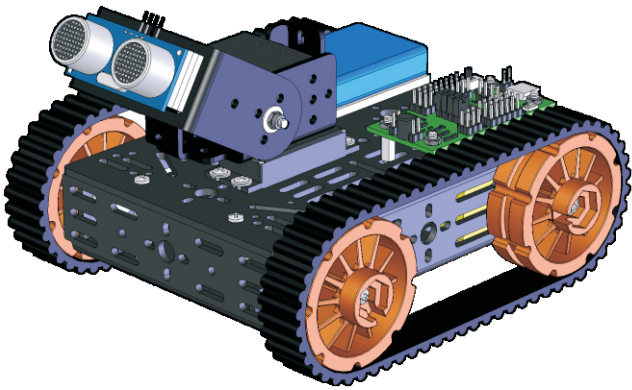
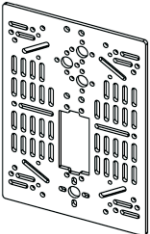
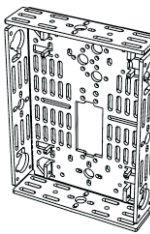
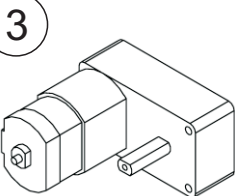
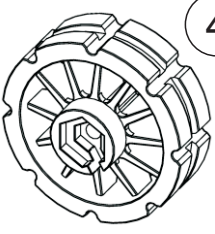
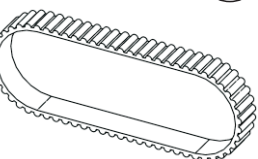
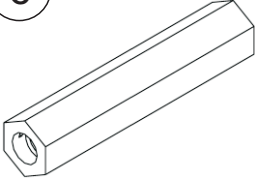
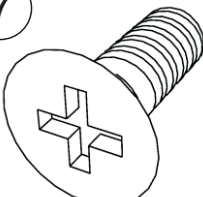
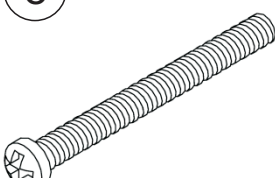
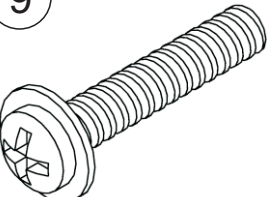
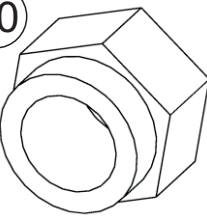
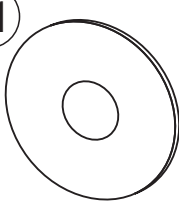
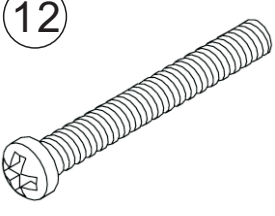
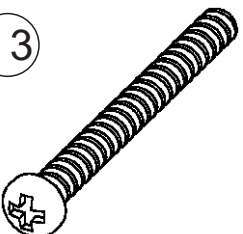
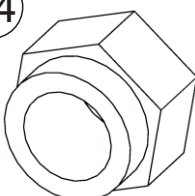
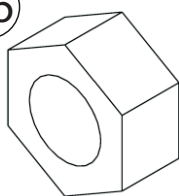
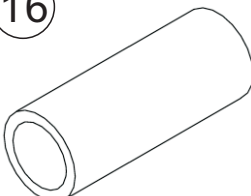
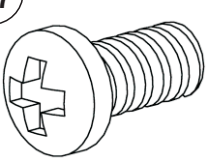
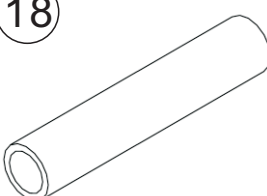
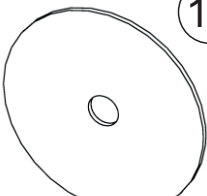
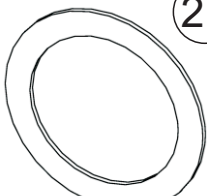
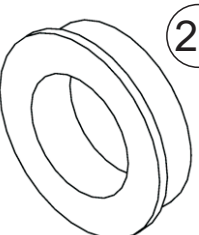
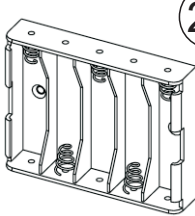
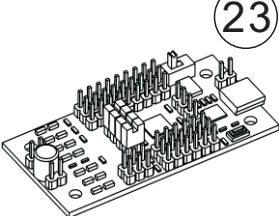
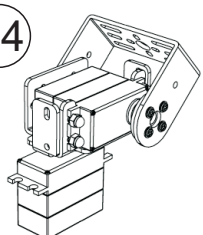
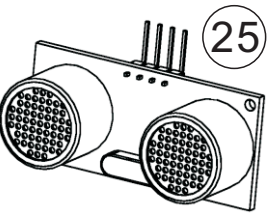
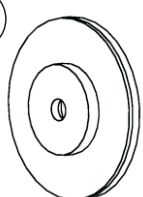
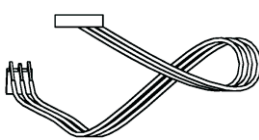
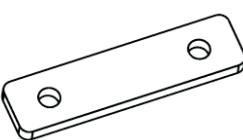
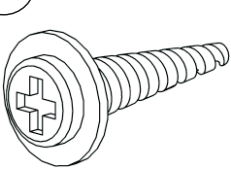
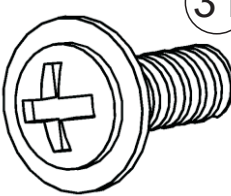
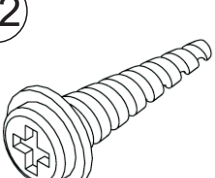
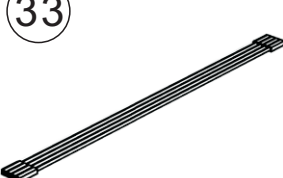
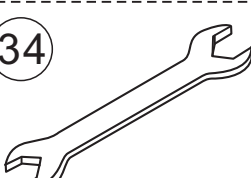
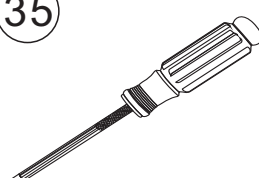


DG012-EV
Multi chassis_tank
(explore version)



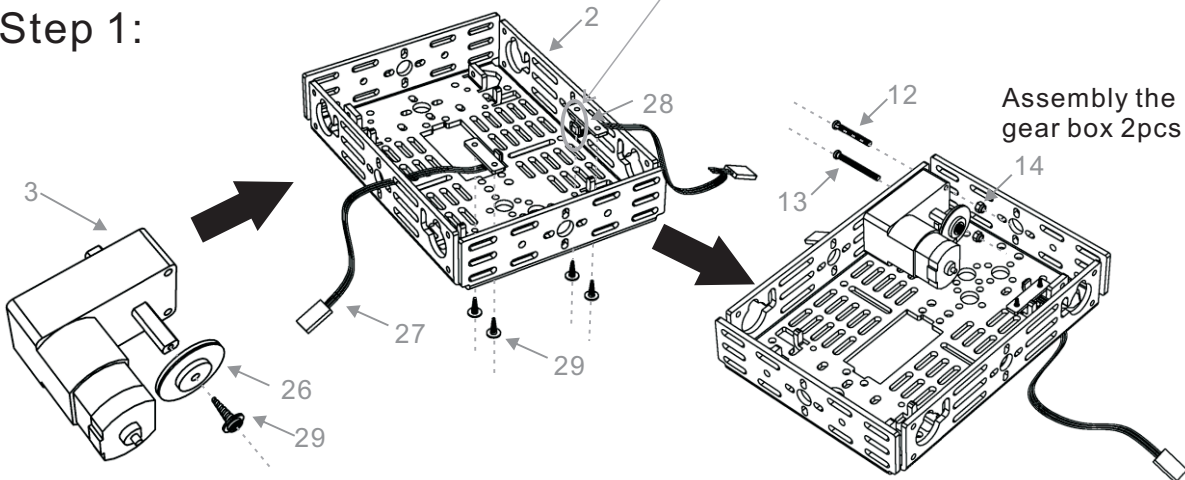
Parts list:

 1 Aluminum chassis top 1pc	 2 Aluminum chassis bottom 1pc	 3 DC motor 2pcs	 4 Track wheel 4pcs
 5 Rubber belt 2pcs	 6 Distance holder M3*25 4pcs	 7 M3*8 flat-head screw 6pcs	 8 M3*30 screw 2pcs
 9 M3*16 screw with flange 2pcs	 10 M3 nylon nut 2pcs	 11 M3 washer (thin) 6pcs	 12 M2.5*20 screw 2pcs
 13 M2.5*25 screw 2pcs	 14 M2.5 nylon nut 4pcs	 15 M3 nut 2pcs	 16 Phi 3*10 collar 2pcs
 17 M3*8 screw 8pcs	 18 Phi 3*20 collar 2pcs	 19 Washer(3*20) 4pcs	 20 Washer(15*20) 4pcs
 21 10*15*4 flange bearings 4pcs	 22 5*AA battery holder 1pc	 23 Mini driver 1pc	 24 Pan tile kit 1pc
 25 Ultrasonic sensor 1pc	 26 Hall sensor encoder kit 2pcs	 27 Encoder sensor 2pcs	 28 Fibreboard 2pcs
 29 M2.3*8 self-tapping screw with flange 6pcs	 30 ISP programmer 1pc	 31 M3*8 screw with flange 4pcs	
 32 M2.3*10 self-tapping screw with flange 2pcs	 33 4 Pin cable 1pc	 34 Spanner (M2.5 & M4) 1pc (M2 & M3) 1pc	 35 Screw driver 1pc
 36 USB cable 1pc			

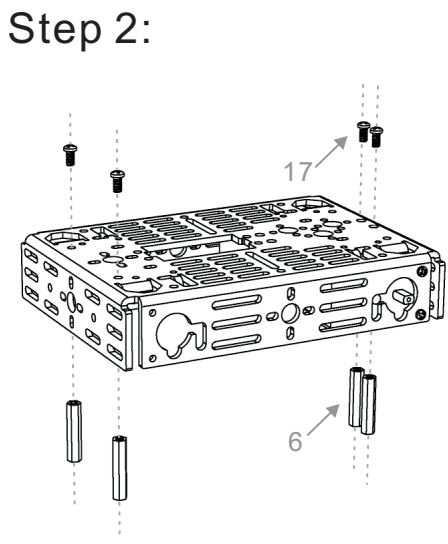
Assembly Instructions:

Please assembly the hall sensor with a marking "20L" facing towards the magnet.

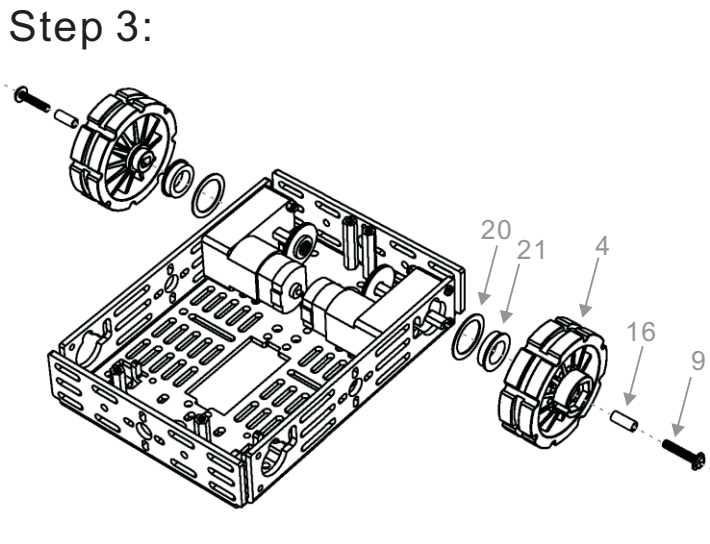
Step 1:



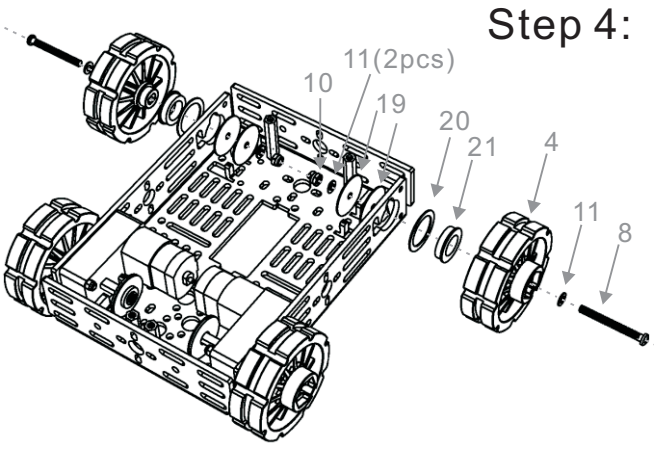
Step 2:



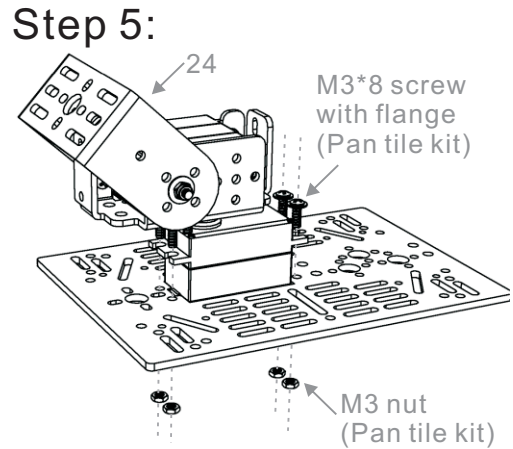
Step 3:



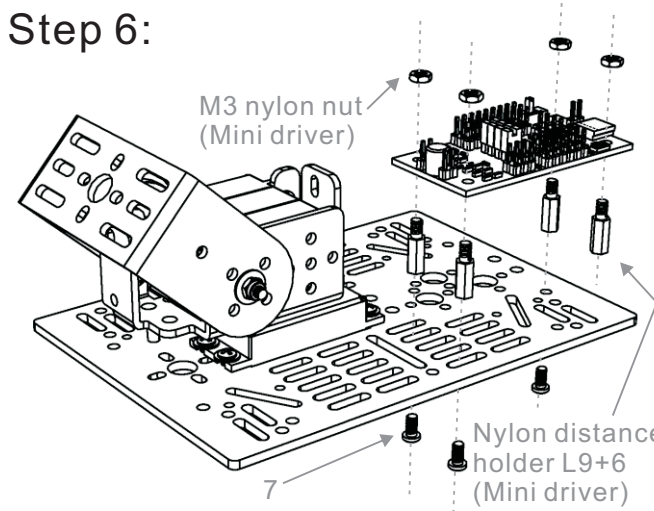
Step 4:



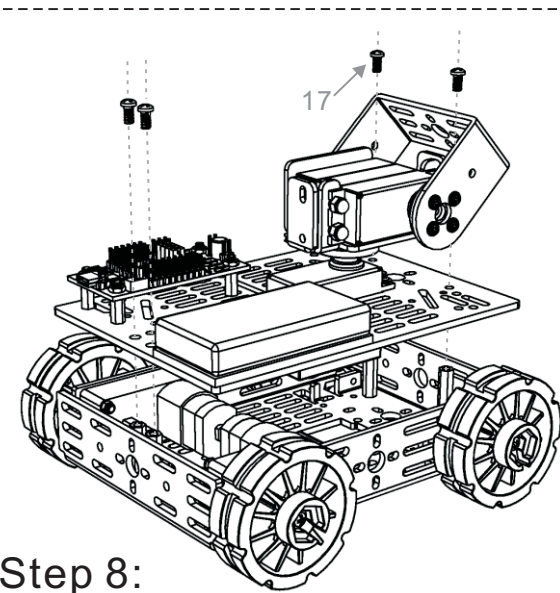
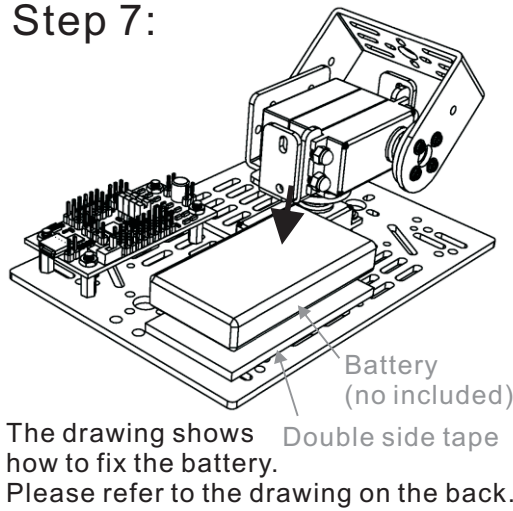
Step 5:



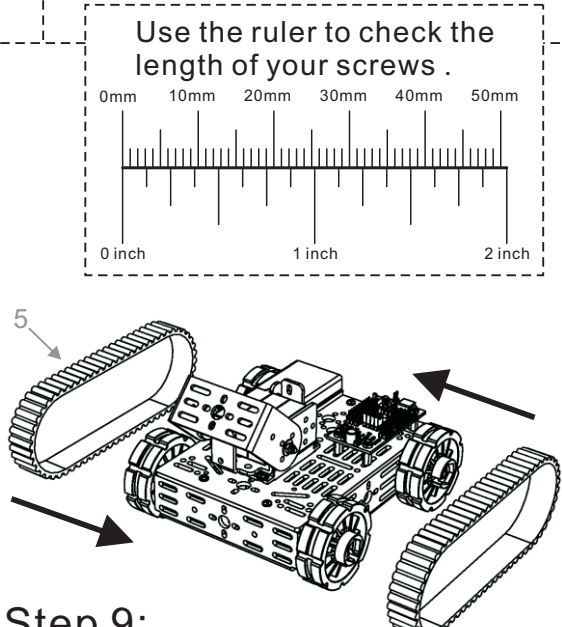
Step 6:



Step 7:

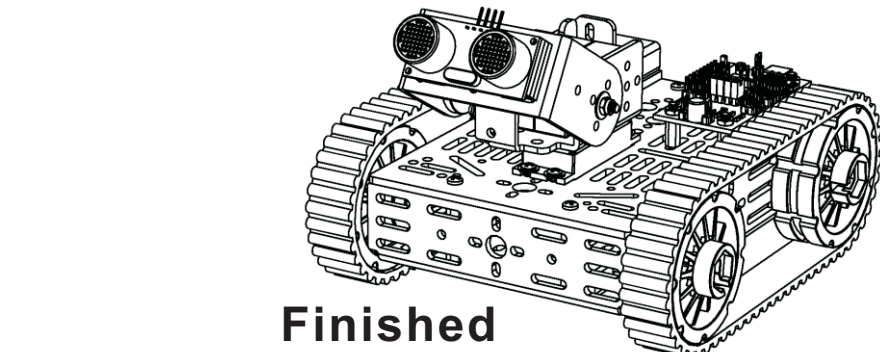
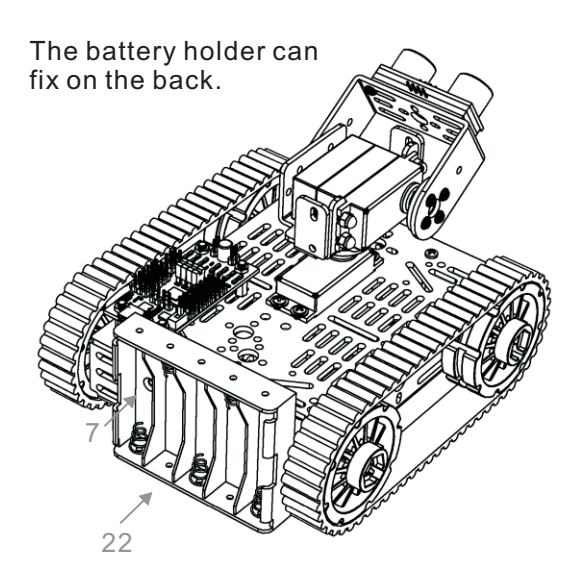
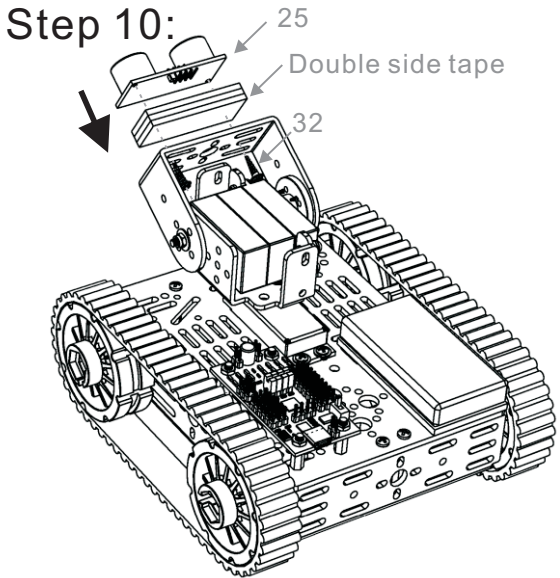


Step 8:



Step 9:

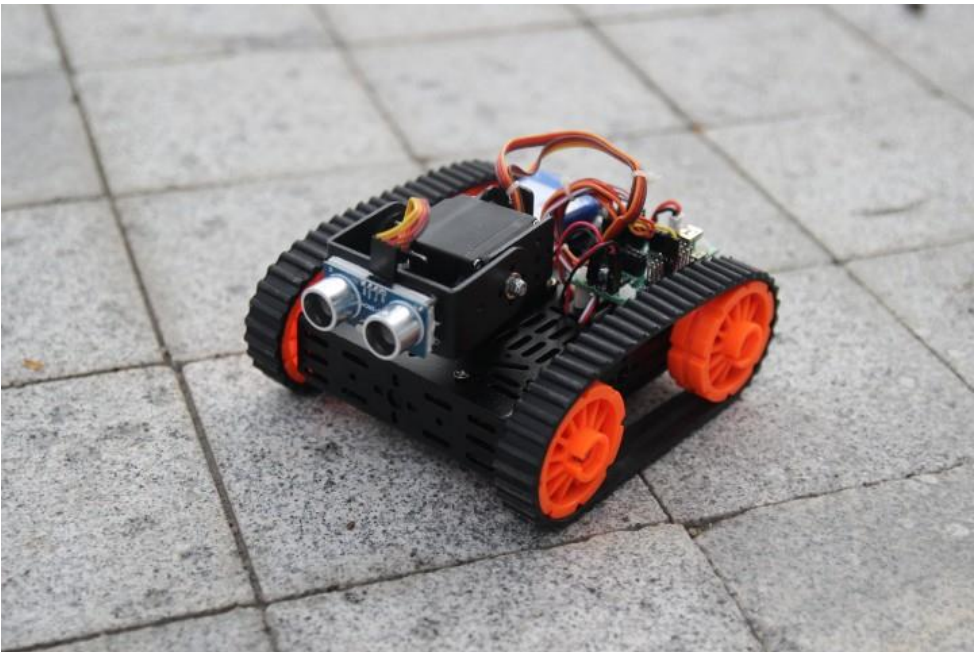
Step 10:



Finished

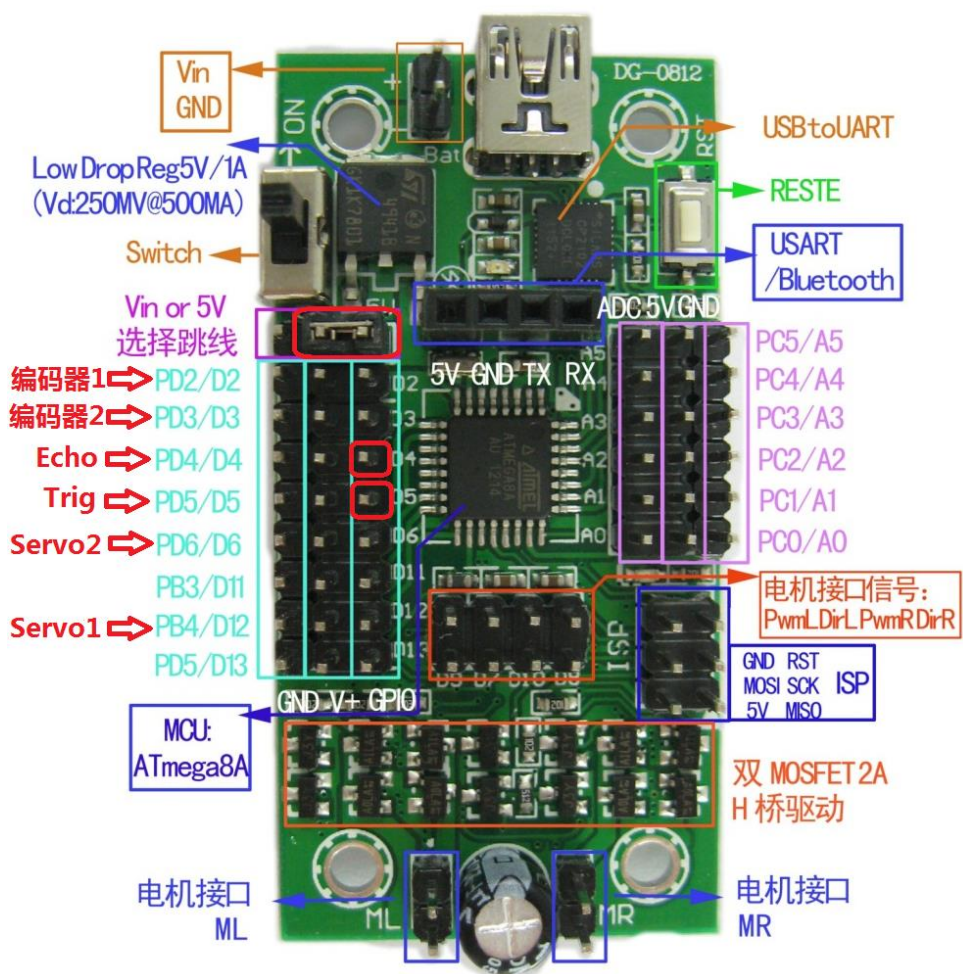
Multi Chassis-Tank

(Explore version)



The functions include: ultrasonic sensor detection, servos control, mini driver control, motor driving, and simple encoder measurement.

programming and wire connection:



Ultrasonic sensor and encoder must supply 5V, and the servos can work at 5V, so please plug the jumper at 5V.

Do not connect any powerful device on the analog side while you plug USB only.



1. Ultrasonic sensor detection

```
//Ultrasonic module pins  
int inputU1=4;    // receive  ECHO -> D4  
int outputU1=5;   // send   TRIG -> D5
```



Pay attention to the pin order: GND,ECHO,TRIG,VCC

GND : negative

ECHO:data reading

TRIG:control the signal transmitting

VCC:positive,5V

Turn on the ultrasonic sensor and receive the data:

The image shows a screenshot of the Arduino IDE interface. The title bar reads "Ultrasonic | Arduino 1.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for saving, running, uploading, and downloading. The main text area contains the following code:

```
//Ultrasonic module pins
int inputU1=4;    // receive  ECHO -> D4
int outputU1=5;   // send  TRIG -> D5

unsigned int distance = 0;

void setup()
{
    pinMode(inputU1, INPUT);    // make the Echo pins an input
    pinMode(outputU1, OUTPUT);  // make the Trig pins an output
    Serial.begin(9600);         // initialize serial communication at 9600 bits per seco
}

void loop()
{
    distance = readUltrasonic(inputU1,outputU1);
    Serial.println(distance);
}

/** read the date of ultrasonic
 * remember to deal with no reflect condition
 * once receive nothing it will return the length as 5m.
 * @param inputPin  receiver pulse  ECHO
 *      outputPin output pulse  TRIG
 * @return distance between the objects
 */
int readUltrasonic(int inputPin, int outputPin)
{
    unsigned int mesure = 0;
    digitalWrite(outputPin, LOW);
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW);

    int distance = pulseIn(inputPin, HIGH, 10000); //read pulse last time in 10000 mircosec
    return distance/58;                          //caculate the distance.0~172
}
```

The status bar at the bottom indicates "7" and "Arduino NG or older w/ ATmega8 on COM8".

2. There are two different methods for servos and motor

controlling.



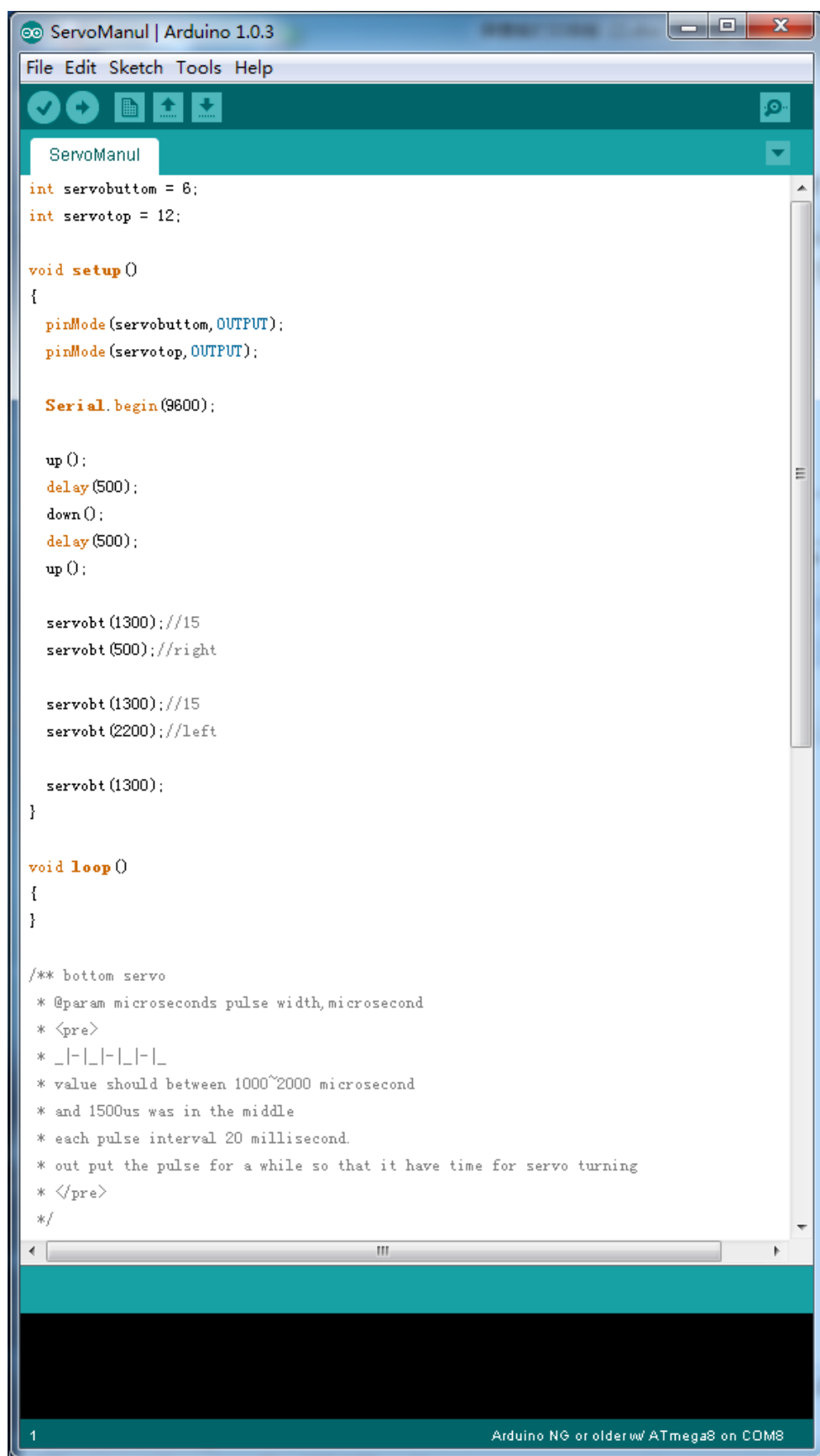
Servo 1 connect to D12

Servo 2 connect to D6

Method 1: Manually output the pulse to control the servos instead of using <Servo.h> in Arudino official library.

servo control routine:

```
void servobt(unsigned int degreed)
{
    for(int i=0;i<30;i++)
    {
        digitalWrite(servobuttom,1);
        delayMicroseconds(degreed);
        digitalWrite(servobuttom,0);
        delay(20);
    }
}
```

The image shows a screenshot of the Arduino IDE interface. The title bar at the top reads "ServoManul | Arduino 1.0.3". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Underneath the menu bar is a toolbar with icons for checking, running, serial monitor, and uploading/downloading. The main text area contains the following code:

```
int servobottom = 6;
int servotop = 12;

void setup()
{
  pinMode(servobottom, OUTPUT);
  pinMode(servotop, OUTPUT);

  Serial.begin(9600);

  up();
  delay(500);
  down();
  delay(500);
  up();

  servobt(1300); //15
  servobt(500); //right

  servobt(1300); //15
  servobt(2200); //left

  servobt(1300);
}

void loop()
{
}

/** bottom servo
 * @param microseconds pulse width, microsecond
 * <pre>
 * _|-|-|-|-|_
 * value should between 1000~2000 microsecond
 * and 1500us was in the middle
 * each pulse interval 20 millisecond.
 * out put the pulse for a while so that it have time for servo turning
 * </pre>
 */
```

The status bar at the bottom indicates "1" on the left and "Arduino NG or older w/ ATmega8 on COM8" on the right.

‘degreed’ just like pulse time, and the value between

1000 and 2000 as usual. And 1500 was the middle place. By controlling the pulse width to control steering gear rotation degree

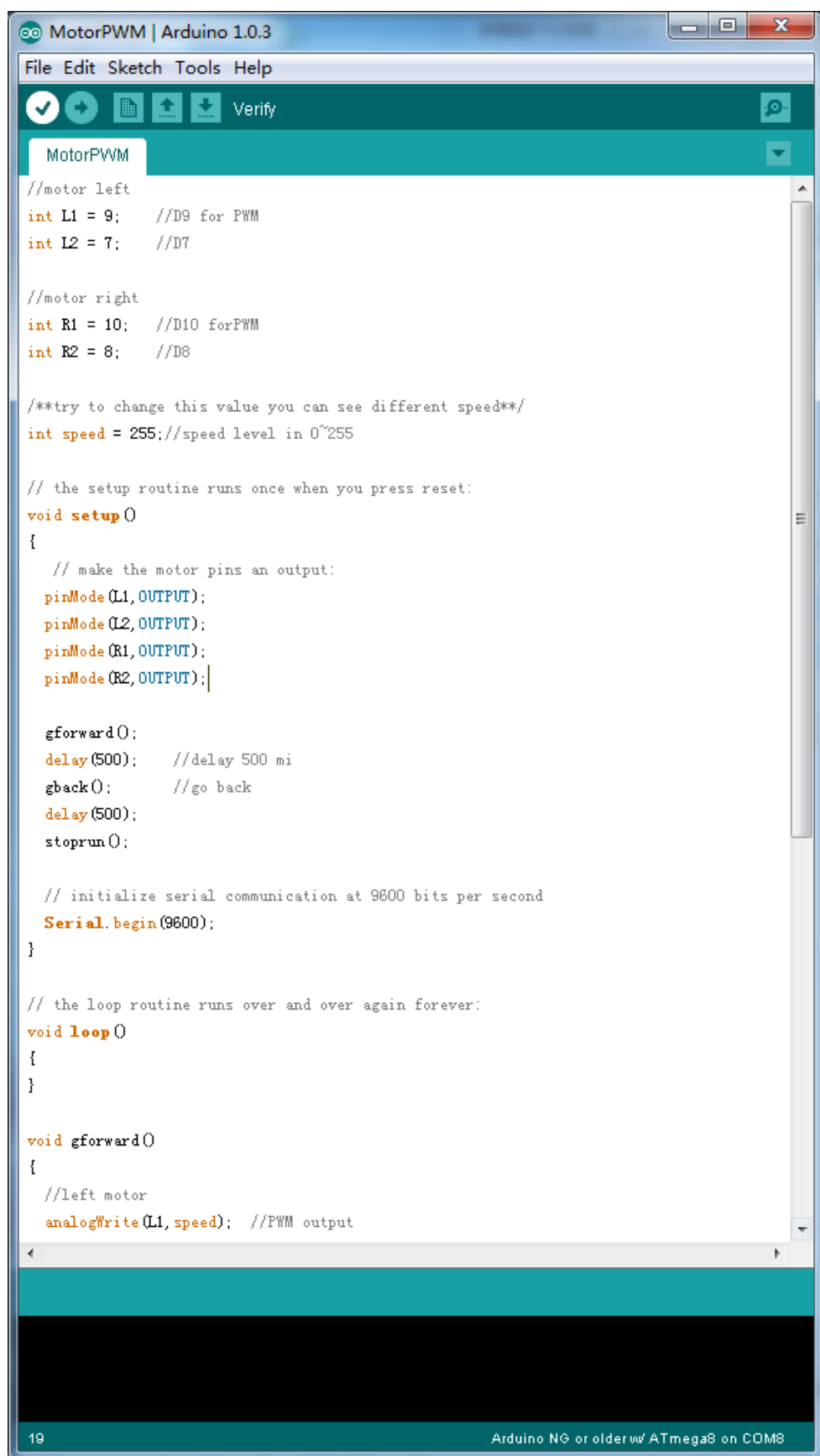
Motor control with PWM (Pulse-Width Modulation):

```
//motor left
int L1 = 9;    //D9 for PWM
int L2 = 7;    //D7

//motor right
int R1 = 10;   //D10 for PWM
int R2 = 8;    //D8
```

sample:

```
void gforward()
{
    //left motor
    analogWrite(L1, speed); //PWM output
    digitalWrite(L2, LOW);
    //right motor
    analogWrite(R1, speed); //PWM output
    digitalWrite(R2, LOW);
}
```



```
MotorPWM | Arduino 1.0.3
File Edit Sketch Tools Help
[Icons] Verify
MotorPWM
//motor left
int L1 = 9;    //D9 for PWM
int L2 = 7;    //D7

//motor right
int R1 = 10;   //D10 for PWM
int R2 = 8;    //D8

/**try to change this value you can see different speed**/
int speed = 255; //speed level in 0~255

// the setup routine runs once when you press reset:
void setup()
{
    // make the motor pins an output:
    pinMode(L1, OUTPUT);
    pinMode(L2, OUTPUT);
    pinMode(R1, OUTPUT);
    pinMode(R2, OUTPUT);

    gforward();
    delay(500);    //delay 500 ms
    gback();       //go back
    delay(500);
    stoprun();

    // initialize serial communication at 9600 bits per second
    Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop()
{
}

void gforward()
{
    //left motor
    analogWrite(L1, speed); //PWM output
}
```

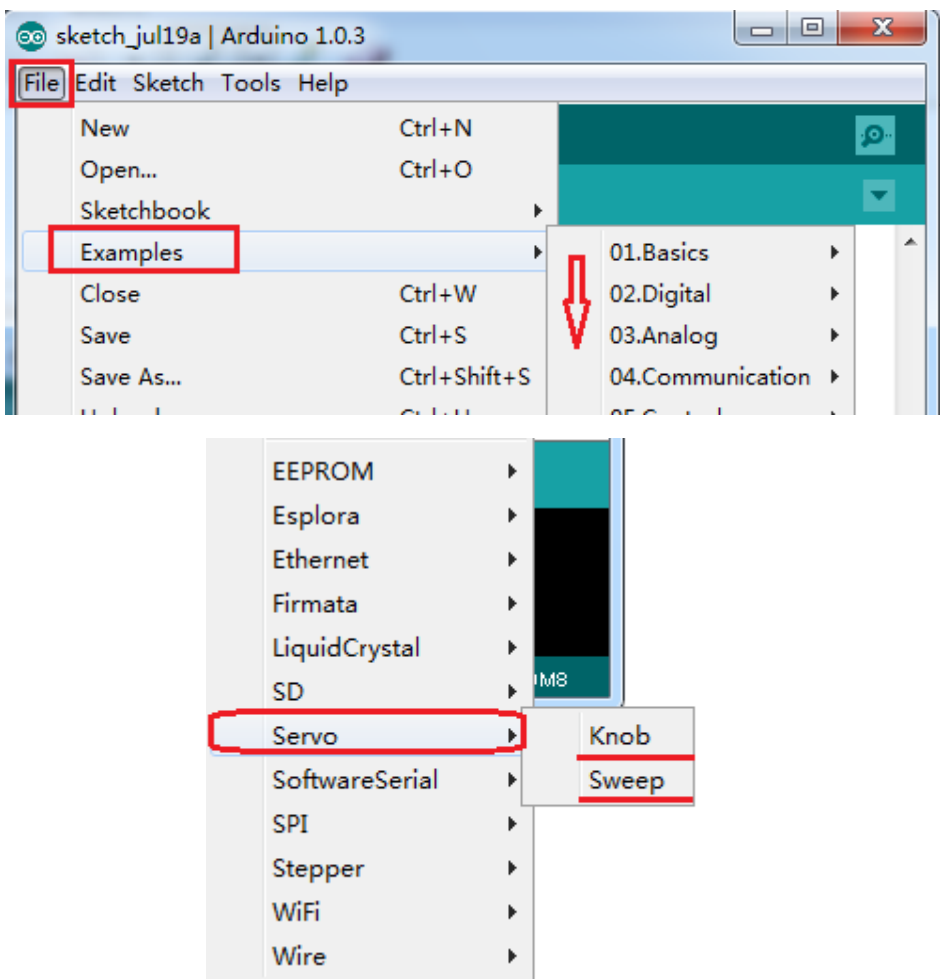
19 Arduino NG or older w/ ATmega8 on COM8

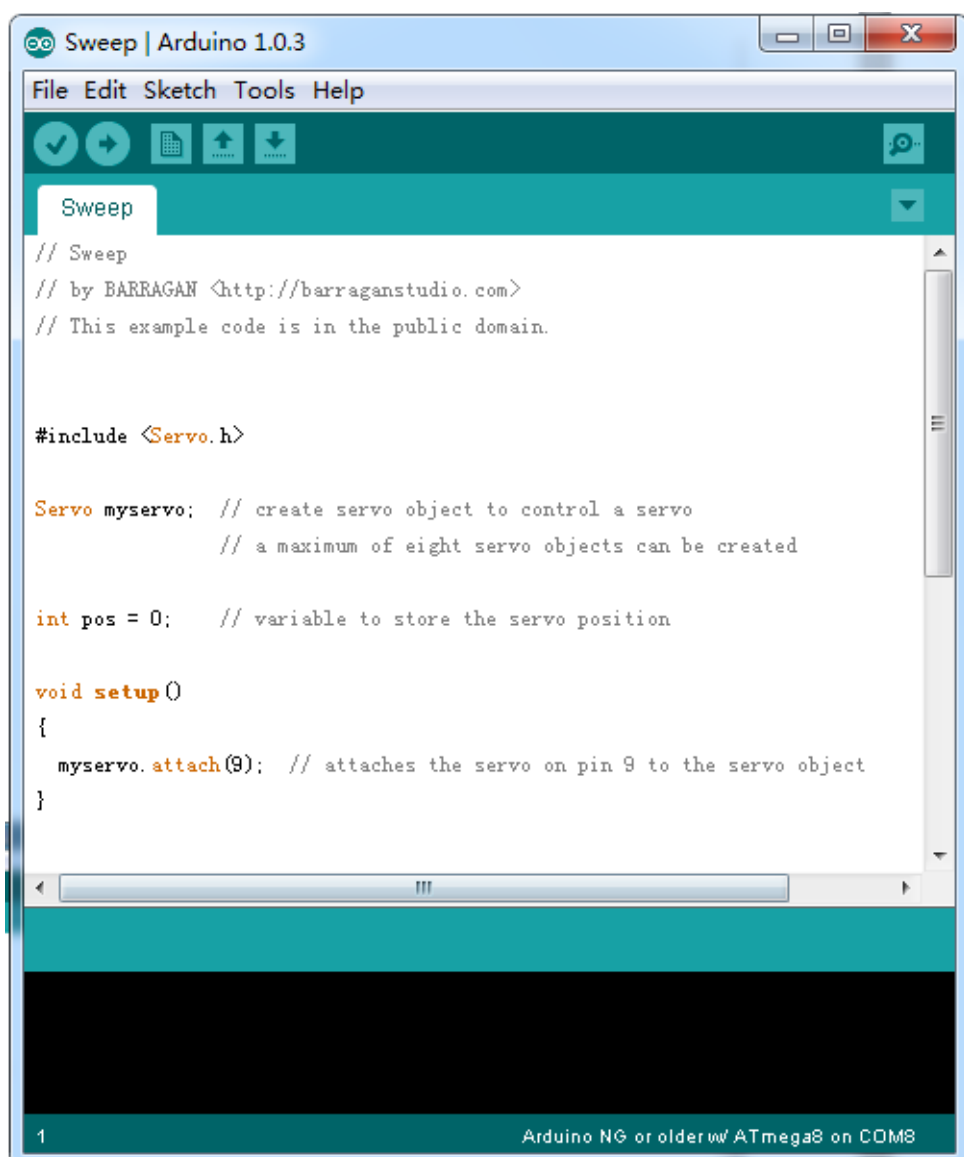
[Attention: it can not output the PWM to adjust motor

speed if using the Arduino library controlling servos by mini driver controller.]

Method 2: use the Arduino library controlling servos

For the servo control program, please refer to the Arduino program sample: File -> Examples -> Servo (or the manually control servo program)





The PWM output power function in motor control program can not be used (can not adjust the speed):

how to use the function, please refer to the Arduino reference:

Syntax

`analogWrite(pin, value)`

Parameters

`pin`: the pin to write to.

`value`: the duty cycle: between 0 (always off) and 255 (always on).

Change the `analogWrite(pin,value)` to `digitalWrite(pin,value)`.

example:

```
void gforward()
{
    //left motor
    digitalWrite(L1, HIGH);
    digitalWrite(L2, LOW);
    //right motor
    digitalWrite(R1, HIGH);
    digitalWrite(R2, LOW);
}
```



[Attention: With minidriver, once you need PWM to adjust

the motor speed, just use method 1 otherwise use Arduino library to control servos.]

Encoder: [voltage suitable in 5V]

Initialize and banding pins to the external interrupt 0 and external interrupt 1

external interrupt 0 was on pin D2

external interrupt 1 was on pin D3

[Reference](#) [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

attachInterrupt()

Description

Specifies a function to call when an external interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The Arduino Mega has an additional four: numbers 2 (pin 21), 3 (pin 20), 4 (pin 19), and 5 (pin 18).

Syntax

```
attachInterrupt(interrupt, function, mode)
```

Parameters

interrupt: the number of the interrupt (*int*)

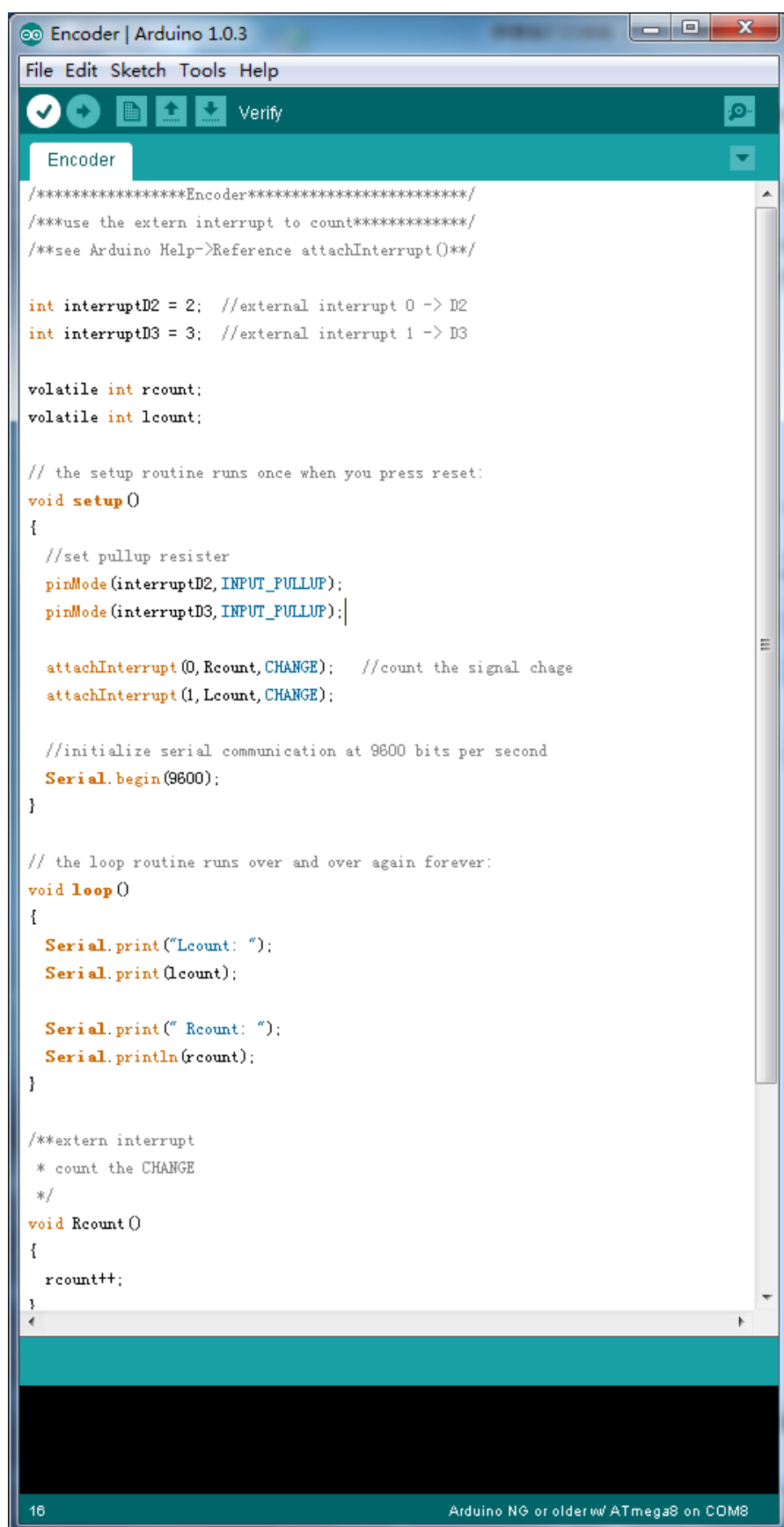
function: the function to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an *interrupt service routine*.

mode defines when the interrupt should be triggered. Four constants are predefined as valid values:

- **LOW** to trigger the interrupt whenever the pin is low,
- **CHANGE** to trigger the interrupt whenever the pin changes value
- **RISING** to trigger when the pin goes from low to high,
- **FALLING** for when the pin goes from high to low.

Returns

none



```
Encoder | Arduino 1.0.3
File Edit Sketch Tools Help
Encoder
/*****Encoder*****/
/**use the extern interrupt to count*****/
/**see Arduino Help->Reference attachInterrupt()**/

int interruptD2 = 2; //external interrupt 0 -> D2
int interruptD3 = 3; //external interrupt 1 -> D3

volatile int rcount;
volatile int lcount;

// the setup routine runs once when you press reset:
void setup()
{
  //set pullup resister
  pinMode(interruptD2, INPUT_PULLUP);
  pinMode(interruptD3, INPUT_PULLUP);

  attachInterrupt(0, Rcount, CHANGE); //count the signal chage
  attachInterrupt(1, Lcount, CHANGE);

  //initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop()
{
  Serial.print("Lcount: ");
  Serial.print(lcount);

  Serial.print(" Rcount: ");
  Serial.println(rcount);
}

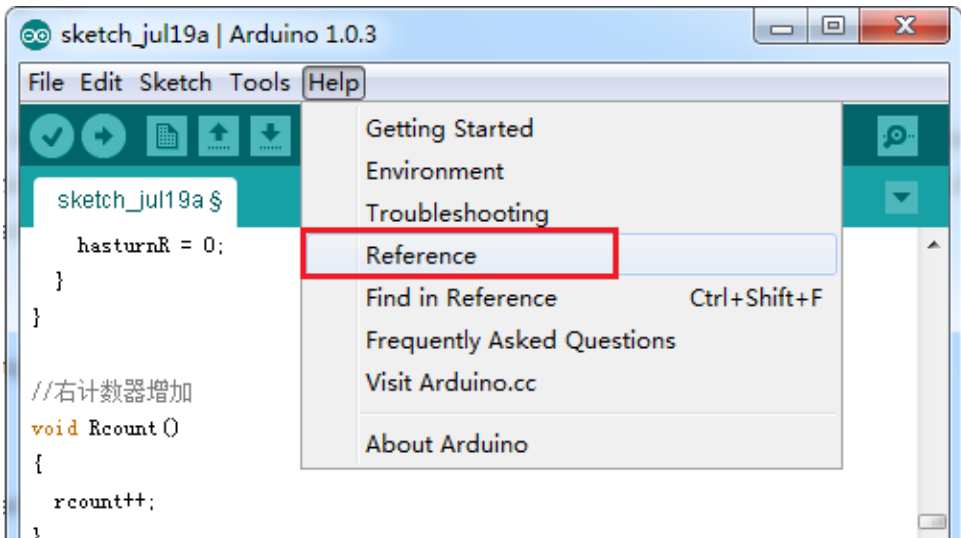
/**extern interrupt
 * count the CHANGE
 */
void Rcount()
{
  rcount++;
}
```

16 Arduino NG or older w/ ATmega8 on COM8

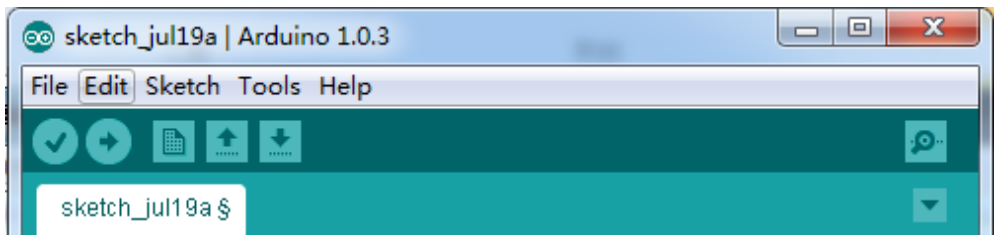
For details, please refer to the samples code in CD and

Arduino Reference.

How to use Arduino (a good help for Arduino learning):



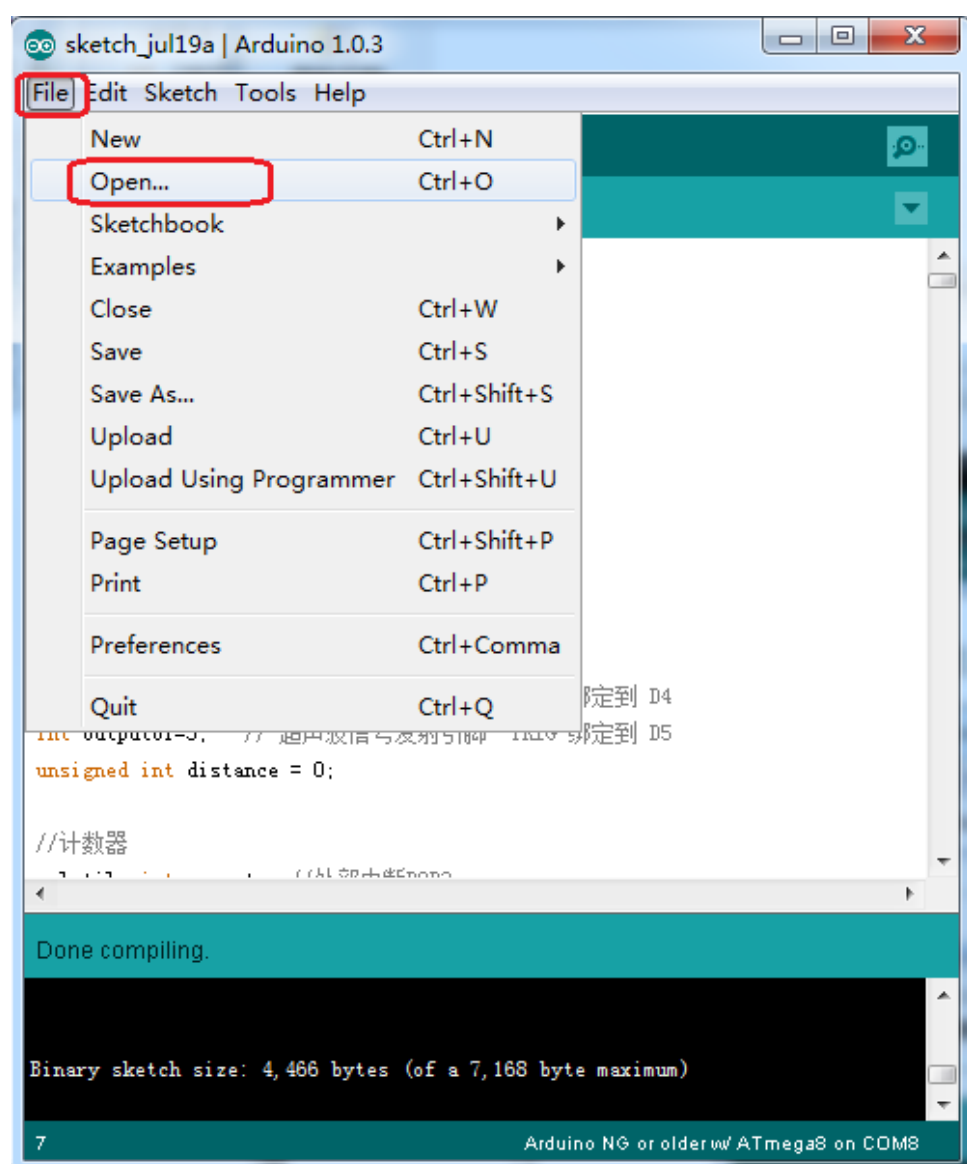
Software interface :

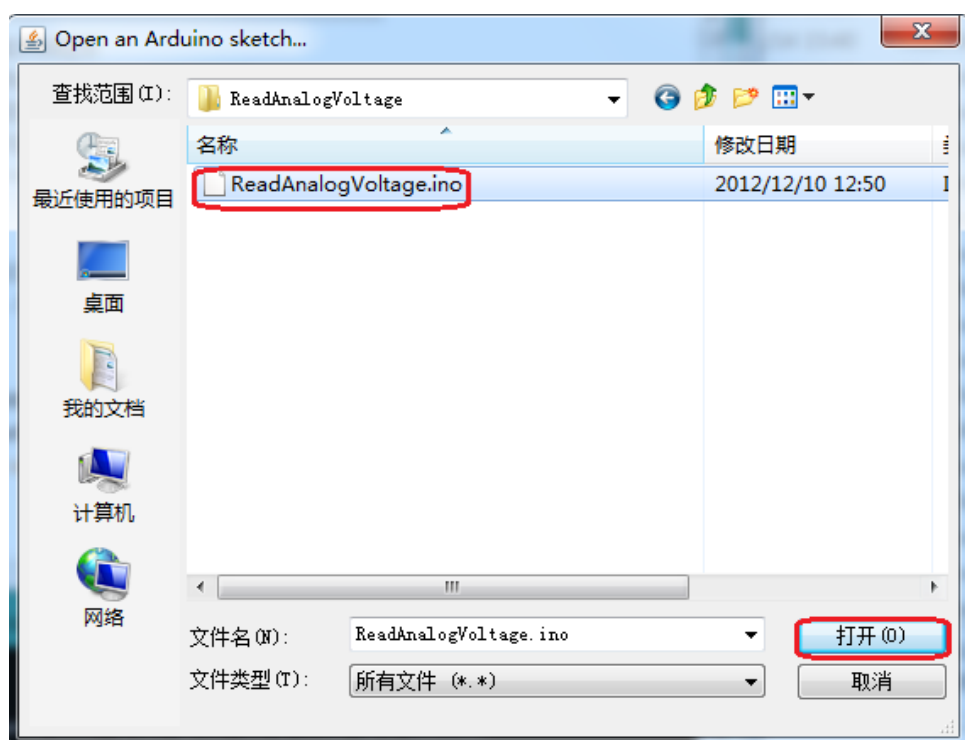


-  -> Verify
-  -> Upload
-  -> New file
-  -> Open file
-  -> Save file
-  -> Serial Monitor: data communicate with the board

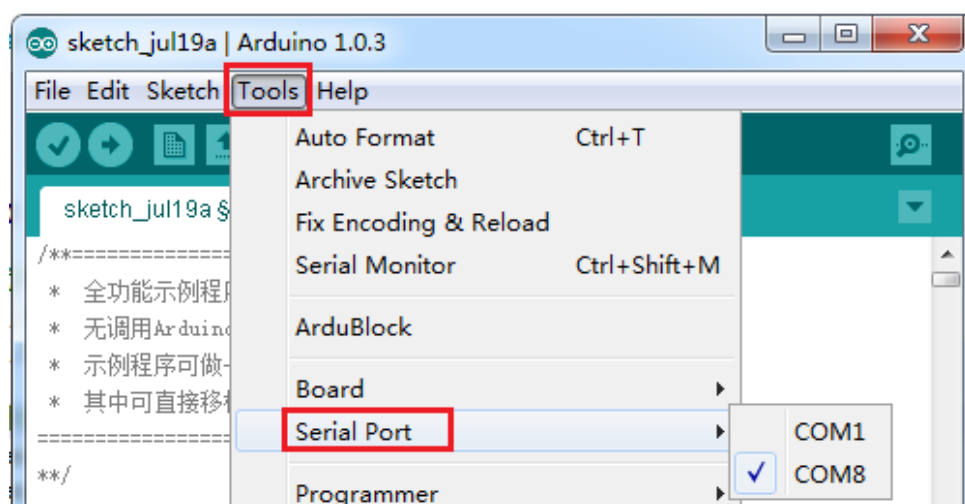
example:

1: open the file.

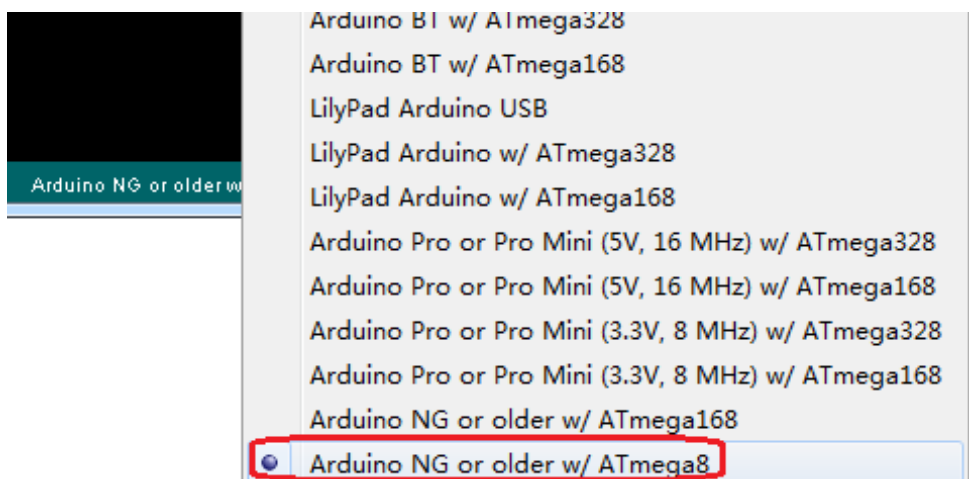
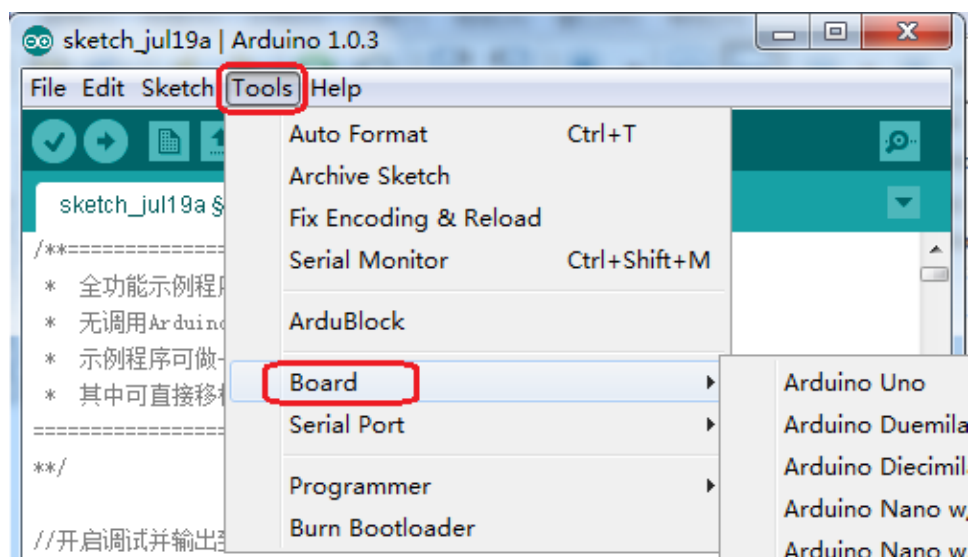




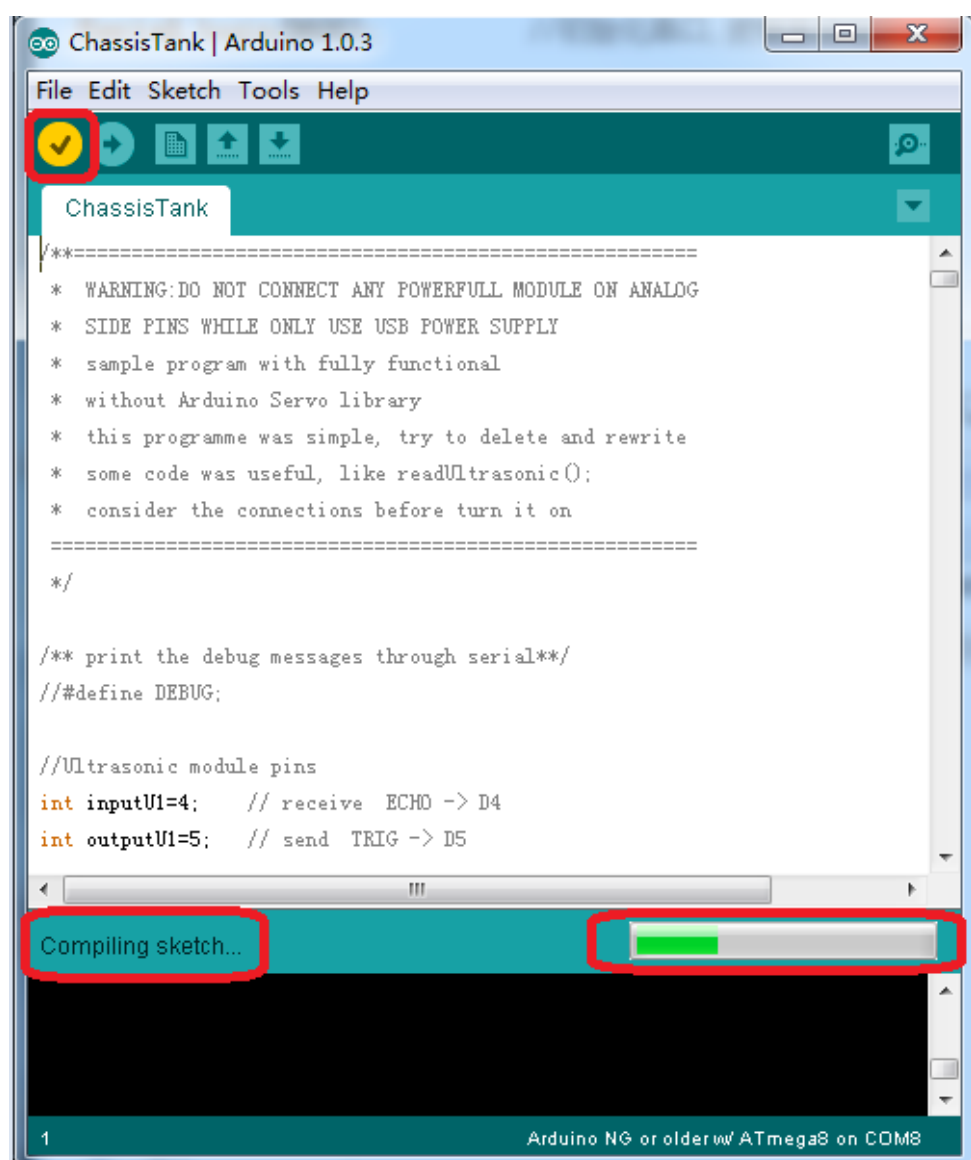
2. Please connect the mini driver USB to computer, install serial port driver before download and comply (please read the basic manual). Then select the correct COM port.



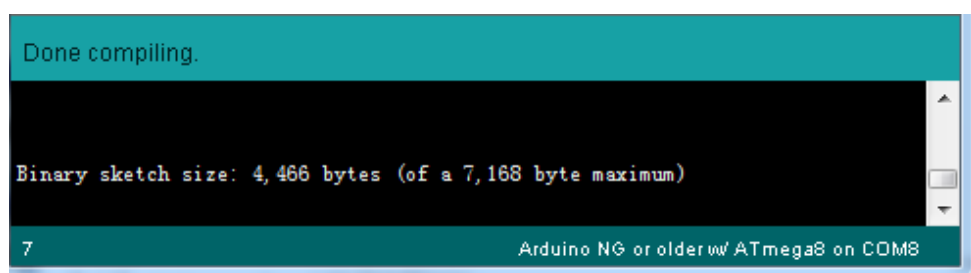
3. Select the correct controller board type, now please click Arduino NG or older w/ ATmega8



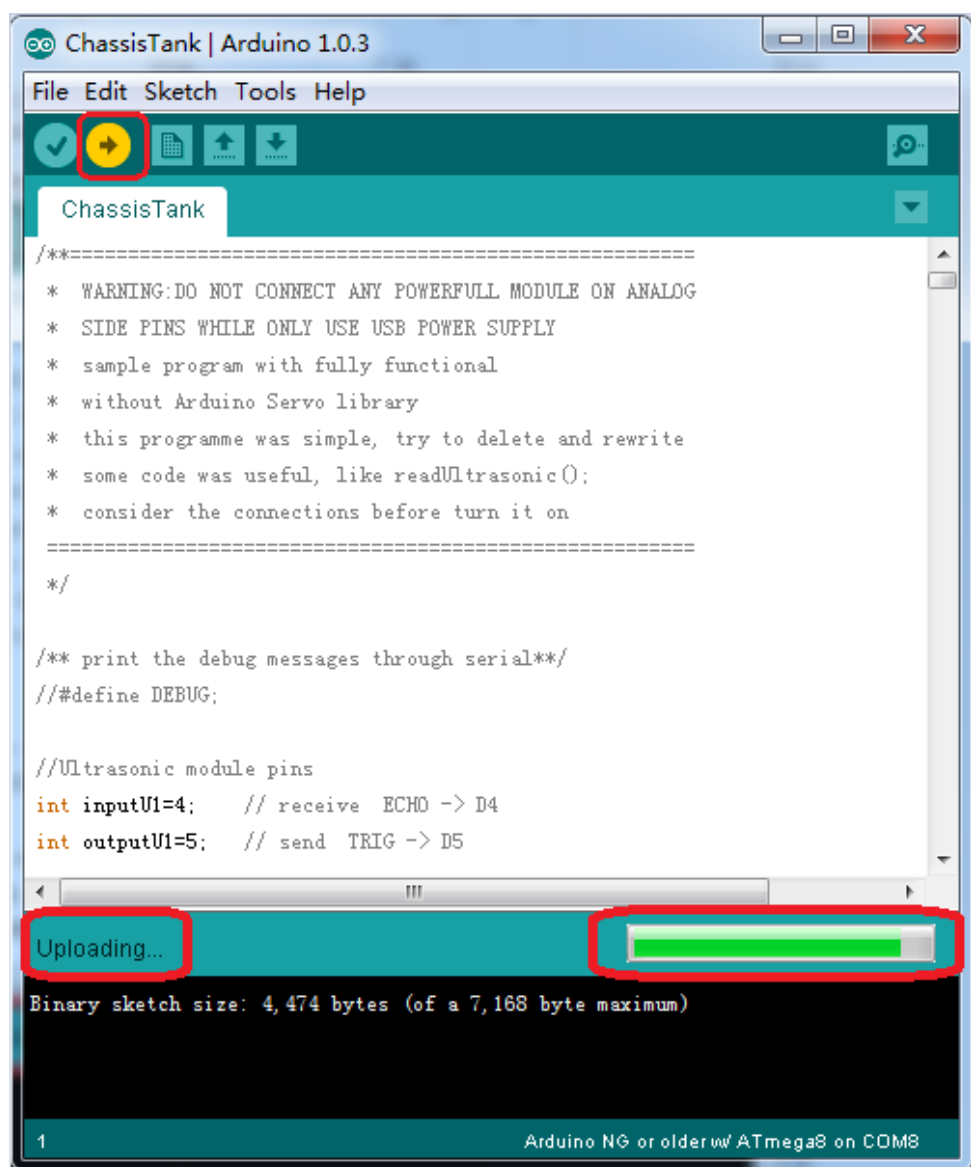
4. We can see the processing when compiling



If there is no red text error occur, it means comply success. Otherwise, there are some error occur.



5. upload the sample code to controller board.



If there is no red text error occur, it means uploading to the controller successfully.

Otherwise, please check if serial port is correct, or controler board connection failure or bootloader missing.

A.For bootloader missing, you can upload the bootloader again by ISP programmer.

B.Go to TOOLS, and check serial COM port is correct or not

C.pluck all the connections

Done uploading.

Binary sketch size: 4,466 bytes (of a 7,168 byte maximum)

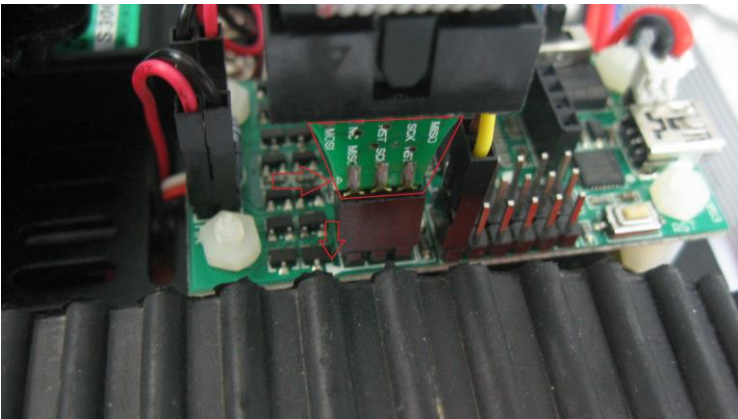
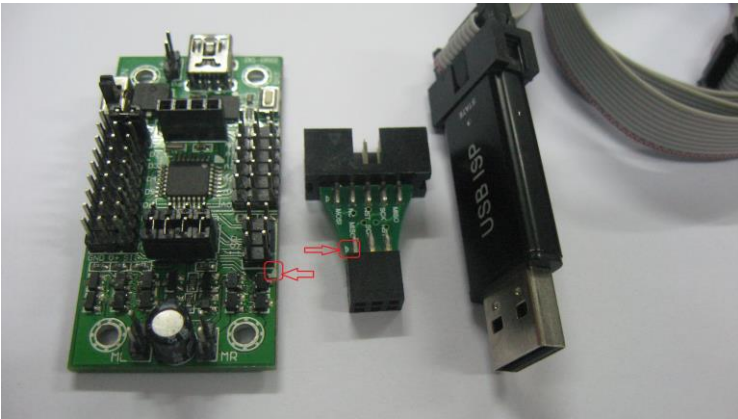
5

Arduino NG or older w/ ATmega8 on COM8

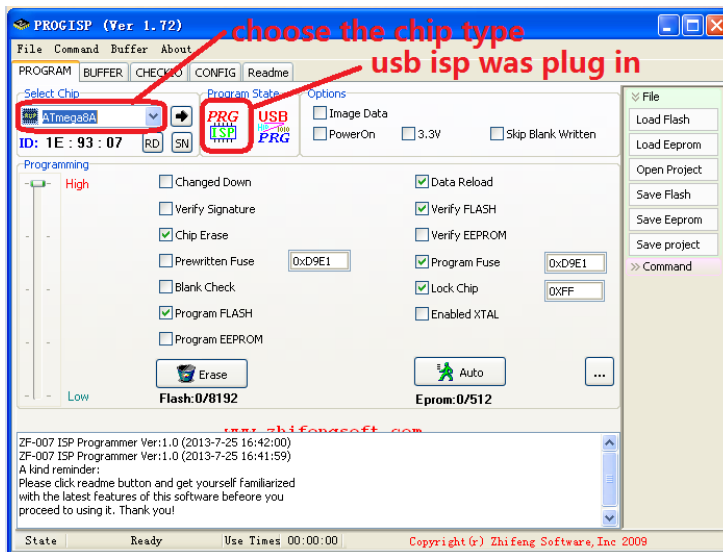
Burn bootloader by the ISP programmer



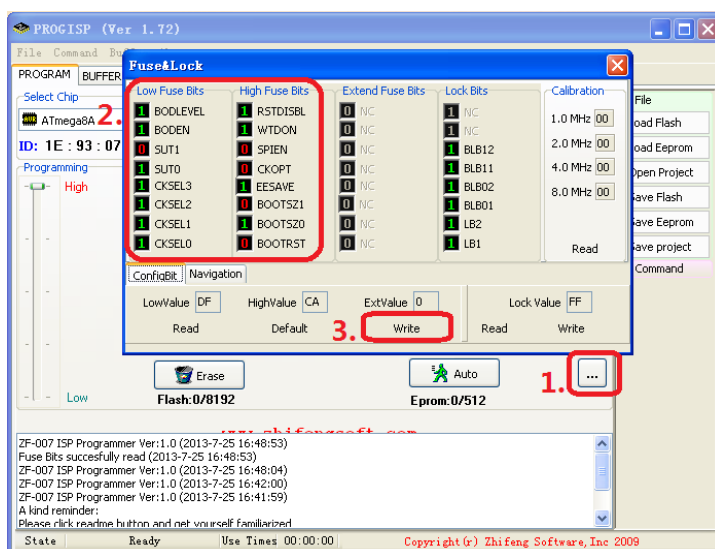
1. Before programming, connect the ISP cable in a correct pin order, (MISC corresponds with the triangle single on the mini driver)



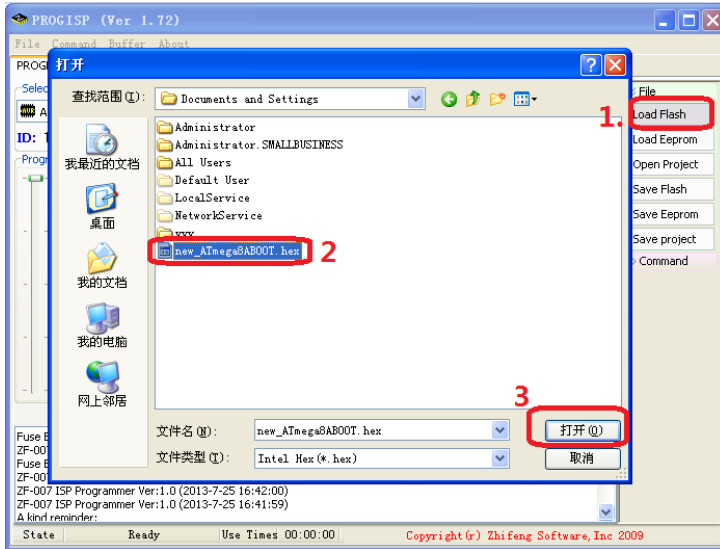
2. open the software (if driver needed, install like CP2101, and the driver was in ISP folder): ATmega8A



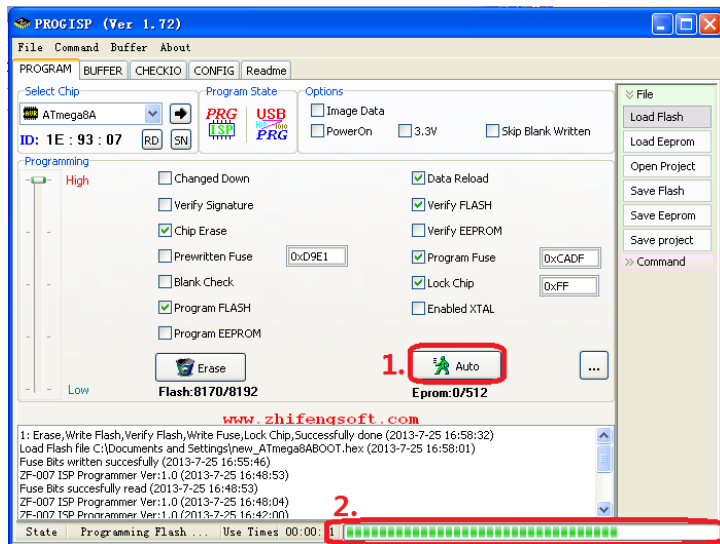
3. Reset the fuse bit, click below to 0 or 1.(if after button Read is already like this, there is no need to change)
Warning: do not make it wrong, otherwise will damage the chip.



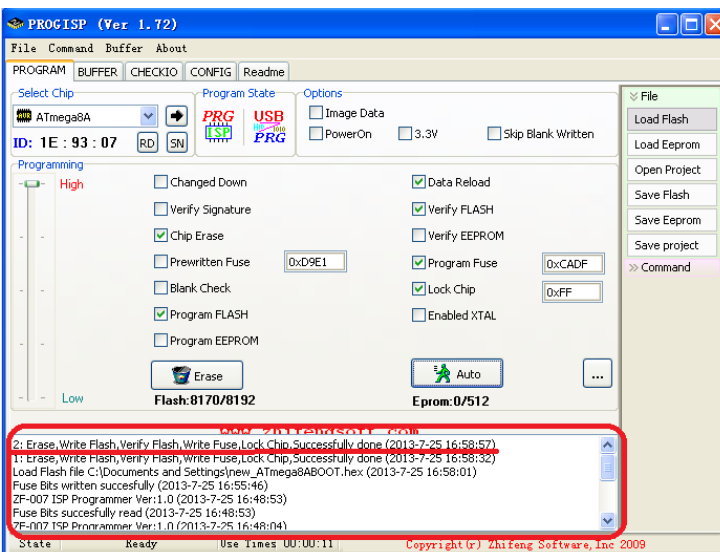
4. Burn the correct files.



5.click the Auto button



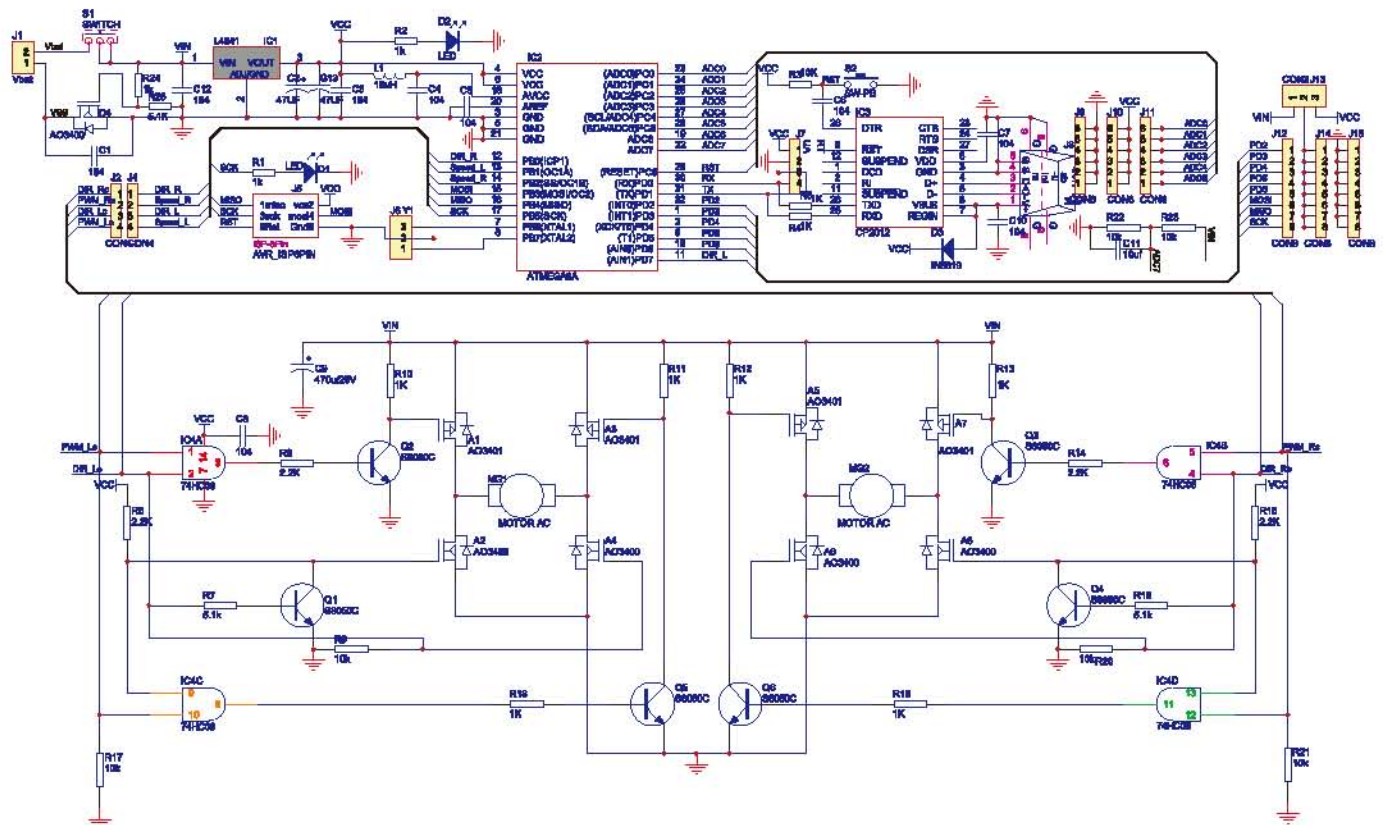
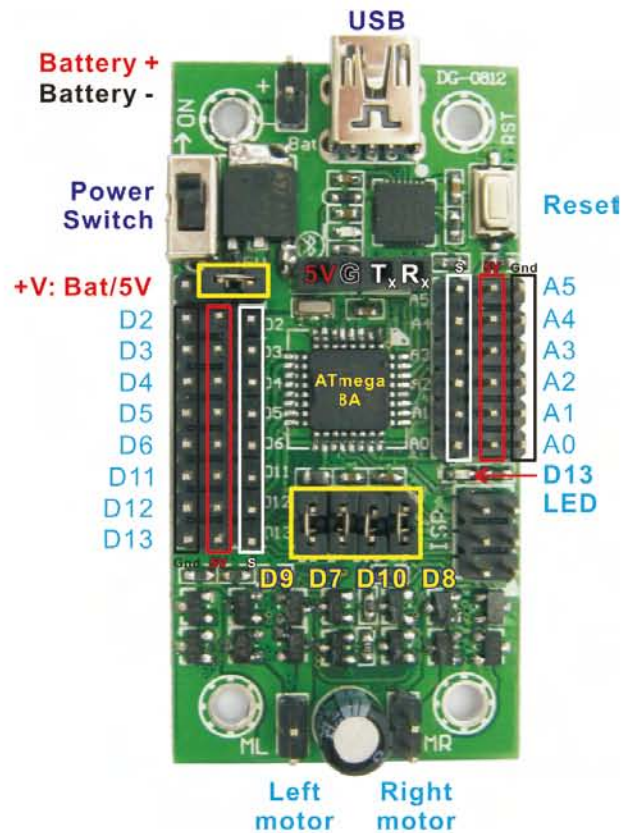
After finished, it will warn that successfully done.



Robot Controller:

The Playful Puppy uses the **Mini Driver** which is a general purpose robot controller from DAGU.

This is a small, Arduino compatible controller that comes with the Arduino bootloader pre-installed.



Specifications:

- Input voltage 4.5V - 9V with reverse polarity protection rated at 4A.
- LDO 5V regulator rated at 1A.
- USB interface and ISP socket, can be programmed with USB cable or ISP programmer.
- ATmega8A processor @ 16MHz with 8K FLASH, 1K SRAM and 512 Bytes EEPROM memory.
- 8x digital I/O pins with servos compatible 3 pin male headers and selectable power (+5V / +Battery).
- 6x analog inputs with regulated 5V and Gnd for powering analog sensors.
- Built in battery monitor on analog input A7.
- Dual FET "H" bridge rated at 2.5A per channel can drive 2x DC motor or 1x Stepper motor.
- Serial port for optional Bluetooth transceiver.