el array
61      }// Final if que encuentra el punto
-
62      if(strstr(cadena,"?")!=0){
63        ayuda( );  //  Ayuda para el Usuario
64        clean( );   //Ejecutamos la función clean( ) para limpiar el array
65      }
66    } // Final del if donde detecta que hay caracteres que leer
67  }  // Fin del loop
-
68  void clean( ){ //Limpia el array
69    int cl;
70    for ( cl=0; cl<=i; cl++){
71      cadena[cl]=0;
72    }
73    i=0;
74  } // Final de la Función que limpia
-
75  void ayuda( ){
76    Serial.println("=========================================");
77    Serial.println("          Ayuda para el USAUARIO");
78    Serial.println("a- Los comandos deben terminar con un Punto");
79    Serial.println("b- verde +.  ====> Prende led Verde");
80    Serial.println("c- verde -.  ====> Apaga  led Verde");
81    Serial.println("d- amarillo +.  => Prende led Verde");
82    Serial.println("e- amarillo -.  => Apaga  led Verde");
83    Serial.println("f- rojo +.  =====> Prende led Verde");
84    Serial.println("g- rojo -.  =====> Apaga  led Verde");
85    Serial.println("h- todo +.  =====> Prende TODOS los Leds");
86    Serial.println("i- todo -.  =====> Apaga  TODOS los Leds");
87    Serial.println("=========================================");
88  }
```



Estos programas fueron probados en el modelo Arduino Mega, por lo tanto, si quieres que funcionen en otro modelo, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino**

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 3 Led - 3 Resistencias de 470Ω. - 4 Cables Macho/Macho - 4 Cables Macho/Hembra - Modulo Bluetooth Modelo HC-05 o Modelo HC-06 - Placa Arduino y 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular).



Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

Los cables deberán ser conectados: Arriba, de Izquierda a Derecha, Cable Celeste: del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). Cable Rosa: Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Cable Azul: conectar a GND en Arduino. Cable Naranja: conectar a 5V de placa Arduino. Cable Rojo Oscuro: Conectar al Pin destinado para controlar el Led Rojo. Cable Amarillo: Conectar al Pin destinado para controlar el Led Amarillo. Cable Verde: Conectar al Pin destinado para controlar el Led Verde. **Si te queda alguna duda como se conectan los pines RX y TX, simplemente debes conectarlos en forma cruzada (RX con TX y TX con RX).**

Para Seleccionar e instalar aplicación que permita enviar comando desde el celular, Ver **Apéndice A - Comandos Bluetooth desde un Celular**

7) Controlar dos Servo Motores através de comandos recibidos por Bluertooth.

Enviar comandos desde Celular por Bluetooth, indicar el Angulo en el que deben posicionarse los Servo Motores. Los dos Servos se mueven en forma idéntica. Los valores (ángulos) permitidos son todos los mayores o iguales a cero y menores o iguales a 180.

(Programa "050_Bluetooth_04_ServoMotor_01")

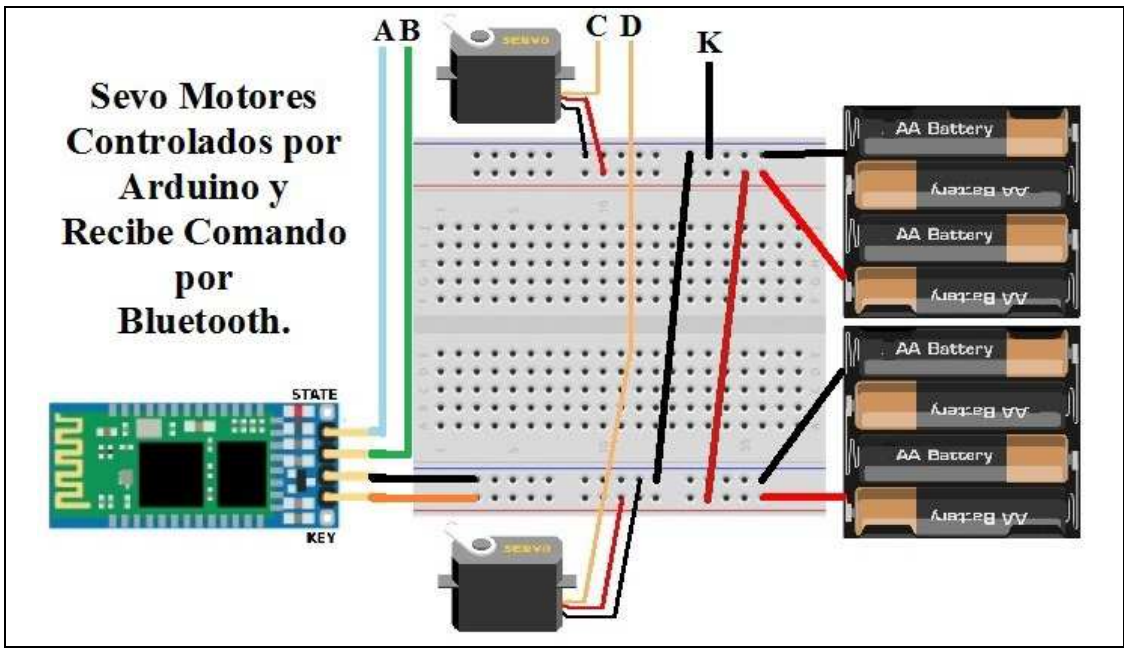
```
1  #include <SoftwareSerial.h>   // Ver Apéndice B - Librería SoftwareSerial.
2  #include <Servo.h>            //
-
3  Servo MyServo_01,     // Instancia objeto (variable) del tipo Servo
4        MyServo_02;
-
5  int   PinServo_01 = 8,   // Pin desde donde controlare el Servo
6        PinServo_02 = 9,
7        PosServo;        // Comando-Posición a la que se moverá el servo
-
8  int Tx = 10,    // Pin usado para Transmisión de datos
9      Rx = 11;    // Pin usado para la Recepción de datos
-
10 SoftwareSerial Bluetooth(Tx, Rx); //Configuración Pines Adicionales (En Arduino).
-
11 void setup(){
12   MyServo_01.attach(PinServo_01); // Configuración Pin de Control del Servo 01
13   MyServo_02.attach(PinServo_02); // Configuración Pin de Control del Servo 02
14   Serial.begin(9600);          // Configuración Velocidad comunicación con Monitor Serie
```

```
15    Bluetooth.begin(9600);      // Configuración Velocidad comunicación con Módulo Bluetooth
16  }
-
17  void loop( ){
18    if(Bluetooth.available( ) > 0 ){   // Pregunto si el Modulo Bluetooth tiene datos para leer
19      PosServo = Bluetooth.read( );  // Leo Modulo Bluetooth
20      Serial.println(PosServo);          // Muestro en Monitor el Valor (ángulo leído)
21      MyServo_01.write(PosServo);      // Muevo (posiciono) el servo 01 al ángulo Indicado
22      MyServo_02.write(PosServo);      // Muevo (posiciono) el servo 02 al ángulo Indicado
23    }
24  }
```

Recordar que si hay más de dos Servos, Arduino no debe alimentarlos, por lo tanto usará Alimentación Externa adicional.

CIRCUITO PARA NUESTRO PROYECTO

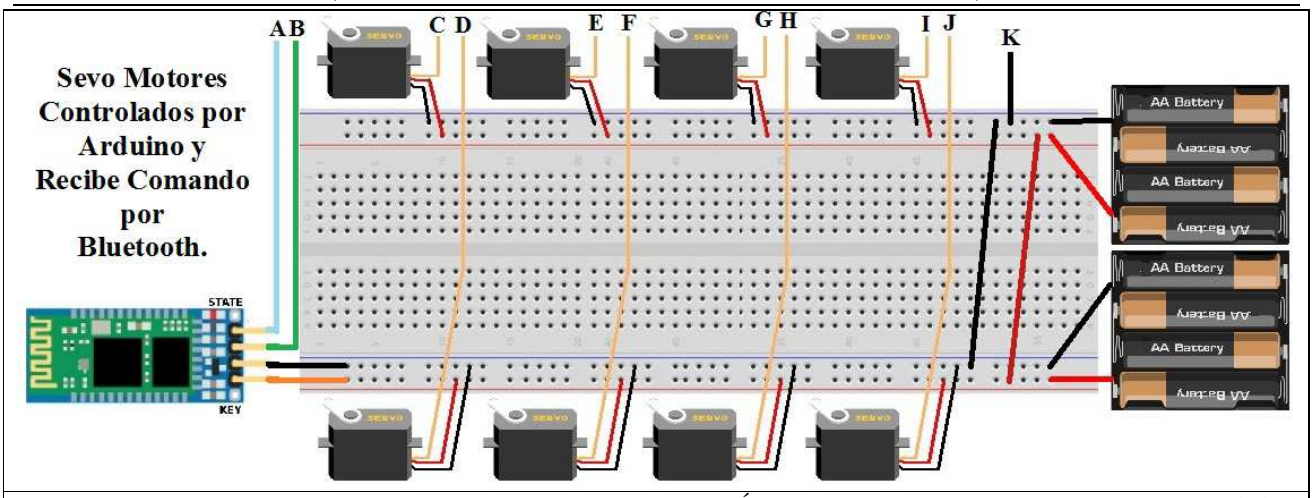
Lista de Materiales: 1 o 2 Porta Pilas, cada uno para 4 pilas comunes AA (1,5V) - 2 Servo Motores **MG90 o MG90S** – Cables Macho/Macho y cables Hembra/Macho - 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular). 1 Protoboard - 1 Placa Arduino.



Los cables deberán ser conectados: Arriba, de Izquierda a Derecha, Cable A (Celeste), del RX del Modulo Bluetooth a TX de Arduino (PIN 10 adicional TX en Arduino). Cable B (Verde), Corresponde en Modulo Bluetooth a TX y se debe conectar al RX de Arduino (PIN 11 adicional RX en Arduino). Los Cables C y D, son los que usaremos para controlar los Servos, y deberán conectarse a los pines de Arduino 8 y 9 respectivamente. Por ultimo el cable K (Negro), es que conectaremos al GND o masa común del dispositivo. Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.

En este ejemplo, los servos y modulo Bluetooth, son alimentados por el Porta Pilas, y es importante entender, que el circuito con dos servos podríamos ampliarlo de 2 a 4Servos (Para un Brazo Robótico), u 8 servos, o 16 Servos (Una Araña Robótica), o 24 Servos, solo deberemos tener Pines Digitales disponibles en nuestra Placa Arduino, para controlar cada uno de los servos (Uno por cada servo).

A modo de ejemplo les dejo el mismo circuito, pero para 8 Servo Motores.



El control de los servos también podría haberse resuelto usando el MÓDULO CONTROLADOR DE SERVOS PCA9685 explicado en guías anteriores y entonces, solo usaríamos dos pines de Arduino.

8) Controlar un Vehiculo (dos motores) por medio de comandos recibidos por Bluetooth.

Este programa permite recibir comando y ejecutar las funciones básicas en un automóvil (A- adelante, R- atrás, D- derecha, I- izquierda, S- alto). El código es muy similar al que usamos para controlar un automóvil desde la PC.

(Programa "050_Bluetooth_05_Motores_DC_Driver_L298N_01_Dos_Motores")

```
1  #include <LEANTEC_ControlMotor.h> //Incluimos la librería control de motores
2  #include <SoftwareSerial.h>              // Ver Apéndice B - Librería SoftwareSerial.
-
3  int PinIN1 = 3,      //El pin 7 de Arduino se conecta con el pin In1 del L298N
4      PinIN2 = 5,      //El pin 4 de Arduino se conecta con el pin In2 del L298N
5      PWM_Izquierdo_ENA = A0; //Pin Analógico Arduino, conecta pin EnB del L298N (Control de Velocidad)
-
6  int PinIN3 = 7,   //Pin Digital que conecta Arduino con el pin In3 del L298N
7      PinIN4 = 9,    //Pin Digital que conecta Arduino con el pin In4 del L298N
8      PWM_Derecho_ENB = A1; //Pin Analógico Arduino, conecta pin EnA del L298N (Control de Velocidad)
-
9  int Tx = 10, // Pin usado para Transmisión de datos
10     Rx = 11; // Pin usado para la Recepción de datos
11 SoftwareSerial Bluetooth(Tx, Rx); //Configuración Pines Adicionales (En Arduino).
-
12 int Sentido = 150, // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
13       Se_A = 0,
14       La_A = 0,
15       Lateral = 1;  // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
-
16 char dato;
-
17 // Configuramos los pines que vamos a usar para controlar los motores
18 ControlMotor control( PinIN1, PinIN2, PinIN3, PinIN4, PWM_Izquierdo_ENA, PWM_Derecho_ENB );
-
19 void setup(){
20   Serial.begin(9600);          // Configuración Velocidad comunicación con Puerto Serie
21   Bluetooth.begin(9600);     // Configuración Velocidad comunicación con Módulo Bluetooth
22   while (!Serial) {
23    ; // Esperamos que el puerto serie este abierto.
24   }
```

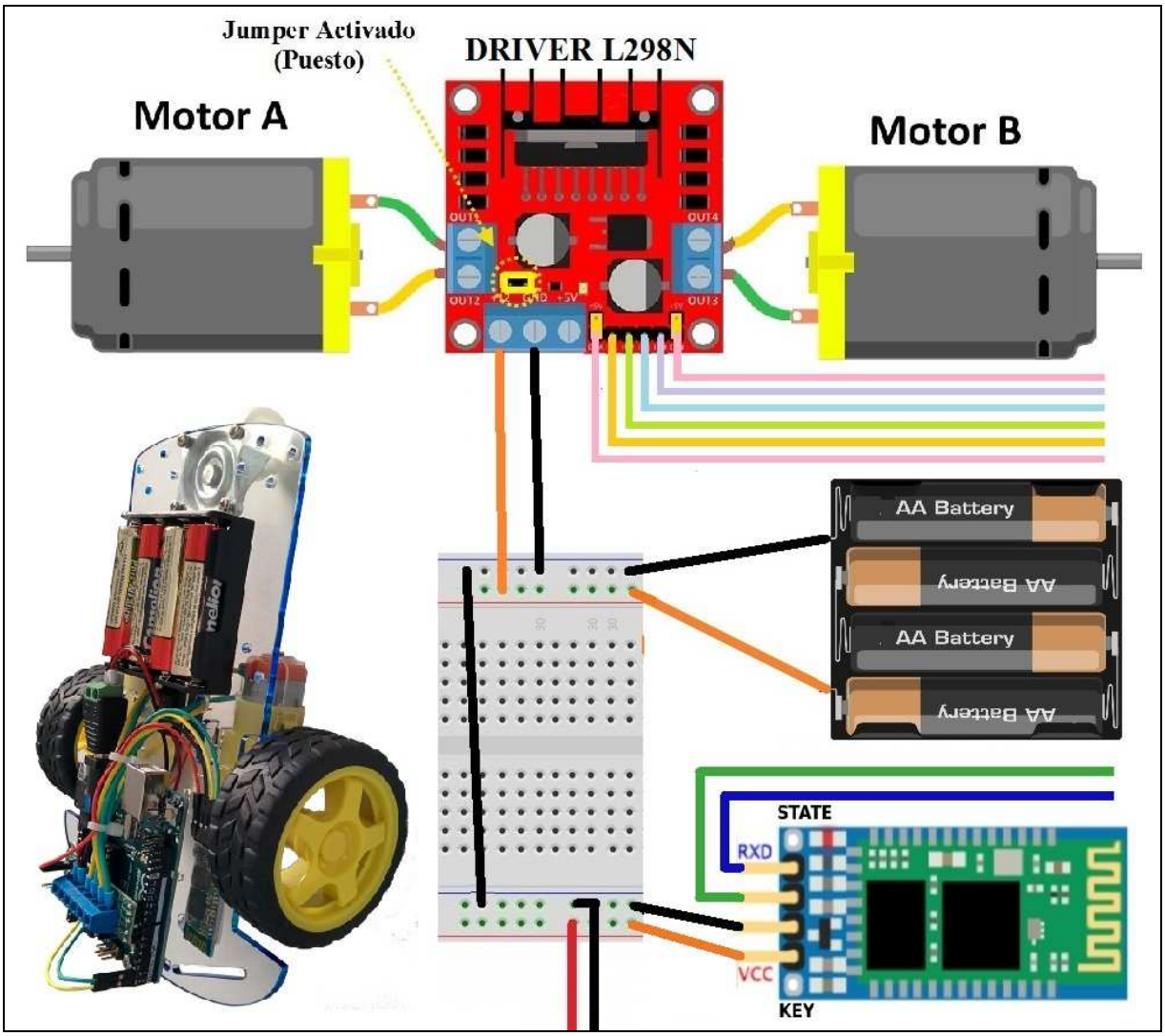
```
25  }
-
26  void loop ( ){
27    if(Bluetooth.available()> 0 ){
28      dato = Bluetooth.read();
29      Serial.print("Comando Recibido: ");
30      Serial.println(dato);
31      switch (dato){ //Seleccionamos el caso, dependiendo del caractér recibido.
32          case 'A':
33          case 'a': {  // Adelante
34              Sentido = 150;  // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
35              Lateral = 1;       // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
36          } break;
37          case 'R':
38          case 'r': {  // Atrás
39              Sentido = -150; // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
40              Lateral = 1;       // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
41          } break;
42          case 'd':
43          case 'D': {  // Doblar a la Derecha
44              Sentido = 150;  // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
45              Lateral = 100;   // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
46          } break;
47          case 'i':
48          case 'I': {  // Doblar a la Izquierda
49              Sentido = 150;  // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
50              Lateral = -100;  // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
51          } break;
52          case 's':
53          case 'S': {  // Detener Motores
54              Sentido = 0;  // Valores Positivos Adelantes - Negativos Atrás - 0 (Cero) detiene
55              Lateral = 1;   // 1 Derecho - Mas de uno dobla a la derecha - Negativos dobla a la Izquierda
56          } break;
57          default: {
58              int d = dato;
59              Serial.print("Comando Incorrecto: ");
60              Serial.print(d);
61              Serial.print(" * ");
62              Serial.println(dato);
63          } break;
64      } // Finaliza switch
65    } // Finaliza if - Donde se controla si se ha leído algo
-
66    if( Se_A != Sentido || La_A != Lateral){ // verifico si hay algo nuevo (distinto de lo Anterior)
67      control.Motor( Sentido, Lateral );
68      Se_A = Sentido;
69      La_A = Lateral,
70      Serial.print("Direcccion: ");
71      Serial.print(Sentido);
72      Serial.print(" * ");
73      Serial.println(Lateral);
74      delay(50);
75    }
76  }
```

Los Módulos Bluetooth, suelen venir configurado por defecto, para trabajar a 9600 baudios. El nombre del dispositivo generalmente es HC-05 o HC-06 y la contraseña 1234.

Este programa podría ampliarse al manejo de 4 motores muy fácilmente, solo hay que controlar dos Drivers y tener en cuanta que debe hacer cada rueda ante cada acción.

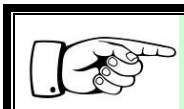
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Porta Pilas para 4 pilas comunes AA (1,5V) - 2 **Motores DC 6v con Caja reductora y Rueda de Goma** - 1 Driver **L298N** - Modulo Bluetooth (HC05 o HC06) - 1 Cable USB - Celular con aplicación instalada y configurada para enviar comandos remotos por Bluetooth (Ver Apéndice A - Comandos Bluetooth desde un Celular) - Cables de conexión Hembra/Macho y Macho/Macho - 1 Protoboard - 1 Placa Arduino.



Los cables deberán ser conectados: Primero veremos los cables de la derecha y los analizaremos ordenadamente desde arriba hacia abajo. Comenzamos por el Cable Rosado, que conectaremos al Pin Analógico que se configure en el Programa ENB (en este ejemplo A1). Cable Morado, al Pin Digital que se haya configurado en el programa a IN4 (en este caso 9). Cable Celeste, al Pin Digital que se haya configurado en el programa a IN3 (en este caso 7). Cable Verde, al Pin Digital que se haya configurado en el programa a IN2 (en este caso 5). Cable Amarillo, al Pin Digital que se haya configurado en el programa a IN1 (en este caso 3). Cable Rosado, deberemos conectarlo al Pin Analógico que se configure en el Programa ENA (en este ejemplo A0). Cable Verde (módulo Bluetooth), al pin digital que redefinimos para TX (10 para este ejemplo). Cable Azul (módulo Bluetooth), lo conectaremos al pin digital que redefinimos para usar RX (11 para este ejemplo).
A continuación los cables que visualizamos en la parte inferior de la imagen, de izquierda a derecha, Cable Rojo a 5V de Arduino y Cable Negro a GND de Arduino.

Importante: En este ejemplo, los Motores y Modulo L298N, son alimentados por el Porta Pilas, y el módulo Bluetooth es alimentado por Arduino, adicionalmente en este ejemplo, dejamos desconectados los pines State y Key (también llamado ES). - Asegúrese que su Modulo Bluetooth requiera 5V en el Pin de alimentación.



Si el o los motores no giran en el sentido que esperaba, solo invierta los cables en el conector del motor A o del motor B según sea el caso - Es decir cambie la polaridad, y funcionará al revés.

Configuracion Modulo Bluetooth - Comandos AT

(Modelos HC-05 y HC-06)

Los comando AT son los usados para configurar el modulo Bluetooth. Es muy importante saber, que cada modulo tiene un conjunto de comandos, y para que sean reconocidos, el modulo debe ingresar al **MODO CONFIGURACIÓN**. A continuación veremos como se configura cada uno de estos módulos.

RECORDAR SIEMPRE: Mientras que el HC-06 entra en modo de programación en cuanto lo enciendes y no haya nadie conectado por Bluetooth, el HC-05 es ligeramente diferente para iniciar en modo configuración, ya que requiere que presionemos el Botón o que el pin KEY (también llamado **EN**), esté en HIGH cuando encendemos el modulo (Ver Modos de configuración).

Tenemos tres maneras distintas para configurar un modulo Bluetooth

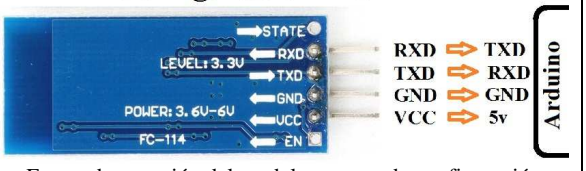
Con cualquiera de los dos módulos (HC-05 y HC-06), podremos usar cualquiera de las tres formas que a continuación se describen, sin embargo en esta guía usaremos la configuración indirecta.

1) **Configuración Directa:** Conectamos el cable USB directamente con el modulo Bluetooth. Para esto necesitaremos un conversor USB-Serial. Este método no lo usaremos por ahora.
2) **Configuración Automática:** Cargamos un programa en la placa Arduino que realice la configuración en forma automática cuando arranca y antes de que un dispositivo se conecte. Esta forma la usaremos más adelante, cuando usemos archivos de Configuración inicial (Archivo.ini)
3) **Configuración Indirecta:** Cargamos un programa que reciba los comandos desde la PC vía cable USB, y luego se los re-envíe al Modulo Bluetooth, y todo esto, sin que un dispositivo se conecte al modulo a través de Bluetooth. **Este es el método que usaremos ahora**.

HC-06 - Comandos AT - Configuración del Módulo

Para configurar el Módulo HC-06 en forma automática o indirecta, no requerimos de una conexión especial, ya que usaremos la misma que se ha usado para realizar los ejemplos anteriores. Igualmente reitero el diagrama de conexión con nuestra placa Arduino.

Configuración HC-06



Forma de conexión del modulo para modo configuración

En esta guía usaremos la forma de configuración Indirecta: Cargamos un programa que reciba los comandos desde la PC vía cable USB, y luego se los re-envíe al Modulo. Comencemos.

Configuración por defecto o de Fábrica. Módulo HC-06	➢ Nombre del dispositivo por defeco: "**HC-06**" - El nombre depende del fabricante. ➢ Código de emparejamiento por defecto (password): 1234 ➢ La velocidad por defecto (baud rate): 9600 ➢ Paridad (por defecto) configurable en versiones V1.5 o sup.: Sin Paridad.

Entrar Modo Configuración	➢ El Módulo HC-06 entra en modo de programación Apenas se lo enciende y no se ha establecido una conexión Bluetooth con ningún otro dispositivo. ➢ EL LED del módulo está parpadeando (frecuencia de parpadeo del LED es de 102ms) ➢ En este modo es cuando se debe enviar los comandos AT en caso se quiera configurar, si se envían comandos diferente a los reconocidos por el Módulo HC-06, serán ignorados. **IMPORTANTE:** Cuando establece una conexión Bluetooth con algún dispositivo, el HC-06 no puede interpretar los comandos AT, y el LED permanece prendido sin parpadear.

Configuración Indirecta del modulo Bluetooth.

Es indirecta porque necesitamos de un programa en la Placa Arduino que lea lo que escribimos por monitor serie y se lo envíe al modulo. Para que nuestro modulo permita ser configurado, no debe haber dispositivos conectados por Bluetooth con el modulo

(Programa "050_Bluetooth_00_Configuracion_Indirecta_HC-06")

```
1   #include <SoftwareSerial.h>  // Incluimos la librería  SoftwareSerial
-
2   SoftwareSerial BT(10,11);    //TX = pin digital 10, RX = pin digital 11
-
3   void setup( ){
4     BT.begin(9600);       // Inicializamos el puerto serie BT que hemos creado
5     Serial.begin(9600);   // Inicializamos  el puerto serie
6   }
-
7   void loop( ){
8     if(BT.available( )){    // Si llega un dato por el puerto BT se envía al monitor serial
9       Serial.write(BT.read( )); // Lee las respuestas dadas Modulo y envía al Monitor
10    }
11    if(Serial.available( )){  // Si llega un dato por el monitor serial se envía al puerto BT
12      BT.write(Serial.read( ));
13    }
14  }
```

Este programa fue probado en el modelo Arduino Mega, conectado a una PC, por medio de un cable USB. Si quieres que funcionen en otro modelo de Arduino, podrías tener que cambiar la Configuración de Pines Adicionales **RXD, TDX según sea el modelo de tu Arduino.**

Una vez hayamos conectado el módulo y enviado este programa a la placa Arduino, abrimos el Monitor serial del IDE de Arduino, pero puedes usar cualquier otro monitor serial si lo deseas.
En la parte inferior debemos escoger "No hay ajuste de línea" y la velocidad "9600 baud" (la velocidad por defecto de nuestro HC-06). Echo esto, ya podemos empezar a enviar los comandos AT a nuestro módulo Bluetooth.

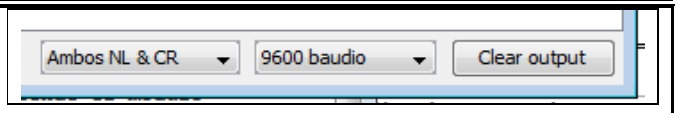
Hay que tener cuidado de introducir **TODAS LAS LETRAS DEL COMANDO en MAYUSCULAS**, ya que de lo contrario, no funcionarán.

1- Test de comunicación.

Una vez se ha ingresado al Modo Configuración, lo primero es comprobar si nuestro bluetooth responde a los comandos AT.
Enviar: AT
Recibe: OK

Si recibimos como respuesta un OK en el Monitor del Puerto Serie entonces podemos continuar, pero si **NO recibimos el OK**, o recibimos basura ilegible, debemos verificar que:

- ✓ Las conexiones estén correctas, **en el Monitor Puerto Serie** "NL&CR". (Ver abajo a la Derecha). Con esto enviaremos apropiadamente las líneas de comandos.



- ✓ Que en nuestro programa, la velocidad de comunicación con el modulo Bluetooth sea la correcta (Hay fabricantes que cambian esta velocidad). Si el punto anterior quedó descartados como posible falla, entonces recomiendo probar con **todas** las velocidades posibles:

Velocidades	
1 -------	1200
2 -------	2400
3 -------	4800
4 -------	9600
5 -------	19200
6 -------	38400
7 -------	57600
8 -------	115200

En el Programa la velocidad de comunicación con el Modulo Bluetooth la configuramos en la línea "**BT1.begin(9600);**", Es acá donde debemos cambiar y probar, hasta encontrar la velocidad con la que SI Funcione.

2- Obtener la versión del Firmware.

Este es uno de los comandos más fáciles de usar.
Enviar: AT+VERSION
Respuesta: OK<Versión>
Ejemplo: OKLinvor1.8

3- Cambiar nombre del dispositivo - módulo HC-06.

Por defecto nuestro módulo bluetooth se llama "**HC-06**" o "**Linvor**" esto se puede cambiar con el siguiente comando:
Enviar: AT+NAME<Nombre> Ejm: AT+NAMEModulo
Respuesta: OKsetname

El nombre puede ser de hasta 20 caracteres máximo

4- Cambiar Password o Código de Vinculación.

Por defecto viene con el código de vinculación (Pin) "1234", para cambiarlo hay que enviar el siguiente comando:
Enviar: AT+PIN<Pin>

Ejm: AT+PIN4321
Respuesta: OKsetPIN

Una vez hayas verificado el cambio de clave, anótala o cámbiala otra vez al valor original. Evitaras futuros problemas por olvidos.

5- Configurar la velocidad de comunicación.

La velocidad por defecto es de 9600 baudios, para cambiarlo se hace uso del siguiente comando:

Enviar: AT+BAUD<Numero>
Respuesta: OK<baudrate>
Donde <Numero> equivale a una velocidad de <baudrate>

Ejemplo:
Enviar: AT+BAUD3
Respuesta: OK4800

Número - baudrate	
1 -------	1200
2 -------	2400
3 -------	4800
4 -------	9600
5 -------	19200
6 -------	38400
7 -------	57600
8 -------	115200

Nota: Después de cambiar la velocidad, para continuar enviando comando AT, hay que hacerlo con la nueva velocidad, para eso, si se está usando un conversor USB serial tan solo hay que cambiar la velocidad en el monitor Serial (parte inferior); pero si está enviando los comandos AT a través de un programa desde una placa Arduino, es necesario volver a programar (Cambiando la velocidad) y cargar un nuevo programa.

6- Configuración de Paridad.

Solo configurable para versiones V1.5 o superiores - Recordar que **sin paridad** es el valor por defecto

Sin Paridad (valor por defecto)
Enviar: AT+PN
Respuesta: OK NONE

Paridad Impar
Enviar: AT+PO
Respuesta: OK ODD

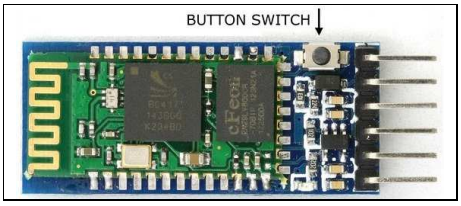
Paridad Par
Enviar: AT+PE
Respuesta: OK EVEN

Nota: Después de cambiar la Paridad en su modulo Bluetooth, para mantener la comunicación con los dispositivos, seguramente, deberá cambiar la configuración en los dispositivos con los que espera comunicarse

.

HC-05 - Comandos AT - Configuración del Módulo

Debido a que existen varios modelos (Fabricantes) y versiones distintas del módulo HC-05, podríamos encontrar algunas diferencias al momento de trabajar con ellos, pero generalmente son muy fáciles de superar, o directamente algunos Comandos que NO Funcionen en nuestro modulo, pero en otros si. El modelo que usaremos es el que se muestra en todas las imágenes, tiene un pulsador que nos servirá para entrar en Modo AT y configurarlo.



Recordar Siempre Que La PC, El Celular O La Tablet, Funcionan Como Maestros. Entonces, si queremos conectarnos con alguno de los dispositivos antes Mencionados, Necesitaremos El Modulo Configurado Como Esclavo. El Modulo Bluetooth Configurado Como Maestro, Nos Sirve Para Comunicarnos Con Otro Modulo Configurado Como Esclavo. EL modulo Bluetooth HC-05 viene configurado de fábrica como Esclavo, pero se puede cambiar para que trabaje como maestro.

Definamos que es, un Dispositivo Maestro y que es, un dispositivo Esclavo

Modulo Bluetooth como esclavo:
Cuando está configurado de esta forma, espera que un dispositivo Bluetooth maestro se quiera conectar con el (similar a un HC-06), generalmente se utiliza cuando se necesita comunicarse con una PC o Celular, pues estos se comportan como dispositivos maestros.

Modulo Bluetooth como Maestro:
En este modo, es el que debe iniciar la conexión. Un dispositivo maestro solo se puede conectarse con un dispositivo esclavo. Generalmente se utiliza este modo para comunicarse con otros módulos Bluetooth. Pero es necesario antes especificar con que dispositivo se tiene que comunicar.

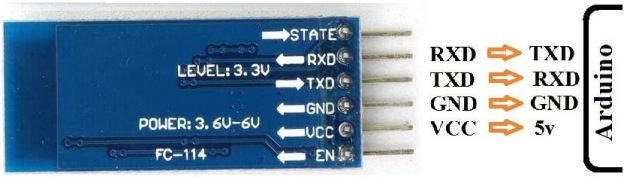
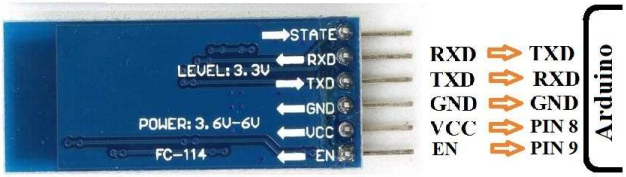
ALGO PARA RECORDAR - En el Módulo Bluetooth		
Pin	**Descripción**	**Analogía**
RX	Es el Pin usado para la Recepción de datos enviados por el otro dispositivo.	Sería nuestra **boca**
TX	Es el Pin usado para Transmisión de datos (Envío), al otro dispositivo.	Sería nuestro **oído**

-o-

Configuración por defecto (Fábrica). Módulo HC-05	➢ Modo o role: Esclavo ➢ Nombre por defeco: "**HC-05**". ➢ Código de emparejamiento por defecto: 1234 ➢ La velocidad por defecto (baud rate): 9600 - En modo configuración generalmente es otra velocidad, que deberemos buscar (Probando). ➢ Paridad: Sin Paridad.

Estados Posibles del Modo Bluetooth HC-05

El Módulo Bluetooth, dependiendo del momento, puede encontrarse en 5 estados distintos, los cuales se describen un poco mas abajo.
Cada estado, tienen asociado un formato de conexión (de cables entre modulo y Placa Arduino) y una forma particular de programas, para realizar las distintas acciones.

Conexión HC-05 - Formato 1	**Conexión HC-05 - Formato 2**
	
Este formato de conexión, permite entrar en Modo Configuración AT1 y AT 2 - Algunos Modelos o versiones del HC-05, NO permiten ingresar a este Modo de Configuración pero TODOS los módulos **SI permiten** Comunicarse con ellos para trabajar fuera del modo Configuración (Enviando ordenes) Inclusive el modelo HC-06.	Este formato de conexión, permite entrar en Modo Configuración AT3 - **Todos los Módulos HC-05 la Aceptan**. Se puede configurar generando una rutina automática, o accediendo a la placa Arduino desde otro dispositivo, por ejemplo una PC. Para Usar este modo hay que programar una rutina especial.

Generalmente, cuando trabajamos por primera vez con un modulo, nos conectamos, vemos la configuración y si es necesario reconfiguramos con nuestras preferencias. El Módulo mantendrá la configuración hasta que decidamos cambiarla. El proceso de Configuración presentas algunas variantes (según modelo) así que trataré de explicar y ayudar en cada una de las opciones. En único problema que solemos tener, es encontrar la velocidad de comunicación apropiada (de nuestro programa con el modulo Buetooth), el que se soluciona leyendo las características de nuestro modulo, o probando con todas las 8 velocidades posibles

Modo Desconectado (Ver Conexión Formato 1 o 2)	➢ Entra a este estado tan pronto se alimenta el modulo, y no se ha establecido una conexión Bluetooth con ningún otro dispositivo. ➢ EL LED del módulo en este estado parpadea rápidamente ➢ En este estado, el módulo HC-05, no puede interpretar los comandos AT (a diferencia del HC-06).
Estado Conectado o de comunicación (Ver Conexión Formato 1 o 2)	➢ Entra a este estado, cuando se establece una comunicación (conexión) con otro dispositivo Bluetooth. ➢ El LED hace un doble parpadeo. ➢ Todos los datos que se ingresen al HC-05 por el Pin RX se trasmiten por Bluetooth al dispositivo conectado, y los datos recibidos desde el dispositivo conectado, se enviaran por el pin TX a nuestra placa Arduino. La comunicación es.

Modo Configuración AT 1 (Ver Conexión Formato 1)	➢ Para entrar a este estado, después de conectar y alimentar el modulo es necesario presionar el botón del HC-05. ➢ En este estado, podemos enviar comandos AT (a la misma velocidad con el que está configurado). ➢ EL LED del módulo, en este estado parpadea rápidamente igual que en el estado desconectado. ➢ No todos los modelos, permiten esta forma de ingresar al modo Configuración AT1.
Modo Configuración AT 2 (Ver Conexión Formato 1)	➢ Para entrar a este estado es necesario tener presionado el botón al momento de alimentar el modulo, es decir el modulo debe encender con el botón presionado, después de haber encendido se puede soltar y permanecerá en este estado. ➢ En este estado, para enviar comandos AT es necesario hacerlo a la velocidad de 38400 baudios, esto es muy útil cuando nos olvidamos la velocidad con la que hemos dejado configurado nuestro modulo. ➢ También debemos asegurarnos que el monitor este configurado como "NL & CR", (Ver abajo a la Derecha). ➢ EL LED del módulo en este estado parpadea lentamente. ➢ No todos los modelos, permiten esta forma de ingresar al modo Configuración AT2.
Modo Configuración AT 3 (Ver Conexión Formato 2)	➢ Para ingresar en este modo de configuración, requiere que el pin "**EN**" (KEY en algunos modelos) esté en HIGH cuando se enciende el modulo. Para esto conectamos el pin "**EN**" del Modulo al 9 de Arduino y el pin Vcc del módulo BlueTooth al pin 8 de Arduino. Luego podremos controlar todo desde el programa. ➢ Entonces, ponemos en LOW el pin 8 (Conectado a "Vcc" del Modulo) y en HIGH el Pin 9 de Arduino (conectado al pin "**EN**" del modulo). Luego prendemos el Módulo poniendo en HIGH el pin 8. De este modo cuando arranque entrara sin más en el modo de comandos AT. ➢ El resto de los pines se conectan de forma similar a lo que hicimos antes. Txd y Rxd se deben conectar cruzados con los pines de comunicación de Arduino, que usaremos mediante la librería software Serial. ➢ Una vez terminada la configuración, se puede usar el formato de conexión 1 (desconectar y conectar otra vez con formato distinto), o habiendo programado la secuencia apropiada, reiniciar el modulo desde nuestro programa, y esta vez no entrar al modo Configuración.

COMANDOS PARA CONFIGURAR MODULO HC-05

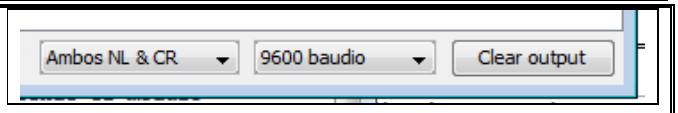
1- Test de comunicación.

Una vez se ha ingresado al Modo Configuración (Cualquiera de los explicados anteriormente), lo primero es comprobar si nuestro Bluetooth responde a los comandos AT.

Enviar: AT
Recibe: OK

Si recibimos como respuesta un OK en el Monitor del Puerto Serie entonces podemos continuar, pero si **NO recibimos el OK**, o recibimos basura ilegible, debemos verificar que:

- ✓ Las conexiones estén correctas, **en el Monitor Puerto Serie** "NL&CR". (Ver abajo a la Derecha). Con esto enviaremos apropiadamente las líneas de comandos.



- ✓ Que en nuestro programa, la velocidad de comunicación con el modulo Bluetooth sea la correcta (Hay fabricantes que cambian esta velocidad). Si el punto anterior quedó descartados como posible falla, entonces recomiendo probar con **todas** las velocidades posibles:

Velocidades	
1 -------	1200
2 -------	2400
3 -------	4800
4 -------	9600
5 -------	19200
6 -------	38400
7 -------	57600
8 -------	115200

En el Programa la velocidad de comunicación con el Modulo Bluetooth la configuramos en la línea "**BT1.begin(38400);**", Es acá donde debemos cambiar y probar, hasta encontrar la velocidad con la que SI Funcione.

IMPORTANTE Salir del Modo Configuración.

Para salir del modo de Configuración, deberá reiniciar el modulo, para esto ejecute el Comando **RESET**

2- Obtener la versión del Firmware.

Este comando, además de conocer la Versión del Firmware de nuestro Modulo, Nos permitirá practicar con el uso de los comando AT, con la certeza de que no tendremos inconvenientes ante un error de nuestra parte.

Enviar: AT+VERSION?
Respuesta: +VERSION<Versión>
Resp Ej: +VERSION:2.0-20100601

- AT+ROLE Nos informa de si está configurado como Maestro 1, o como esclavo 0.
  - AT+ROLE1 Configura el modulo como Master.
  - AT+ROLE0 Configura el modulo como Slave.

Enviar: AT+ORGL
Respuesta: OK
*Al hacer esto todos los parámetros del módulo se restablecen, a los valores por defecto de fábrica. En algunas versiones la velocidad cambia a 38400 baudios y en otros a 9600.

AT+PSWD?

Pregunta cual es el Password
Ver si anda

AT+UART?

Pregunta la velocidad de trabajo, que puede ser distinta a la de configuracion

Obtener Dirección de Nuestro Módulo Bluetooth.

Enviar: AT+ADDR?
Respuesta: +ADDR:<dirección>

Resp Ej: AT+ADDR: 98d3:31:2052e6

Resetear Modulo Bluetooth

Este Comando reinicia el Módulo, y al hacerlo salimos del Modo Configuración AT en que nos encontremos.

Enviar: AT+RESET
Respuesta: OK

Restablecer Valores por Defecto.

Al ejecutar este comando, todos los parámetros del módulo se restablecen, a los valores por defecto de fábrica. En algunas versiones la velocidad cambia a 38400 baudios y en otros a 9600. Consulte en esta guía: "**Configuración por defecto o de Fábrica**" correspondiente al Modulo **HC-05**

Enviar: AT+ORGL
Respuesta: OK

Ahora es necesario compilar y cargar el siguiente sketch que hemos preparado, que como vemos lee los datos enviados de la PC a través de nuestro IDE y se lo envía serialmente hacia los pines RXD y TXD de nuestro módulo HC-05.

```
#include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
SoftwareSerial BT(10,11);    // Definimos los pines RX y TX del Arduino conectados al
Bluetooth

void setup( )
{
  BT.begin(9600);       // Inicializamos el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600);   // Inicializamos  el puerto serie
}

void loop( )
{
  if(BT.available( ))    // Si llega un dato por el puerto BT se envía al monitor
serial
  {
    Serial.write(BT.read( ));
  }

  if(Serial.available( ))  // Si llega un dato por el monitor serial se envía al
puerto BT
  {
     BT.write(Serial.read( ));
  }
}
```

Configurando nuestro Módulo HC-05

En nuestro ejemplo usaremos un conversor USB serial CP2102 que se ha instalado como puerto serial COM5, por lo que antes de abrir el Monitor serial, en nuestro IDE Arduino debemos escoger dicho Puerto.

El siguiente paso es entrar al Modo AT 1 o Modo AT 2:

-Para entrar al modo AT 1, después de alimentar el modulo y haber encendido tan solo basta presionar el botón que tiene el módulo HC-05, el LED del módulo seguirá parpadeando rápidamente, por lo que para saber si hemos entrado al Modo AT 1 es necesario enviar comandos AT y ver si responde, estos comandos se verán más adelante.

-Para entrar al modo AT 2, antes de alimentar o encender el modulo es necesario presionar su botón, mantener presionado y alimentar el modulo, después que enciende recién podemos soltar el botón. Si el LED Parpadea lentamente es porque ya está en Modo AT 2.

Cambiar nombre de nuestro módulo HC-05

Por defecto nuestro bluetooth se llama “HC-05” esto se puede cambiar con el siguiente comando AT

Enviar: AT+NAME=<Nombre> Ejm: AT+NAME=Robot

Respuesta: OK

Cambiar Código de Vinculación

Por defecto viene con el código de vinculación (Pin) "1234", para cambiarlo hay que enviar el siguiente comando AT

Enviar: AT+PSWD=<Pin> Ejm: AT+PSWD=2560

Respuesta: OK

Se puede saber cuál es el pin actual de nuestro modulo, para eso hay que enviar el siguiente comando: **AT+ PSWD?**

Configurar la velocidad de comunicación:

La velocidad por defecto es de 9600 baudios, con Stop bit =0 (1 bit de parada), y sin Paridad, para cambiar estos parámetros, se hace uso del siguiente comando AT:

Enviar: AT+UART=<Baud> ,< StopBit>,< Parity>

Respuesta: OK

Donde :

< Baud > equivale a una velocidad, los valores pueden ser: 4800, 9600, 19200, 38400, 57600, 115200, 23400, 460800, 921600 o 1382400.

< StopBit> es el Bit de parada, puede ser 0 o 1, para 1 bit o 2 bits de parada respectivamente, Para aplicaciones comunes se trabaja con 1 bit por lo que este parámetro normalmente se lo deja en 0.

< Parity> Es la paridad, puede ser 0 (Sin Paridad), 1 (Paridad impar) o 2 (Paridad par). Para aplicaciones comunes no se usa paridad, por lo que se recomienda dejar este parámetro en 0.

Ejemplo:

Enviar: AT+UART=9600,0,0

Respuesta: OK

Se puede saber cuál es la configuración actual, para eso hay que enviar el siguiente comando: **AT+UART?**

Configurar el Role: para que trabaje como Maestro o Esclavo

Por defecto nuestro HC-05 viene como esclavo, el Siguiente comando nos permite cambiar esto:

Enviar: AT+ROLE=<Role> Ejm: AT+ROLE=0

Respuesta: OK

Donde:<Role>

0 -> Esclavo

1 -> Maestro

Para saber cuál es la configuración actual, enviar el siguiente comando: **AT+ ROLE?**

Configurar el modo de conexión (cuando se trabaja como maestro)

Esta configuración aplica para cuando el modulo está trabajando como maestro, el modulo necesita saber si se va a conectar con un dispositivo en particular o con cualquiera que esté disponible.

Enviar: AT+CMODE=<Mode> Ejm: AT+CMODE=1

Respuesta: OK

Donde: < Mode >

0 -> Conectarse a un dispositivo con la dirección especificada(Se utiliza otro comando AT para especificar esta dirección).

1 -> conectar el módulo a cualquier dirección disponible(aleatorio).

Enviar el siguiente comando para averiguar el modo actual de conexión: **AT+ CMODE?**

Especificar la dirección del dispositivo al cual nos vamos a conectar

Esta configuración aplica cunado nuestro modulo está configurado como maestro, y a la vez el modo de conexión está en 0 (CMODE=0) el cual indica que nos vamos a conectar al dispositivo esclavo en particular. Para especificar la dirección al cual nos vamos a conectar se usa el siguiente comando AT

Enviar: AT+BIND=<Address>
Respuesta: OK

Donde:

< Address > Es la dirección del dispositivo al cual nos vamos a conectar, la dirección se envía de la siguiente forma: **1234,56,ABCDEF** la cual equivale a la dirección 12:34:56:AB:CD:EF

Ejemplo:

Enviar: AT+BIND=E668,46,9277F2

Respuesta: OK

Para ver la dirección actual en este parámetro hay que enviar el siguiente comando: **AT+ BIND?**

Ahora veremos Dos ejemplos prácticos, con los pasos para configurar nuestro HC-05 como maestro y esclavo:

Configurando nuestro módulo HC-05 como esclavo:

Realizaremos un ejemplo para configurar nuestro modulo con las siguientes características:

- Modo o role: Esclavo

- Nombre: Robot
- Código de emparejamiento: 1212

- Velocidad o Baud rate: 9600 baudios

A continuación se muestra los pasos para realizar la configuración:

- Entrar en modo AT 1 o Modo AT 2

- Verificar si estamos en modo AT

Enviar: AT

Recibe: OK

- Establecer el Role como Esclavo

Enviar: AT+ROLE=0

Respuesta: OK

- Configurar el Nombre del modulo

Enviar: AT+NAME=Robot

Respuesta: OK

- Establecer el Pin de vinculación

Enviar: AT+PSWD=1212

Respuesta: OK

- Configura la Velocidad

Enviar: AT+UART=9600,0,0

Respuesta: OK

- Verificar los parámetros cambiados

Enviar:

AT+ROLE?

AT+PSWD?

AT+UART?

Respuesta:

+ROLE:0

OK

+PSWD:1212

OK

+UART:9600,0,0

OK

- Resetear el modulo

Enviar: AT+RESET

Respuesta: OK

En la siguiente imagen podemos ver la secuencia de datos recibidos por el monitor serial en el mismo orden en que se realizaron los pasos anteriores.

Después de hacer la configuración anterior, podemos usar el modulo como un dispositivo esclavo, el cual estará siempre en espera de una conexión por parte de una dispositivo bluetooth maestro.

Configurando nuestro módulo HC-05 como Maestro:

Ahora veremos un ejemplo para configurar nuestro modulo como maestro, con las siguientes características:

-Modo o role: Maestro

-Nombre: Naylamp

-Código de emparejamiento: 1465 (La misma que el dispositivo a conectarse)

-Velocidad o Baud rate: 57600 baudios

-Dirección del dispositivo esclavo con el que se desea conectar: 98:D3:31:20:3A:D0

A continuación se muestra los pasos para realizar la configuración:

- Entrar en modo AT 1 o Modo AT 2

- Verificar si estamos en modo AT

Enviar: AT

Recibe: OK

- Establecer el Role como Maestro

Enviar: AT+ROLE=1

Respuesta: OK

- Configurar el Nombre del modulo

Enviar: AT+NAME=Naylamp

Respuesta: OK

- Establecer el Pin de vinculación

Enviar: AT+PSWD=1465

Respuesta: OK

- Configura la Velocidad

Enviar: AT+UART=57600,0,0

Respuesta: OK

- Configurar el modo de conexión

Enviar: AT+CMODE=0

Respuesta: OK

- Especificar la dirección del dispositivo a conectarse

Enviar: AT+BIND=98D3,31,203AD0

Respuesta: OK

- Verificar los parámetros cambiados

Enviar:

AT+ROLE?

AT+PSWD?

AT+UART?

AT+CMODE?

AT+BIND?

Respuesta:

+ROLE:1

OK

+PSWD:1465

OK

+UART:57600,0,0

OK

+CMOD:0

OK

+BIND:98d3:31:203ad0

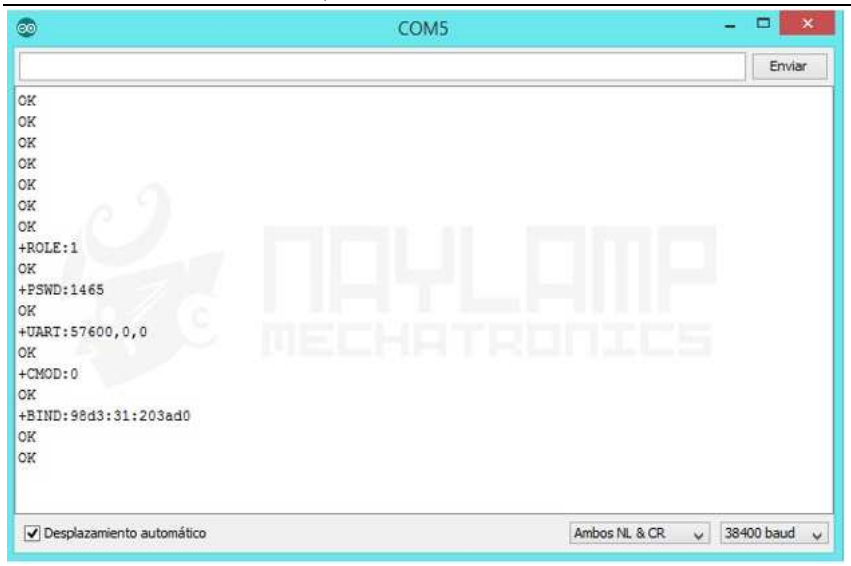
OK

- Resetear el modulo

Enviar: AT+RESET

Respuesta: OK

En la siguiente imagen podemos ver la secuencia de datos recibidos por el monitor serial en el mismo orden en que se realizaron los pasos anteriores.



Después de haber hecho las configuraciones, nuestro modulo se comporta como un dispositivo maestro, el cual estará constantemente buscando el dispositivo especificado hasta encontrarlo y conectarse. Para que el Maestro pueda conectarse con el dispositivo esclavo, ambos deben tener el mismo código de vinculación.

Hc-06	https://naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html
HC-05 01	https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html
HC-05 02	https://www.prometec.net/bt-hc05/
HC-05 03	https://miarduinounotieneunblog.blogspot.com/2016/12/configurar-modulos-bluetooth-hc-05-y-hc.html
HC-05 04	http://tutoriales.coopertec.com.ar/2016/10/22/configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at/
HC-05 05	https://saber.patagoniatec.com/2014/11/hc-05-bluetooth-conectar-esclavo-hc05-maestro-master-save-wireless-tutorial-iot-celular-smartphone-arduino-argentina-ptec/

Apéndice A - Comandos Bluetooth desde un Celular

Esto es realmente muy simple. Solo necesitaremos un celular con muy poco espacio libre, y en pocos minutos ya estará listo.

Existen muchos programas que se nos ofrecen listos para usar, que sirven específicamente para controlar dispositivos por medio de Bluetooth.. Estos programas emulan joystick de todo tipo, y nos permiten configurar el comando a enviar. Solo debemos elegir el que más nos guste.

Los pasos a seguir son:

1. Ingresamos a la carpeta de aplicaciones de nuestro celular y abrimos **Play Store** ó **App Store** (Según sea el Celular que estemos usando).
2. Buscaremos "**Arduino Bluetooth Controler**" y el buscador nos mostrara varios programas. Abra que instalar y ver cual nos sirve mejor (si no sirve hay que desinstalarlo). Concretamente hay que buscar

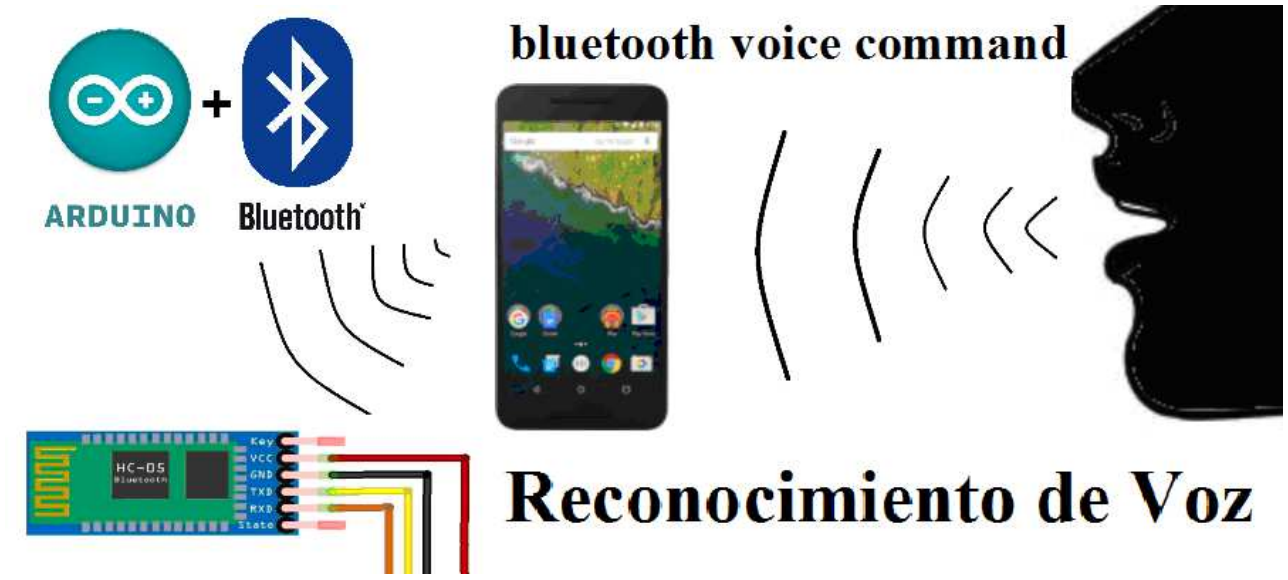
uno que se parezca a un joystick, y cuanto más simple mejor, por ahora solo será necesario que nos permita configurar las funciones básicas y elementales, por ejemplo: 1- Arrancar, 2- Parar, 3- Adelante, 4- Atrás, 5- Derecha, 6- Izquierda, ya que será lo que deberemos reconocer en nuestro programa para ejecutar una orden.

3. Instalar la aplicación seleccionada. Recuerda dar acceso al programa para que active automáticamente el Bluetooth de tu celular.
4. Configurar una Letra o número a enviar para cada uno de los comandos, y te recomiendo que los anotes, o deberás ingresar a la aplicación a consultar, cada vez que te haga falta.
5. y Ahora a Divertirse..!!

Por otro lado, si buscas "**bluetooth voice command**" encontraras aplicaciones que reconocen nuestra voz, lo transforman a texto y envían la cadena por Bluetooth. Esto permitirá hablar y que tu Arduino obedezca. Y lo más interesante es que si los programas que usaste con el emulador de joystick funcionaron, con el reconocimiento de vos también funcionarán.

Tu Hablas	Aplicación Del Celular Reconoce La Voz, Convierte En Texto Y Envía A Tu Módulo Bluetooth	Modulo Bluetooth, Recibe Texto Y Entrega A Placa Arduino	Tu Programa En Placa Arduino Reconoce Texto Y Ejecuta Orden

Para saber más sobre reconocimiento de texto y las funciones que se usan, debe consultar "Anexo_01_Procesamiento_de_Texto".



Apéndice B - Librería SoftwareSerial

(SoftwareSerial.h)

El hardware Arduino ha incorporado soporte para la comunicación serie en los pines 0 y 1 (que también va al ordenador a través de la conexión USB).

La biblioteca SoftwareSerial ha sido desarrollada para permitir la comunicación serie en otros pines digitales del Arduino, usando el software para replicar la funcionalidad (de ahí el nombre de "SoftwareSerial"). Es posible tener múltiples puertos serie de programas con velocidades de hasta 115200 bps. Un parámetro permite la señalización invertida para dispositivos que requieren ese protocolo.

Todas las versiones de SoftwareSerial, desde la versión 1,0, se basan en la biblioteca NewSoftSerial escrita por Mikal Hart.

LIMITACIONES

A continuación, se detallan algunas limitaciones encontradas, pero hay que tener en cuenta que hay muchos fabricantes que emulan placas Arduino y otros tantos desarrolladores del Software, por lo tanto estos pines que les dejo, pueden variar de una placa a otra.

- No todos los pines en las placas soportan interrupciones del tipo requeridas para la comunicación, por lo tanto, si en una, un par de pines reasignados como RX - TX no funcionan, probar con otros. A continuación algunos que Si funcionan, que he podido probar.

Modelo	Si Funcionan
Mega y Mega 2560	10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).
Leonardo y Micro	8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI)
UNO	10, 11 y otros

- Con la librería **SoftwareSerial** pueden ser creados objetos múltiples, sin embargo, sólo uno puede estar activo en un momento dado. Si hubiera más de uno, se presentan fallas o directamente no se activa el segundo.

Para saber más sobre reconocimiento de texto y las funciones que se usan, debe consultar "Anexo_01_Procesamiento_de_Texto".

FUNCIONES DE LA LIBRERÍA

SoftwareSerial(rxPin, txPin, Inversor_Logico)

Descripción:
La función SoftwareSerial( ), se utiliza para crear una instancia de un objeto del tipo SoftwareSerial, cuyo nombre es necesario proporcionar al igual que en el ejemplo. Con SoftwareSerial pueden ser creados objetos múltiples, sin embargo, sólo uno puede estar activo en un momento dado. Es necesario llamar a SoftwareSerial.begin( ) para establecer la velocidad y permitir la comunicación.

Parámetros:
rxPin: el pin en que se desea recibir datos serie
TxPin: el pin en el que se transmiten los datos
Inversor_Logico: este argumento es opcional, y si no se usa es falso por defecto. Se utiliza para invertir el sentido de los bits entrantes. Si se activa, SoftwareSerial pone un nivel LOW (0 voltios) en el pin RX como 1 bit (estado de reposo) y un nivel HIGH (5 voltios) como 0 bits. También afecta a la forma en que se escribe en el pin TX. El valor por defecto es false.

Retornos:
No retorna Valor.

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones
 const byte rxPin = 2;
 const byte txPin = 3;
```

```
SoftwareSerial NuevoPuerto(rxPin, txPin);  // configura un nuevo objeto serie
```

available( )

Descripción:
Obtiene el número de bytes (caracteres) disponibles para la lectura de un puerto serie software. Se trata de datos que ya llegaron y se almacenan en la memoria intermedia de recepción (buffer) que tiene un tamaño de 64 bytes. Cuando se llena ese buffer el resto de elementos recibidos se pierden.

Sintaxis:
NuevoPuerto.available( )

Parámetros:
Ninguno

Retornos:
El número entero, que representa la cantidad de bytes disponibles para leer.

Ejemplo:
En este ejemplo, Recibe desde hardware serie, envía al software serie y luego, recibe desde software serie, envía al hardware serie.

```
// No todos los pines de la Mega y Mega 2560 soportan interrupciones del cambio de nivel,
// por lo que solamente los siguientes se puede utilizar para
//RX: 10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69
// No todos los pines de Leonardo soportan las interrupciones de cambio de nivel,
// por lo que solamente la siguiente se pueden utilizar para RX: 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

 SoftwareSerial NuevoPuerto(10,11);  //TX = pin digital 10, RX = pin digital 11
// RECORDAR QUE:
// * RX debe conectarse al otro dispositivo en TX
// * TX debe conectarse al otro dispositivo en RX

int cantidad;

void setup( ) {
  Serial.begin(9600);  // Abre las comunicaciones serie y espera a que se abra el puerto:
  while (!Serial) {
    ;  // Espera a que el puerto serie se conecte. Necesario para el puerto USB nativo solamente
  }

  Serial.println("Hoola..!!");

  // Configura la velocidad de datos para el puerto SoftwareSerial
  NuevoPuerto.begin(9600);
  NuevoPuerto.println("Hola, mundo?");
}

void loop( ) {
   cantidad = NuevoPuerto.available( );
   Serial.println( cantidad );
   if (cantidad > 0) {
```

```
      Serial.println( " ");
      Serial.write( NuevoPuerto.read( ) );
  }
  cantidad = Serial.available( );
  Serial.println( cantidad );
  if (cantidad > 0) {
     Serial.println( " ");
     NuevoPuerto.write(Serial.read( ));
  }
}
```

begin(velocidad)

Descripción:
Configura la velocidad (en baudios) para la comunicación serie. Las velocidades de transmisión admitidas son 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, o 115200. También es posible configurar el número de bits de datos, la paridad y los bits de stop, por defecto es 8 bits de datos, sin paridad y un bit de stop.

Sintaxis:
NuevoPuerto.begin(velocidad). Esta función (método) puede recibir un segundo parámetro o argumento, pero no lo usaremos en esta guía

Parámetros:
velocidad: la velocidad de transmisión (long)

Retornos:
Ninguno

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

#define rxPin 10
#define txPin 11

// configura un nuevo puerto serie
SoftwareSerial NuevoPuerto(rxPin, txPin);

void setup( )  {
 NuevoPuerto.begin(9600);  // Configura velocidad de comunicación
}

void loop( ) {
  // ...
}
```

end( )

Descripción:
Deshabilita la comunicación serie permitiendo a los pines RX y TX ser usado como pines digitales (los pines quedan liberados). Para re establecer la comunicación debe llamarse nuevamente la función **begin( )**

Parámetros:
Ninguno

Retornos:
Esta función no retorna información o dato alguno.

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11

int bytesSent;

void setup( ){
    PuertoUno.begin(9600);   // Inicia PuertoUno  (puertos software)
}

void loop( ){
  PuertoUno.end( ); // detiene la comunicación. Los pines quedan liberados
                    // Debe ejecutar begin( ) nuevamente para re establecer comunicacion
 }
```

isListening( )

Descripción:
Verifica si el puerto serie software solicitado está escuchando activamente (si está activo).

Sintaxis:
NuevoPuerto.isListening( )

Parámetros:
Ninguno

Retornos:
Retorna un verdadero o falso si esta o no activamente escuchando (boolean)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

 SoftwareSerial portOne(10,11);  //TX = pin digital 10, RX = pin digital 11

void setup( ) {
  Serial.begin(9600);      // Inicia el puerto serie hardware
  portOne.begin(9600);  // Inicia el puerto serie software
 }

 void loop( ) {
    if (portOne.isListening( )) {
       Serial.println(" El puerto Uno esta a la escucha! ");
    }else{
       Serial.println(" El puerto Uno No escucha! - Está Activo");
    }
 }
```

listen( )

Descripción:
Permite al puerto serie software seleccionado escuchar. Sólo un software de puerto serie puede escuchar a la vez; los datos que llegan de otros puertos serán descartados. Cualquier dato recibido se descarta ya durante la llamada a **listen( )** (a menos que la instancia ya esté a la escucha).

Sintaxis:
NuevoPuerto.listen( )

Parámetros:
NuevoPuerto: el nombre de la instancia a escuchar

Retornos:
Ninguno

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11
SoftwareSerial PuertoDos( 8 ,  9 ); //TX = pin digital 8,   RX = pin digital 9

void setup( ){
    Serial.begin(9600);     // Inicia puerto serie hardware (que usaremos para el monitor)

    PuertoUno.begin(9600);   // Inicia PuertoUno  (puertos software)
    PuertoDos.begin(9600);   // Inicia PuertoDos  (puertos software)
}

void loop( ){
  PuertoUno.listen( );

 if (PuertoUno.isListening( )) {
    Serial.println("Puerto Uno a la escucha!");
 }else{
    Serial.println("Puero Uno no esta a la escucha!");
 }

 if (PuertoDos.isListening( )) {
     Serial.println("Puerto Dos a la escucha!");
 }else{
     Serial.println("Puerto Dos no esta a la escucha!");
 }
}
```

overflow( )

Descripción:
Verifica si se ha producido un desbordamiento de la memoria intermedia serie de software (buffer) que generalmente tiene 64 bytes (Cuando se llena ese buffer el resto de elementos recibidos se pierden). Es muy importante saber que la llamada a esta función, borra la bandera de desbordamiento, lo que significa que las llamadas posteriores retornarán false a no ser que otro byte de datos se haya recibido y se

descarte mientras tato.

Sintaxis:
NuevoPuerto.overflow( )

Parámetros:
Ninguno

Retorna:
Esta función retorna un valor lógico true - false - (boolean)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial portOne(10,11);  //TX = pin digital 10, RX = pin digital 11

void setup( ) {
   Serial.begin(9600);     // Inicia puerto serie hardware (que usaremos para el monitor)
   portOne.begin(9600); // Inicia el puerto serie software
}

void loop( ) {
  if (portOne.overflow( )) {
    Serial.println("Desbordamiento de SoftwareSerial!");
  }
}
```

---

parseFloat( )

Descripción:
No he encontrado una adecuada documentación para ser usado en Librería SoftwareSerial, sin embrago debería devolver el primer número de coma flotante válido del buffer serie. Los caracteres que no sean dígitos (o el signo menos) se omiten. **parseFloat( )** termina con el primer carácter encontrado que no sea un número de coma flotante. **Interesante:** si uso esta función no presenta error de compilación.

---

parseInt( )

Descripción:
No he encontrado una adecuada documentación para ser usado en Librería SoftwareSerial, sin embrago debería Busca el siguiente número int válido en la entrada. **Interesante:** si uso esta función no presenta error de compilación.

---

peek( )

Descripción:
Devuelve un carácter recibido el pin RX del puerto serie de software. Hay que tener en cuenta, que las posteriores llamadas a esta función devolverán el mismo carácter.
Recordar siempre, que sólo una instancia SoftwareSerial, puede recibir datos entrantes a la vez. Si hubiere más de uno, debe seleccionar uno de ellos con la función **listen( )**.

Sintaxis:
NuevoPuerto.peek( );

Parámetros:
Ninguno

Retornos:
El carácter leído, o -1 si no hay disponibles.

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial NuevoPuerto(10,11);  //TX = pin digital 10, RX = pin digital 11

void setup( ){
    NuevoPuerto.begin(9600); // Inicia el puerto serie software
}

void loop( ){
  char c = NuevoPuerto.peek( );
}
```

---

print(dato)

Descripción:
Envía datos al pin de transmisión del puerto serie software. Los datos son entregados como texto ASCII, aunque también permite enviar o imprimir en otros formatos. Funciona igual que la función **Serial.print(**).

Sintaxis:
NuevoPuerto.print( );

Parámetros:
Varían igual que en la función **Serial.print(**). Sugiero consultarla función **Serial.print( )** para más detalles.

Retornos:
La función **print ( )** devolverá el número de bytes escritos, la lectura de ese número es opcional (byte)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial NuevoPuerto(10,11);  //TX = pin digital 10, RX = pin digital 11
int ValorAnalogico;

void setup( ){
    Serial.begin(9600);     // Inicia puerto serie hardware (que usaremos para el monitor)
    NuevoPuerto.begin(9600); // Inicia el puerto serie software
}

void loop( ){
  ValorAnalogico = analogRead(A0); // lee la entrada analógica del pin A0

  // Formatos en los que se puede imprimir :
```

```
  serial.print( ValorAnalogico );        // imprime como un ASCII-codificado decimal
  serial.print("\t");              // imprime un caractér tab
  serial.print( ValorAnalogico, DEC);    // imprime como un ASCII-codificado decimal
  serial.print("\t");              // imprime un caractér tab
  serial.print( ValorAnalogico, HEX);    // imprime como un ASCII-codificado hexadecimal
  serial.print("\t");              // imprime un caractér tab
  serial.print( ValorAnalogico, OCT);    // imprime como un ASCII-codificado octal
  serial.print("\t");              // imprime un caractér tab
  serial.print( ValorAnalogico, BIN);    // imprime como un ASCII-codificado binary
  serial.print("\t");              // imprime un caractér tab
  serial.print( ValorAnalogico/4, BYTE); // imprime como un valor byte en bruto (divide el
                                         // valor por 4 porque analogRead( ) devuelve números
                                         // de 0 a 1023, pero un byte solo puede soportar valores
                                         // de hasta 255)
  serial.print("\t");              // imprime un caractér tab
  serial.println( );                // imprime un caractér de retorno de línea

   delay(10);  // espera 10 millisegundos antes de la siguiente lectura:
}
```

println(dato)

Descripción:
Envía datos al pin de transmisión del puerto serie software (igual que **Serial.print( ) -** y agrega al final de la cadena caractér de retorno de línea - Baja de renglón).

Sintaxis:
NuevoPuerto.println( );

Parámetros:
Varían igual que en la función **Serial.print(**). Sugiero consultarla función **Serial.print(**) para más detalles.

Retornos:
La función **print ( )** devolverá el número de bytes escritos, la lectura de ese número es opcional (byte)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial NuevoPuerto(10,11);  //TX = pin digital 10, RX = pin digital 11
int ValorAnalogico;

void setup( ){
    Serial.begin(9600);     // Inicia puerto serie hardware (que usaremos para el monitor)
    NuevoPuerto.begin(9600); // Inicia el puerto serie software
}

void loop( ){
  ValorAnalogico = analogRead(A0); // lee la entrada analógica del pin A0

  // Formatos en los que se puede imprimir
  serial.println( ValorAnalogico );  // imprime como un ASCII-codificado decimal y baja
  serial.println( ValorAnalogico );  // imprime como un ASCII-codificado decimal y baja
```

```
  serial.println( ValorAnalogico );  // imprime como un ASCII-codificado decimal y baja
   serial.println( );                  // imprime un caractér de retorno de línea (baja de línea)

  delay(10);  // espera 10 millisegundos antes de la siguiente lectura:
}
```

read( )

Descripción:
Devuelve un caractér recibido por el pin RX del puerto serie de software. Recordar siempre, que sólo una instancia SoftwareSerial, puede recibir datos entrantes a la vez. Si hubiere más de uno, debe seleccionar uno de ellos con la función **listen( )**.

Sintaxis:
NuevoPuerto.read( );

Parámetros:
Ninguno

Retornos:
El carácter leído, o -1 si no hay disponibles

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial NuevoPuerto(10,11);  //TX = pin digital 10, RX = pin digital 11

void setup( ){
    NuevoPuerto.begin(9600); // Inicia el puerto serie software
}

void loop( ){
  char c = NuevoPuerto.read( );
}
```

readBytes( )

Descripción:
Lee datos del buffer serie la cantidad de bytes que se indican y lo guarda en una variable buffer. La función finaliza si la longitud determinada se ha leído, y ya no hay más que leer, termina cuando se agota el tiempo. Serial.readBytes( ).Ver función **setTimeout ( )**.

Sintaxis:
Serial.readBytes (buffer, longitud)
Serial.readBytes (buffer, sizeof(buffer))

Parámetros:
búfer: el búfer para almacenar los bytes en (char [] o byte [])
longitud: el número de bytes para leer (int)

Devoluciones:
Devuelve la cantidad de caracteres que pudo leer y fueron colocados en el búfer. Un cero significa que

no se encontraron datos válidos. (byte)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11

byte bytesLeidos;
char buffer[100];

void setup( ){
    Serial.begin(9600);
    PuertoUno.begin(9600);   // Inicia PuertoUno  (puerto software)
}

void loop( ){
    bytesLeidos = PuertoUno.readBytes ( buffer,  sizeof(buffer) );
    Serial.print( "Leídos: ");
    Serial.print(bytesLeidos);
    Serial.print( " - ");
    Serial.println( buffer );
}
```

write(dato)

Descripción:
Envía o escribe datos en el pin de transmisión del puerto serie software como bytes en bruto. Funciona igual que la función **Serial.write( )**

Parámetros:
Ver **Serial.write( )** para detalles

Retornos:
La función **write( )** devolverá el número de bytes escritos, la lectura de ese número es opcional (byte)

Ejemplo:

```
#include <SoftwareSerial.h> // incluye biblioteca SoftwareSerial para usar funciones

SoftwareSerial PuertoUno(10,11);  //TX = pin digital 10, RX = pin digital 11

int bytesSent;

void setup( ){
    PuertoUno.begin(9600);   // Inicia PuertoUno  (puertos software)
}

void loop( ){
  PuertoUno.write(4); // envia un byte con el valor 4
  bytesSent = PuertoUno.write("Hola");  //Envía cadena "Hola" y devuelve longitud  de la cadena.
 }
```

setTimeout ( )

Descripción
Permite configurar el máximo de milisegundos de espera para la lectura del puerto. **Esta función se comporta igual que lo hace cuando configuramos el puerto serial.**

Serial.setTimeout ( ) establece los milisegundos máximos para esperar datos en serie al usar

En las funciones Serial.readBytesUntil( ), Serial.readBytes( ), Serial.parseInt( ) o Serial.parseFloat( ). El valor del tiempo de espera por defecto es 1000 milisegundos.

Serial.setTimeout( ) hereda de la clase de utilidad Stream.

Sintaxis
Serial.setTimeout (tiempo)

Parámetros
Es el tiempo de espera en milisegundos (long).

Retornos
Ninguno

---

readBytesUntil( ) – lee caracteres del buffer serie y los guarda en un array de caracteres, la función termina si el carácter terminador es encontrado o la longitud determinada se a leído o ha alcanzado el timeout.
https://www.arduino.cc/en/Serial/ReadBytesUntil

---

serialEvent( ) – llamado cuando hay datos disponibles. https://www.arduino.cc/en/Reference/SerialEvent

---

flush( ) – espera hasta la transmisión completa de los datos salientes.
https://www.arduino.cc/en/Serial/Flush

---

peek( ) – devuelve el siguiente carácter del buffer serie pero sin borrarlo de él.
https://www.arduino.cc/en/Serial/Peek

---

readString( ) – lee caracteres del buffer serie y los guarda en un string. La función termina cuando se produce un timeout.
https://www.arduino.cc/en/Serial/ReadString

00100010 01000101 01101100 00100000 01110000 01110010 01100101 01110011 01100101
01101110 01110100 01100101 00100000 01100101 01110011 00100000 01100100 01100101
00100000 01100101 01101100 01101100 01101111 01110011 00101100 00100000 01110000
01100101 01110010 01101111 00100000 01100101 01101100 00100000 01100110 01110101

01110100 01110101 01110010 01101111 00101100 00100000 01110000 01101111 01110010
00100000 01100101 01101100 00100000 01110001 01110101 01100101 00100000 01111001
01101111 00100000 01101000 01100101 00100000 01110100 01110010 01100001 01100010
01100001 01101010 01100001 01100100 01101111 00101100 00100000 01100101 01110011
00100000 01101101 11000011 10101101 01101111 00100010 00101110 01001110 01101001
01101011 01101111 01101100 01100001 01010100 01100101 01110011 01101100 01100001
00100000 00110001 00110000 00101111 00110000 00110111 00101111 00110001 00111000
00110101 00110110 00100000 00101101 00100000 00110000 00110111 00101111 00110000
00110001 00101111 00110001 00111001 00110100 00110011 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.