Robot "Arima" Bluetooth

José Ríos Sanchez



Tabla de contenido

[Materiales 3](#_Toc452953055)

[Conexión ARDUINO 3](#_Toc452953056)

[- Conexión Bluetooth *(HC-06)* : 3](#_Toc452953057)

[- Programa Android *(Bluetooth Serial Controller [NEXT PROTOTYPES])*: 4](#_Toc452953058)

[- Controller(A-E) : 5](#_Toc452953059)

[- Name: 5](#_Toc452953060)

[- Command: 5](#_Toc452953061)

[- Visibility: 5](#_Toc452953062)

[- Conexión Puente H (L298N): 5](#_Toc452953063)

[ARDUINO-Bluetooth 7](#_Toc452953064)

[- Intento 1:LED 13 7](#_Toc452953065)

[- Intento 2: Puente H 8](#_Toc452953066)

# Materiales

* 4 Motores DC - 9V
* Engranajes y Cajas reductoras
* 2 LEDs
* 2 Puentes H L298n ($8000 c/u)
* 1 Bluetooth esclavo HC-06 ($6900 c/u)
* 1 ARDUINO nano ($7000 c/u)
* 1 Celular con sistema Android
* Total a gastar: $30.000 aprox.

# Conexión ARDUINO

La conexión a ARDUINO se separa en dos tipos de conexiones principalmente, debido a que son conexiones de diferente estructura, pero que necesitan ser complementadas para ser ocupadas de manera integra en el robots a control remoto que se desea diseñar.

## - Conexión Bluetooth (HC-06) :

El modulo Bluetooth funciona mediante el protocolo IIC (I2C) , en donde será necesario configurar inicialmente el modulo para que pueda ser sincronizado, esto se hace mediante los comandos AT (comandos de atención, que se ocupan para indicar que lo que viene a continuación será una orden de configuración y no deberá transmitirse), para lo cual será necesario realizar la configuración de la Ilustración 1[[1]](#footnote-1), de esta manera el modulo HC-06 queda listo para ser sincronizado, pero antes de eso, es necesario cargar el siguiente programa, que se encarga de mandar configuraciones AT y de este modo poder colocar un nombre visible a nuestro Bluetooth, y una contraseña para la vinculación (en algunos casos puede ser necesario cambiar la velocidad de comunicación entre modulo y consola):

#include <SoftwareSerial.h>

SoftwareSerial BT1(3,2); // Habilita dos pines cualquieras a RX, TX

char lectura1;

void setup() {

Serial.begin(9600); //Inicia el serial PC-Arduino

Serial.println("Enter AT commands:"); //Mensaje de inicio

BT1.begin(9600); //Inicia serial Android-Arduino

}

void loop() {

if (BT1.available()) // Si existen datos desde el Android

lectura1=(BT1.read()); // Guarda en la variable "lectura1" lo que manda el android

Serial.write(lectura1); // Escribe en el serial el dato mandado

if (Serial.available()) // Si el serial del PC recibe datos

{ String S = GetLine(); // guarda lo escrito en una cadena llamada S

BT1.print(S); // y se imprime en el Android S

Serial.println("---> " + S); // y además se imprime en el Serial del PC el valor ingresado

}

}

String GetLine()

{ String S = "" ;

if (Serial.available())

{ char c = Serial.read(); ;

while ( c != '\n') //Hasta que el caracter sea intro

{ S = S + c ;

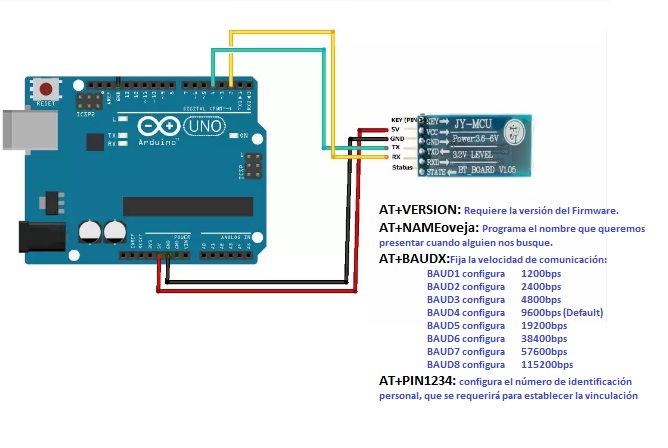
delay(25) ;

c = Serial.read();

}

return( S + '\n') ;

}

 }

## - Programa Android (Bluetooth Serial Controller [NEXT PROTOTYPES]):

Ilustración

El programa ocupado posee un interfaz fácil de manejar, entregando la opción de 5 controles distintos con un total de 25 botones seleccionables por control; además de un botón de conexión, con el cual se buscara el dispositivo Bluetooth con el que nos sincronizaremos. Y un botón de preferencias en donde nos encargaremos de todos los aspectos visuales y estructurales del funcionamiento del programa, entre los cuales se encuentran:

- Controller(A-E) : Selección de el control que uno desee ocupar para diferentes tipos de proyectos.

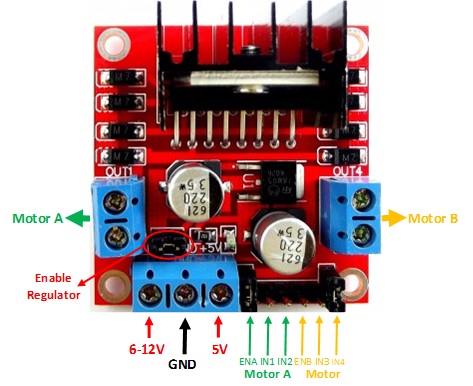
- Name: Nombre con el cual se visualizaran los botones seleccionados

- Command: Es la opción mas importante a tener en consideración, puesto que en ella se decide el tipo de comando que se enviara desde el celular al ARDUINO (ASCII ó HEX), y lo que llegara hasta el modulo (futura acción a realizar mediante la lectura de este dato). Además posee la opción de "STOP COMMAND" la cual indica si se desea mandar un comando en el momento en que se termina de presionar uno de estos botones; esto es de gran importancia para generar el fin de una acción en el mismo instante en que se deja de presionar algún botón.

- Visibility: Se seleccionan los botones que se quieren visibles para el usuario, de este modo se elimina el exceso de botones innecesarios en el control.

## - Conexión Puente H (L298N):

El puente H L298N es un modulo útil para el control de motores DC, desde donde se pueden manejar la dirección del giro de al menos dos de estos motores. Además de tener diodos de protección y un regulador de voltaje a 5 v.

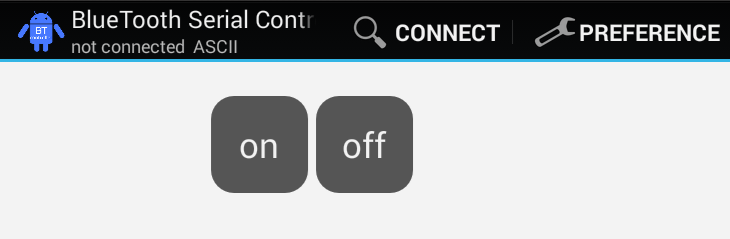


Ilustración

# ARDUINO-Bluetooth

## - Intento 1:LED 13

El primer ejemplo será simple, mediante dos botones de un control del programa anteriormente mencionado, se encenderá o apagara el LED del Pin 13 del ARDUINO.



Ilustración

En este control, se decide dejar la visibilidad del botón 1 y 2, entregando aquel mismo comando a la consola de ARDUINO (y sin necesariamente entregar un carácter que indique cuando se dejo de apretar el botón); finalmente se le cambia el nombre visible a on:1 y off:2.

Es necesario indicar que el comando que entrega esta en código ASCII, por lo que leerá ARDUINO será un carácter (char), en el siguiente programa:

#include <SoftwareSerial.h>

SoftwareSerial BT1(3,2);

char lectura1;

void setup() {

Serial.begin(9600); //Inicia el serial PC-Arduino

BT1.begin(9600); //Inicia serial Android-Arduino

Serial.println("Enter AT commands:"); //Mensaje de inicio

pinMode(13,OUTPUT); // Declara el PIN13 como salida

}

void loop() {

if (BT1.available()>0){ // Si existen datos desde el Android los lee

lectura1=(BT1.read()); // Guarda en la variable "lectura1" lo que manda el android

if(lectura1=='1'){ // En caso de que llegue el comando 1 activa lo siguiente:

digitalWrite(13,1); // Enciende el LED13

Serial.println(lectura1); // Escribe en el serial el comando mandado

}

if(lectura1=='2'){

digitalWrite(13,0); // Apaga el LED13

Serial.println(lectura1); // Escribe en el serial el dato mandado

}

}

}

## - Intento 2: Puente H

El siguiente ejemplo ocupa el mismo control anterior de dos botones, en donde si se envía el comando "char 1" el motor girara en el sentido de las agujas del reloj, y de manera contraria en caso de que llegue el comando "char 2". Se ocupa el siguiente programa:

#include <SoftwareSerial.h>

SoftwareSerial BT1(3,2);

char lectura1;

int IN3 = 4; //Nombres control Puente H L298N

int IN4 = 5;

void setup() {

Serial.begin(9600); //Inicia el serial PC-Arduino

BT1.begin(9600); //Inicia serial Android-Arduino

Serial.println("Enter AT commands:"); //Mensaje de inicio

pinMode(IN3,OUTPUT); //Declara pines de salida de motores

pinMode(IN4,OUTPUT);

}

void loop() {

if (BT1.available()>0){ // Si existen datos desde el Android los lee

lectura1=(BT1.read()); // Guarda en la variable "lectura1" lo que manda el android

if(lectura1=='1'){ // En caso de que llegue el comando 1 activa lo siguiente:

digitalWrite(IN3,1); // Indica que el motor valla en una dirección

digitalWrite(IN4,0); // y luego de 250 milisegundos se detenga

Serial.println(lectura1); // Escribe en el serial el comando mandado

delay(250);

digitalWrite(IN3,0);

digitalWrite(IN4,0);

}

if(lectura1=='2'){

digitalWrite(IN3,0); // Indica que el motor valla en dirección opuesta

digitalWrite(IN4,1); // Para luego de 250 milisegundos detenerse

Serial.println(lectura1); // Escribe en el serial el comando mandado

delay(250);

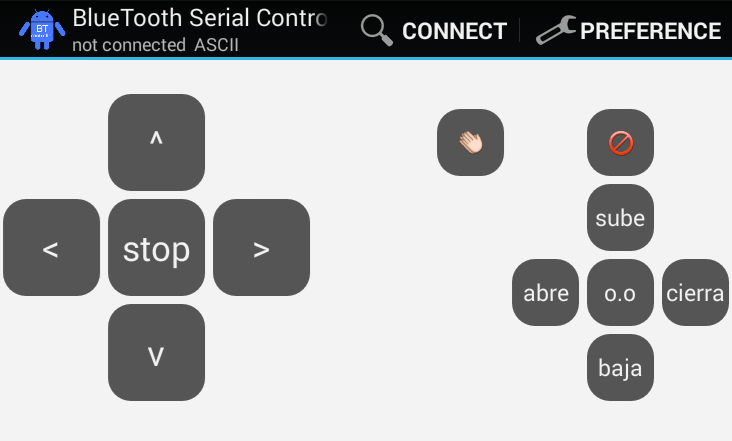
digitalWrite(IN3,0);

digitalWrite(IN4,0);

}

}

}



1. Se toman los pines 2 y 4 como Rx y Tx respectivamente para no ocupar los pines 0 y 1 que vienen configurados por defecto en ARDUINO UNO y de este modo poder mantener la comunicación serial de ordenador con el que se esté trabajando, lo cual es de gran ayuda para ir testeando que el trabajo funciona como se desea. [↑](#footnote-ref-1)