# Simple RC car for beginners (Android control over Bluetooth)

by tolik777 on January 24, 2013

**Table of Contents**

Simple RC car for beginners (Android control over Bluetooth) . . . . . 1

Intro: Simple RC car for beginners (Android control over Bluetooth) . . . . . 2

Step 1: Android device . . . . . 2

Step 2: DIY Car Chassis . . . . . 3

Step 3: Controller (MCU) . . . . . 3

Step 4: Bluetooth module . . . . . 4

Step 5: Motor Driver . . . . . 4

Step 6: Other parts . . . . . 4

Step 7: Theory . . . . . 5

Step 8: Android Application . . . . . 5

Step 9: Arduino RC car wiring . . . . . 7

Step 10: Software . . . . . 8

File Downloads . . . . . 10

Related Instructables . . . . . 10

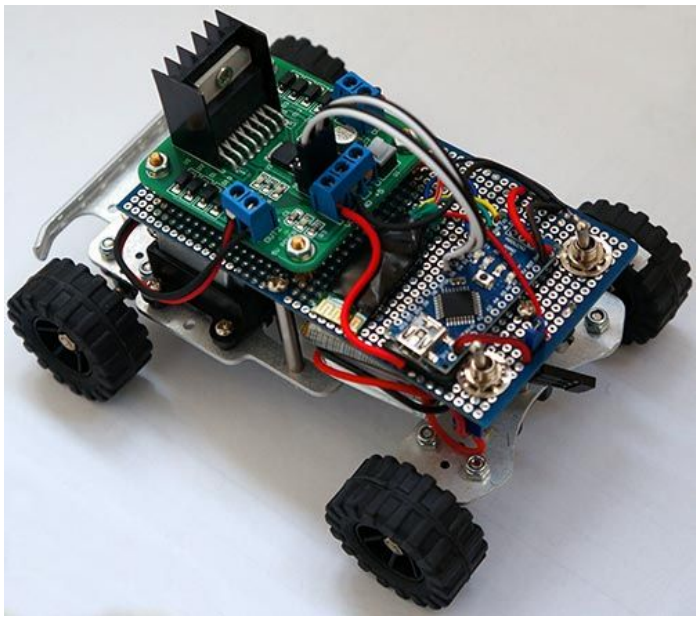
Advertisements . . . . . 11

**Author:tolik777** Electronic circuits
I like electronics, Robots and Arduino!

## Intro: Simple RC car for beginners (Android control over Bluetooth)

This is a simple project of Android Bluetooth Car with Bluetooth control. Arduino controller is used

To control the car used Android-device with a built-in accelerometer. Tilt forward - car goes forward, tilt to the left - car turns to the left, tilt back - car goes back. Speed of movement or rotation depends on how much you tilt the device. Sensitivity and value of the tilt set in the configuration Android-apps. Also are provided a normal way to control: the buttons on the screen. In addition to all I implemented the touch control. Total 3 ways to control the RC Car.

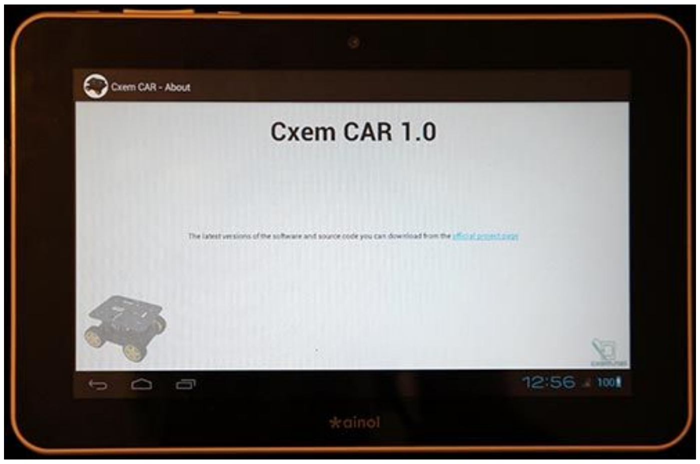


## Step 1: Android device

**Parts needed**

**1. Android device**

The most important part - Android device with accelerometer and Bluetooth: tablet, smartphone and other... As an Android device, I used a cheap Chinese tablet "Ainol Aurora" with an external USB-Bluetooth module (because its not have own), connected via USB Host.



Cxem CAR 1.0

The latest versions of the software and source code you can download from the official project page

## Step 2: DIY Car Chassis

**2. DIY Car Chassis**

We also need any chassis with 2 or 4 DC motors. You can use an old RC toy car. As a platform I used a small RC DIY platform, bought on eBay for $ 25. To control described in this project is most suitable track chassis.

## Step 3: Controller (MCU)

**3. Controller (MCU)**

You need Arduino-compatible controller

Controller must support 2 PWM and UART.

## Step 4: Bluetooth module

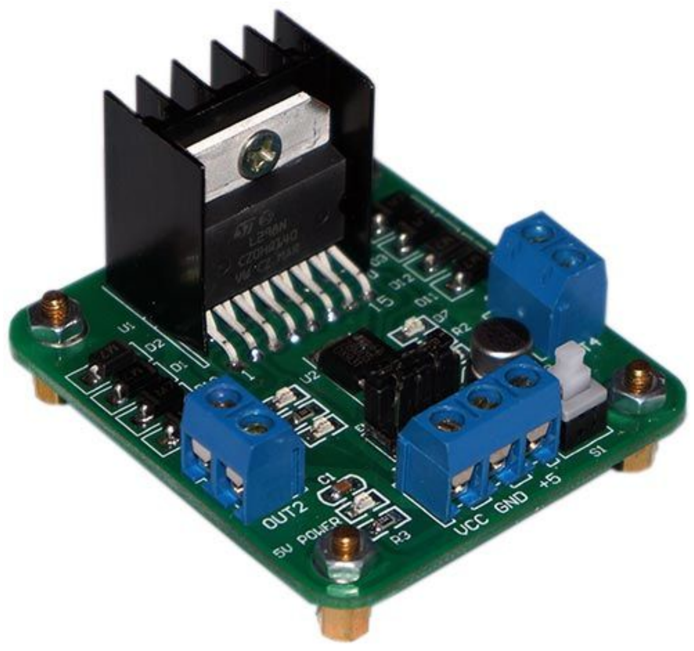
**4. Bluetooth module**

As a Bluetooth module uses cheap Chinese module HC-06

## Step 5: Motor Driver

**5. Motor Driver**

I used L298N Dual Bridge DC stepper Motor Driver module. It cost 4-5$ on eBay.



## Step 6: Other parts

**6. Other parts**

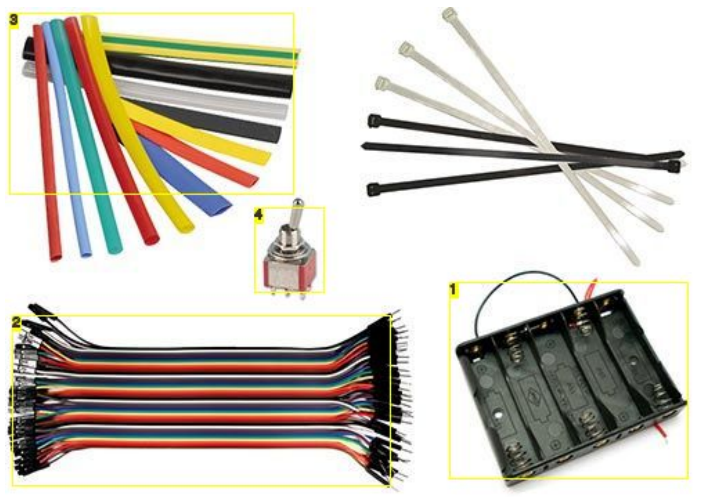


**Image Notes**
1. Battery holder
2. Wires
3. Heat-shrink tubing
4. switch

## Step 7: Theory

**Theory**

All calculations are performed in the Android-application, and immediately calculate the values 2‹2‹of the PWM for the left and right motor. Application has flexible settings, such as the range of the PWM, the sensitivity of tilt, the minimum threshold for the PWM and other. Example commands transmitted by Bluetooth:
L-255\rR-120\r
L - the command to the left engine, R - for the right
A dash means the motor rotation to move back
255 - PWM value (for Arduino is the maximum speed of rotation)
\r - end of command.
On this command RC car will move forward and slightly rotated to the right, as right engine rotates slowly left.

L255\rR-255\r
On this command the left engine will rotate back and forward right, forcing a car to rotate around its axis counterclockwise.

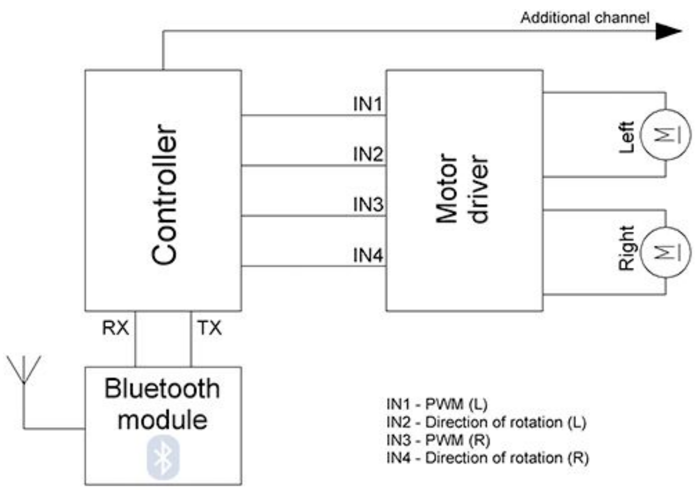
H1\r
Command is an additional channel to which you can connect for example lights, sound, etc.

Symbols command L, R and H can be defined in the settings of Android-applications.

In the MCU control program provides a timer that shuts off the engine if the last command was received more than n-seconds ago. The data are stored in the EEPROM memory of the controller and can be changed from Android device. The range of this setting is from 0.1 seconds to 99.9 seconds. This setting can be disabled. To work with EEPROM provides commands: Fr - reading values 2‹2‹and Fw - record values.

**Electronics**

Block diagram see on picture above



## Step 8: Android Application

As we can see, the Arduino connects to Bluetooth module and a motor driver with two or four connected motors.

**Android Application**

The application for Android was written in Eclipse IDE. All sources of the project and the project for Eclipse, you can download below. Android version on your device must be > 3.0.

The application contains several activity. Main activity is a home screen with buttons running different operating modes and settings

There are 3 control modes Bluetooth-car: from accelerometer, screen buttons and touch-control.

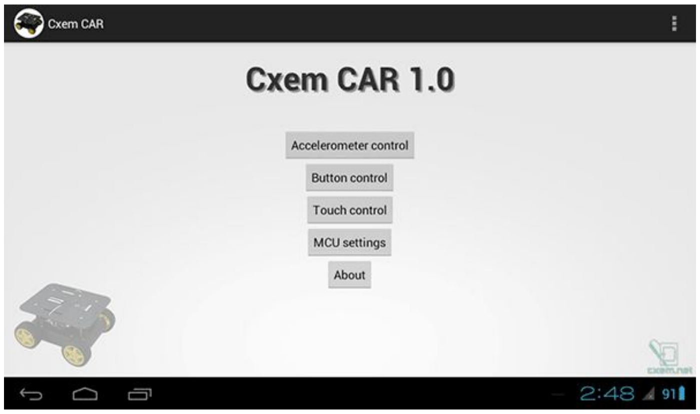
**Android application settings**

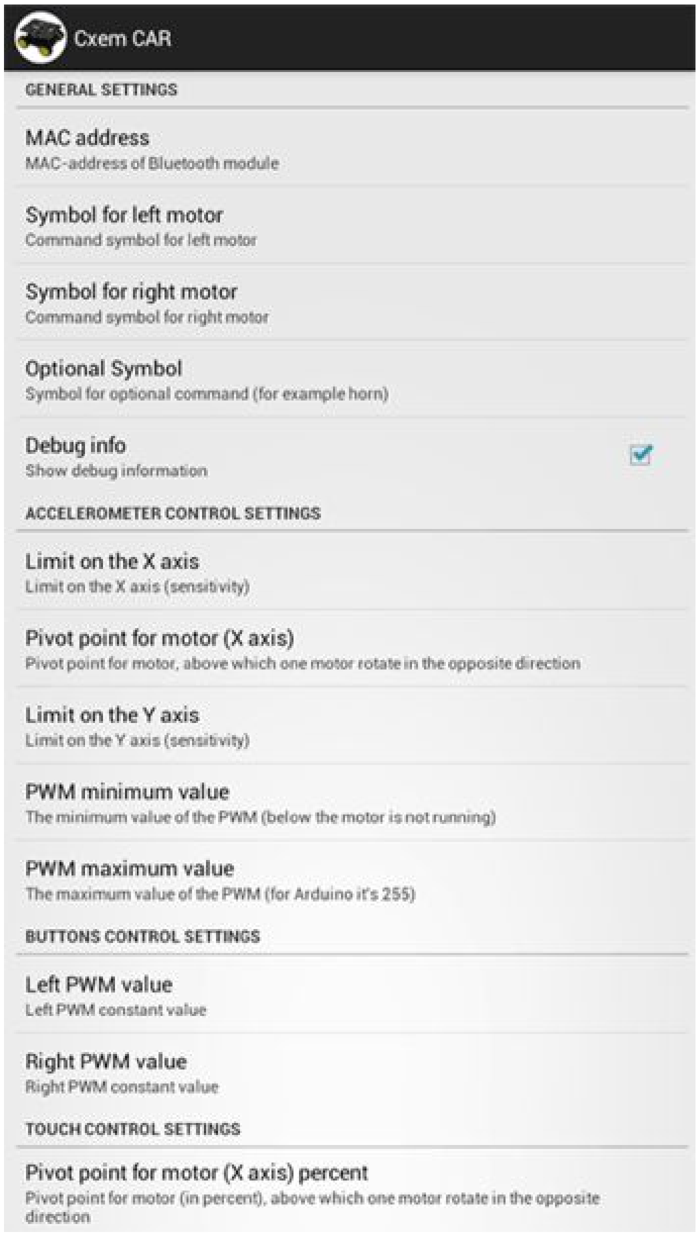
Screenshot of settings CxemCar Android application version 1.0

**MAC address**

To establish a connection with the RC Car's Bluetooth module, you must set MAC-address in the application settings. But first, you must configure the pair the devices on Android-device: open Settings -> Bluetooth and click "Search for devices". When the phone finds our Bluetooth-module, click them and enter password for pairing (usually "1234")

To know Bluetooth module MAC-address possible from any application, such as Bluetooth Terminal . To do this, click "Connect a device - Secure" and in the resulting window, click the button "Scan for devices". Software will be scans the Bluetooth devices and displays them MAC-address.




Cxem CAR

GENERAL SETTINGS

MAC address
MAC-address of Bluetooth module

Symbol for left motor
Command symbol for left motor

Symbol for right motor
Command symbol for right motor

Optional Symbol
Symbol for optional command (for example horn)

Debug info ✓
Show debug information

ACCELEROMETER CONTROL SETTINGS

Limit on the X axis
Limit on the X axis (sensitivity)

Pivot point for motor (X axis)
Pivot point for motor, above which one motor rotate in the opposite direction

Limit on the Y axis
Limit on the Y axis (sensitivity)

PWM minimum value
The minimum value of the PWM (below the motor is not running)

PWM maximum value
The maximum value of the PWM (for Arduino it's 255)

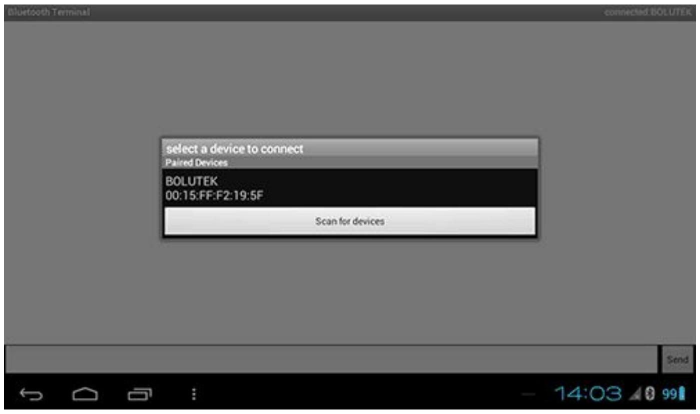
BUTTONS CONTROL SETTINGS

Left PWM value
Left PWM constant value

Right PWM value
Right PWM constant value

TOUCH CONTROL SETTINGS

Pivot point for motor (X axis) percent
Pivot point for motor (in percent), above which one motor rotate in the opposite direction


Bluetooth Terminal

connected:BOLUTEK

select a device to connect
Paired Devices
BOLUTEK
00:15:FF:F2:19:5F

## Step 9: Arduino RC car wiring

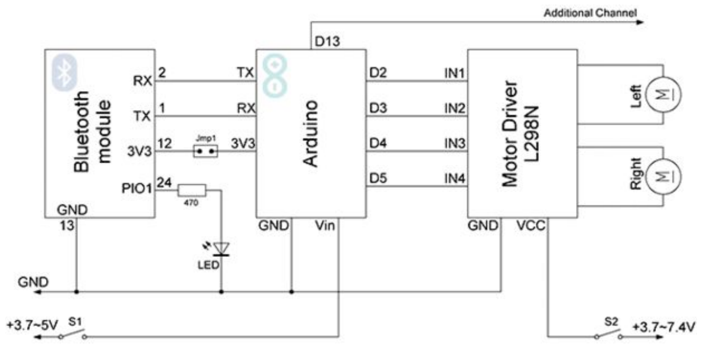
**Wiring diagram for Arduino controller**

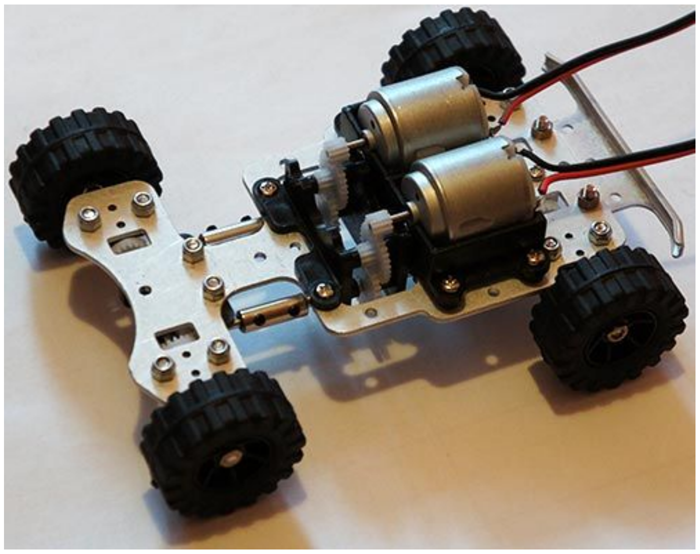
In the circuit I used a jumper (in the scheme Jmp1), because with a connected Bluetooth module is impossible be load sketch to the Arduino.

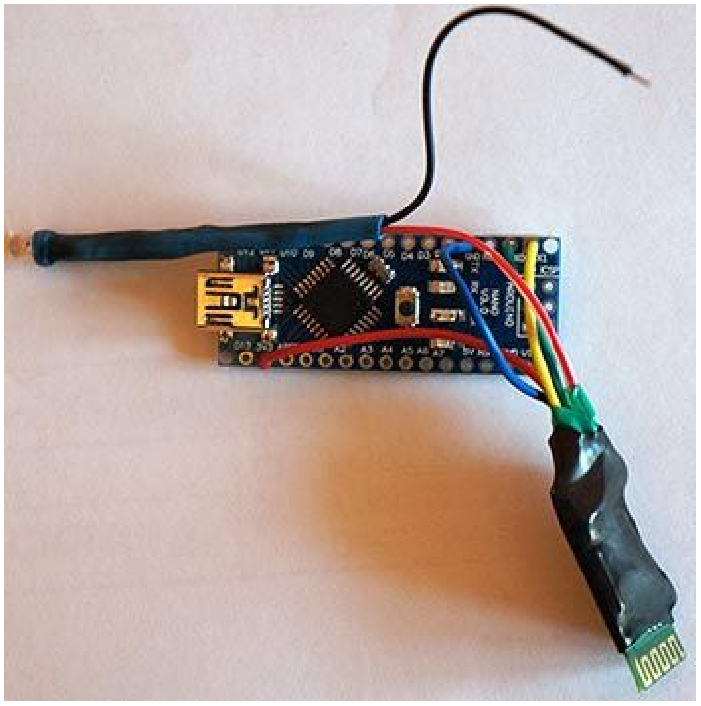
I soldered a Bluetooth-module to the Arduino and led status light. For communication between Arduino and Bluetooth, read this article: Arduino and Bluetooth. Module HC-06 placed in heat-shrink tube 10mm. Bluetooth-state LED with current limiting resistor were also placed in heat-shrink tube.

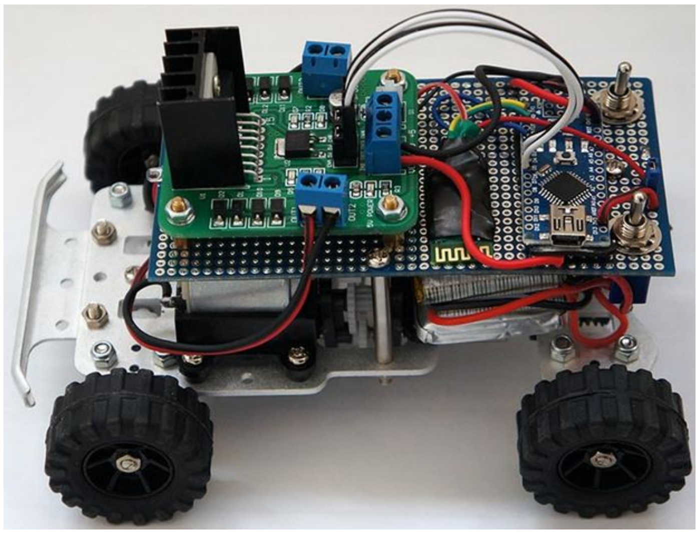
In the breadboard platform, I drilled a hole and secure motor driver L298N. Arduino board attached with double-sided tape

Between the car platform and breadboard I placed 3 Li-Po battery 3.7V 1100 mAh. Power to the controller and motors separately: Arduino powered by a 3.7 V battery, and the motors and driver L298N from two 3.7V batteries connected in series. There are two 2-position power switch - one position is the power from the batteries to consumers, in the other position to charging terminals.









## Step 10: Software

The program was written in Arduino IDE 1.01.

```
/ CxemCAR 1.0 (06.01.2013)
// Project Page: http://english.cxem.net/mcu/mcu3.php

#include "EEPROM.h"

#define D1 2 // direction of motor rotation 1
#define M1 3 // PWM left motor
#define D2 4 // direction of motor rotation 2
#define M2 5 // PWM right motor
#define HORN 13 // additional channel 1
//#define autoOFF 2500 // milliseconds after which the robot stops when the connection

#define cmdL 'L' // UART-command for left motor
#define cmdR 'R' // UART-command for right motor
#define cmdH 'H' // UART-command for additional channel (for example Horn)
#define cmdF 'F' // UART-command for EEPROM operation
#define cmdr 'r' // UART-command for EEPROM operation (read)
#define cmdw 'w' // UART-command for EEPROM operation (write)

char incomingByte; // incoming data

char L_Data[4]; // array data for left motor
byte L_index = 0; // index of array L
char R_Data[4]; // array data for right motor
byte R_index = 0; // index of array R
char H_Data[1]; // array data for additional channel
byte H_index = 0; // index of array H
char F_Data[8]; // array data for EEPROM
byte F_index = 0; // index of array F
char command; // command

unsigned long currentTime, lastTimeCommand, autoOFF;

void setup() {
Serial.begin(9600); // initialization UART
pinMode(HORN, OUTPUT); // additional channel
pinMode(D1, OUTPUT); // output for motor rotation
pinMode(D2, OUTPUT); // output for motor rotation
/*EEPROM.write(0,255);
EEPROM.write(1,255);
EEPROM.write(2,255);
EEPROM.write(3,255);*/
timer_init(); // initialization software timer
}

void timer_init() {
uint8_t sw_autoOFF = EEPROM.read(0); // read EEPROM "is activated or not stopping the car when losing connection"
if(sw_autoOFF == '1'){ // if activated
char var_Data[3];
var_Data[0] = EEPROM.read(1);
var_Data[1] = EEPROM.read(2);
var_Data[2] = EEPROM.read(3);
autoOFF = atoi(var_Data)*100; // variable autoOFF ms
}
else if(sw_autoOFF == '0'){
autoOFF = 999999;
}
else if(sw_autoOFF == 255){
autoOFF = 2500; // if the EEPROM is blank, dafault value is 2.5 sec
}
currentTime = millis(); // read the time elapsed since application start
}
```

```
void loop() {
if (Serial.available() > 0) { // if received UART data
incomingByte = Serial.read(); // raed byte
if(incomingByte == cmdL) { // if received data for left motor L
command = cmdL; // current command
memset(L_Data,0,sizeof(L_Data)); // clear array
L_index = 0; // resetting array index
}
else if(incomingByte == cmdR) { // if received data for left motor R
command = cmdR;
memset(R_Data,0,sizeof(R_Data));
R_index = 0;
}
else if(incomingByte == cmdH) { // if received data for additional channel
command = cmdH;
memset(H_Data,0,sizeof(H_Data));
H_index = 0;
}
else if(incomingByte == cmdF) { // if received data for EEPROM op
command = cmdF;
memset(F_Data,0,sizeof(F_Data));
F_index = 0;
}
else if(incomingByte == '\r') command = 'e'; // end of line
else if(incomingByte == '\t') command = 't'; // end of line for EEPROM op

if(command == cmdL && incomingByte != cmdL){
L_Data[L_index] = incomingByte; // store each byte in the array
L_index++; // increment array index
}
else if(command == cmdR && incomingByte != cmdR){
R_Data[R_index] = incomingByte;
R_index++;
}
else if(command == cmdH && incomingByte != cmdH){
H_Data[H_index] = incomingByte;
H_index++;
}
else if(command == cmdF && incomingByte != cmdF){
F_Data[F_index] = incomingByte;
F_index++;
}
else if(command == 'e'){ // if we take the line end
Control4WD(atoi(L_Data),atoi(R_Data),atoi(H_Data));
delay(10);
}
else if(command == 't'){ // if we take the EEPROM line end
Flash_Op(F_Data[0],F_Data[1],F_Data[2],F_Data[3],F_Data[4]);
}
lastTimeCommand = millis(); // read the time elapsed since application start
}
if(millis() >= (lastTimeCommand + autoOFF)){ // compare the current timer with variable lastTimeCommand + autoOFF
Control4WD(0,0,0); // stop the car
}
}

void Control4WD(int mLeft, int mRight, uint8_t Horn){

bool directionL, directionR; // direction of motor rotation L298N
byte valueL, valueR; // PWM M1, M2 (0-255)

if(mLeft > 0){
valueL = mLeft;
directionL = 0;
}
else if(mLeft < 0){
valueL = 255 - abs(mLeft);
directionL = 1;
}
else {
directionL = 0;
valueL = 0;
}

if(mRight > 0){
valueR = mRight;
directionR = 0;
}
else if(mRight < 0){
valueR = 255 - abs(mRight);
directionR = 1;
}
else {
directionR = 0;
valueR = 0;
}

analogWrite(M1, valueL); // set speed for left motor
analogWrite(M2, valueR); // set speed for right motor
digitalWrite(D1, directionL); // set direction of left motor rotation
digitalWrite(D2, directionR); // set direction of right motor rotation

digitalWrite(HORN, Horn); // additional channel
}

void Flash_Op(char FCMD, uint8_t z1, uint8_t z2, uint8_t z3, uint8_t z4){

if(FCMD == cmdr){ // if EEPROM data read command
Serial.print("FData:"); // send EEPROM data
Serial.write(EEPROM.read(0)); // read value from the memory with 0 address and print it to UART
Serial.write(EEPROM.read(1));
Serial.write(EEPROM.read(2));
Serial.write(EEPROM.read(3));
Serial.print("\r\n"); // mark the end of the transmission of data EEPROM
}
else if(FCMD == cmdw){ // if EEPROM data write command
EEPROM.write(0,z1); // z1 record to a memory with 0 address
EEPROM.write(1,z2);
EEPROM.write(2,z3);
EEPROM.write(3,z4);
timer_init(); // reinitialize the timer
Serial.print("FWOK\r\n"); // send a message that the data is successfully written to EEPROM
}
```

The code uses a library to work with EEPROM AVR-memory. Arduino board by USART from the Bluetooth module receives data ready for the left and right engine. All basic calculations are performed in the Android application.

You can download the source code for the Arduino, and the project for Eclipse
Download APK application for Android-device

P.S. Sorry for my bad english
P.S.S. Soon to be the same project for STM32

## File Downloads

**Arduino_BL_4WD_en.rar** (1 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Arduino_BL_4WD_en.rar']

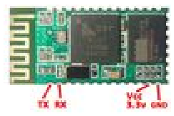
**BL_4WD.rar** (1 MB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'BL_4WD.rar']

## Related Instructables

**Bluetooth mobile phone accessory for Missed calls and SMS** by zmashiah

**Android talks to Arduino** by circuit_breaker

**Control Keyboard & Mouse w/ Android app via Arduino** by plowdk51


**Cheap 2-Way Bluetooth Connection Between Arduino and PC** by techbitar

**Androino! Control an Arduino from your Android device using a cheap bluetooth module.** by metanurb

**Super Nintendo on Android with original controller** (video) by bsoares