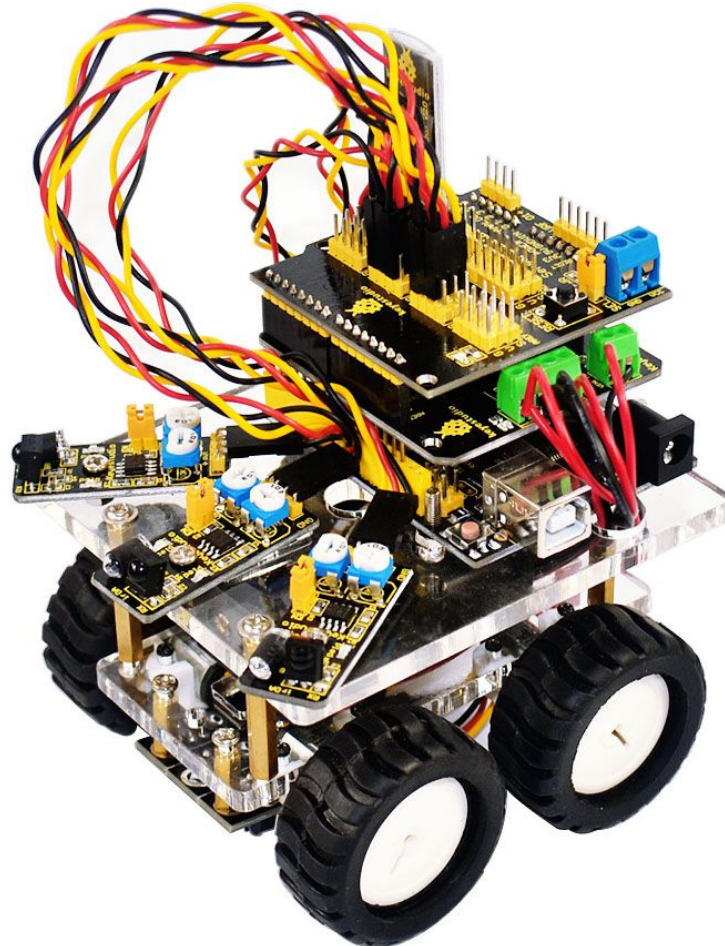


# keystudio

---

## keystudio Desktop Bluetooth Mini Smart Car



### Catalog

1. Introduction.....	1
2. Parameters.....	1
3. Standard component list.....	1
4. Installation instructions.....	2
5. Application of Arduino.....	27
6. Project details.....	36
Project 1: Line-tracking smart car.....	36
Project 2: Obstacle-avoidance smart car.....	40
Project 3: Bluetooth smart car.....	45

# keystudio

---

## 1. Introduction

Multi-functional smart car is a learning application development system of microcontroller that can be controlled by ARDUINO. It has functions of line tracking, obstacle avoidance, Bluetooth remote control etc. This kit contains many interesting programs. It can also be expanded to have external circuit modules to have other functions. This kit is designed to help you interestingly learn Arduino. You can learn Arduino MCU development ability while having fun.

## 2. Parameters

1. Motor parameters: Voltage: 1.5 -12V; shaft length: 10mm; Revolving speed: 6.0V 100rpm/min.
2. Use L298P driver module for motor control.
3. Three-channel line tracking modules, detecting black and white line, high precision, can also be used in fall prevention.
4. Three-channel infrared obstacle avoidance modules makes up the obstacle avoidance sensor, can detect whether there are obstacles ahead.
5. Equipped with Bluetooth wireless module, can remotely control the robot after pairing with mobile phone Bluetooth.
6. Can be connected to external 7 ~ 12V power supply; with various sensor modules, it can realize various functions.

## 3. Standard component list

1. N20 metal gear motor \* 4
2. High quality tire \* 4
3. Motor holder \*4
4. Car chassis \* 2
5. L298P motor driver shield \* 1
6. ARDUINO UNO328 board \* 1
7. ARDUINO sensor shield \* 1
8. Infrared obstacle avoidance module \*3
9. Line tracking module \*3
10. HC-06 Bluetooth module \* 1
11. Toggle switch \* 1
12. 14500 battery pack \* 1
13. 3PIN Dupont wires \* 6
14. 27MM Copper bush \* 4
15. 10MM Copper bush \* 2
16. 3MM screws and nuts \* several

\*\*\*\*\*

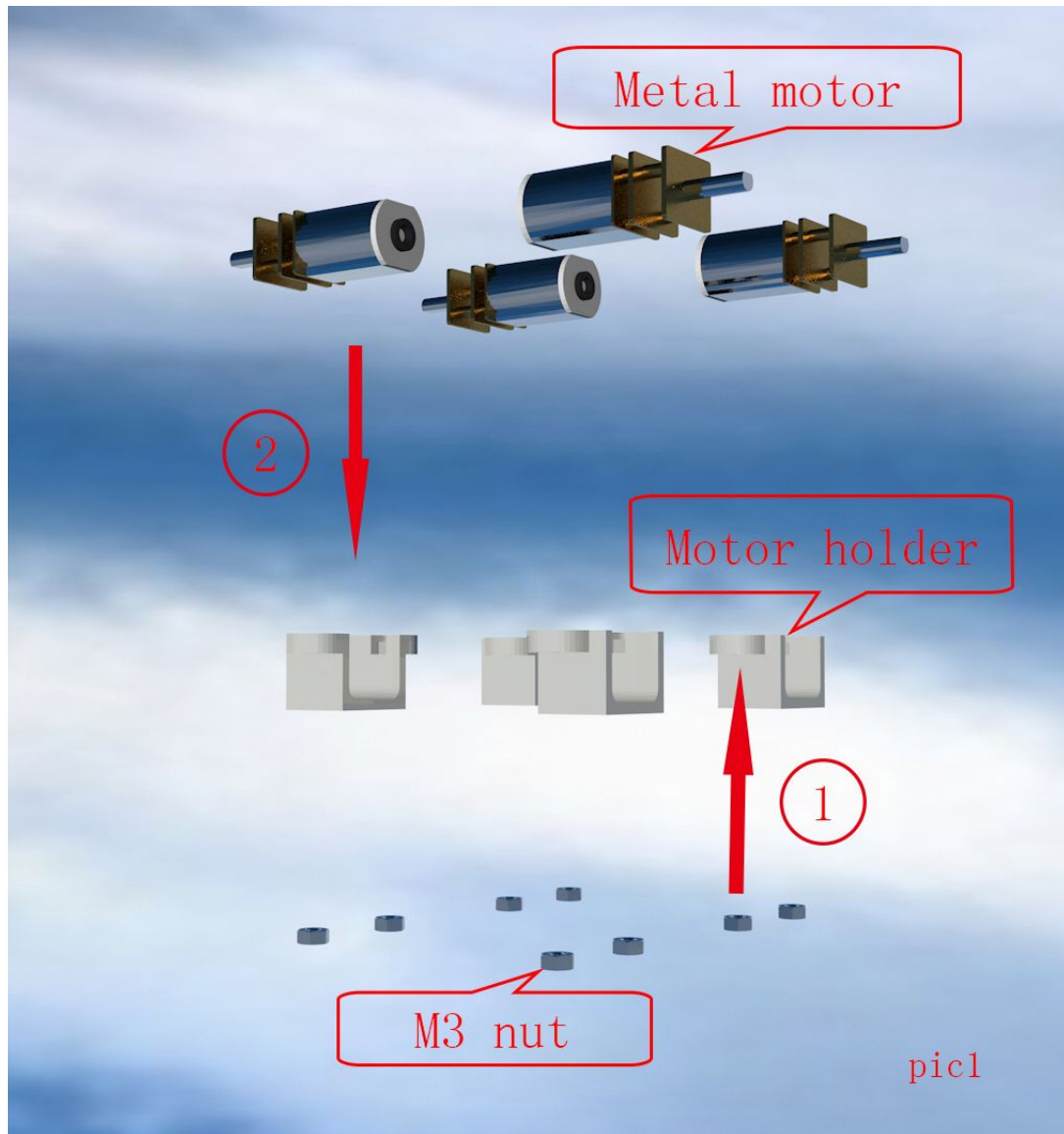
\* Self-prepare part

# keystudio

3.7V large capacity 1300mA imported 14500 rechargeable batteries \*2

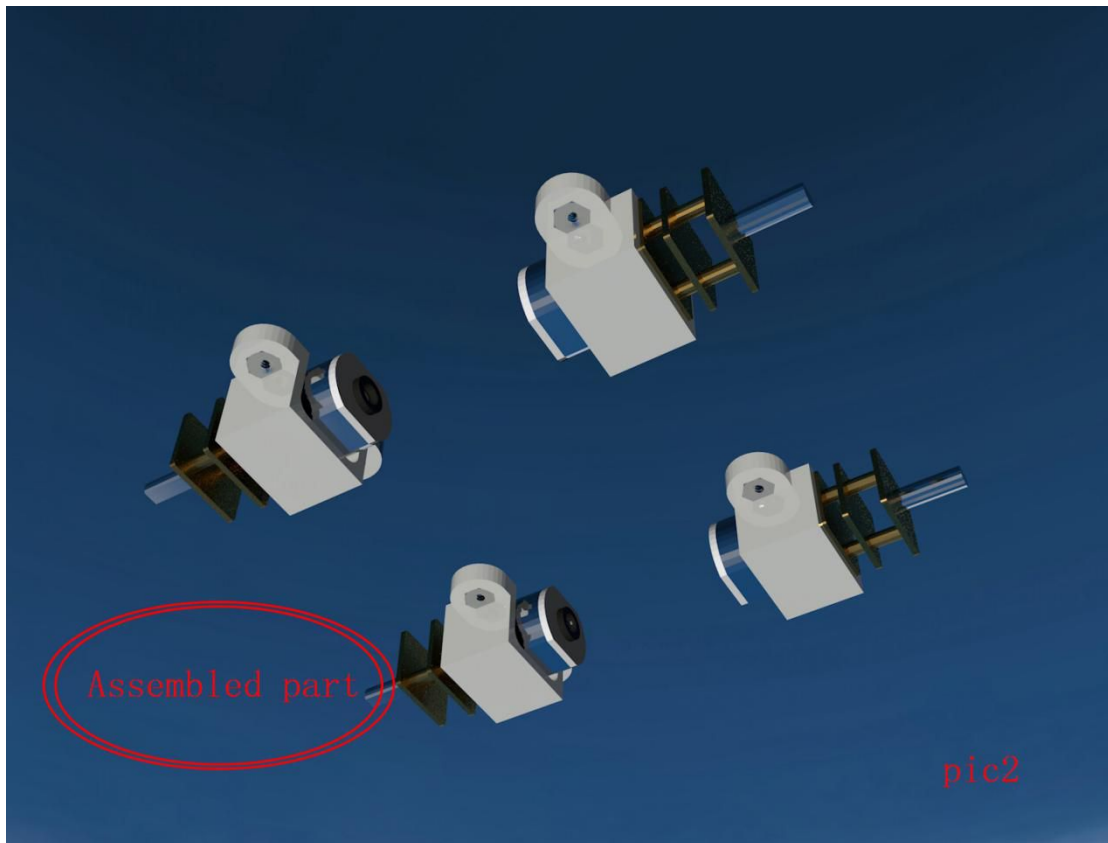
14500 charger \*1

## 4. Installation instructions



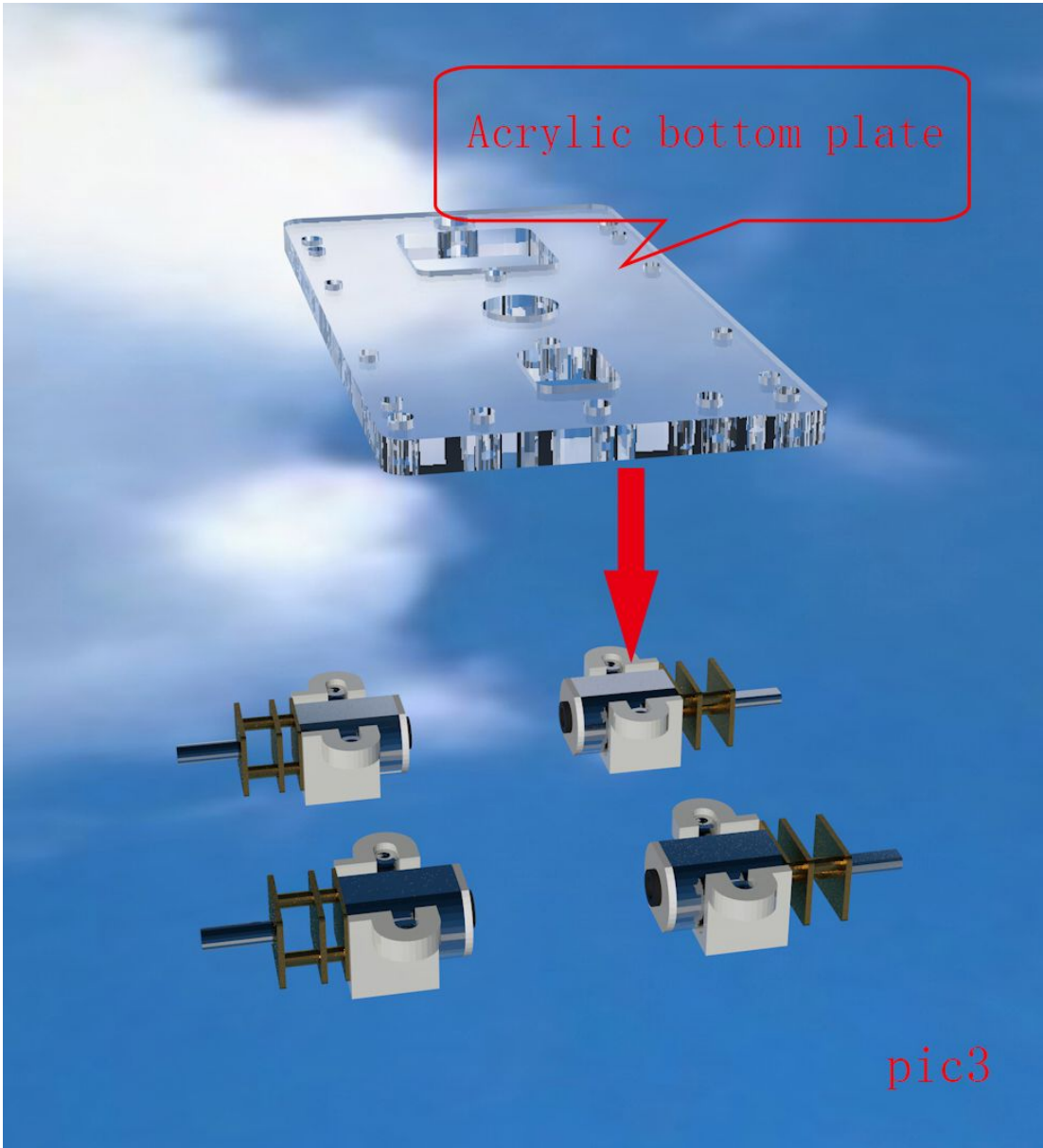
# keystudio

---

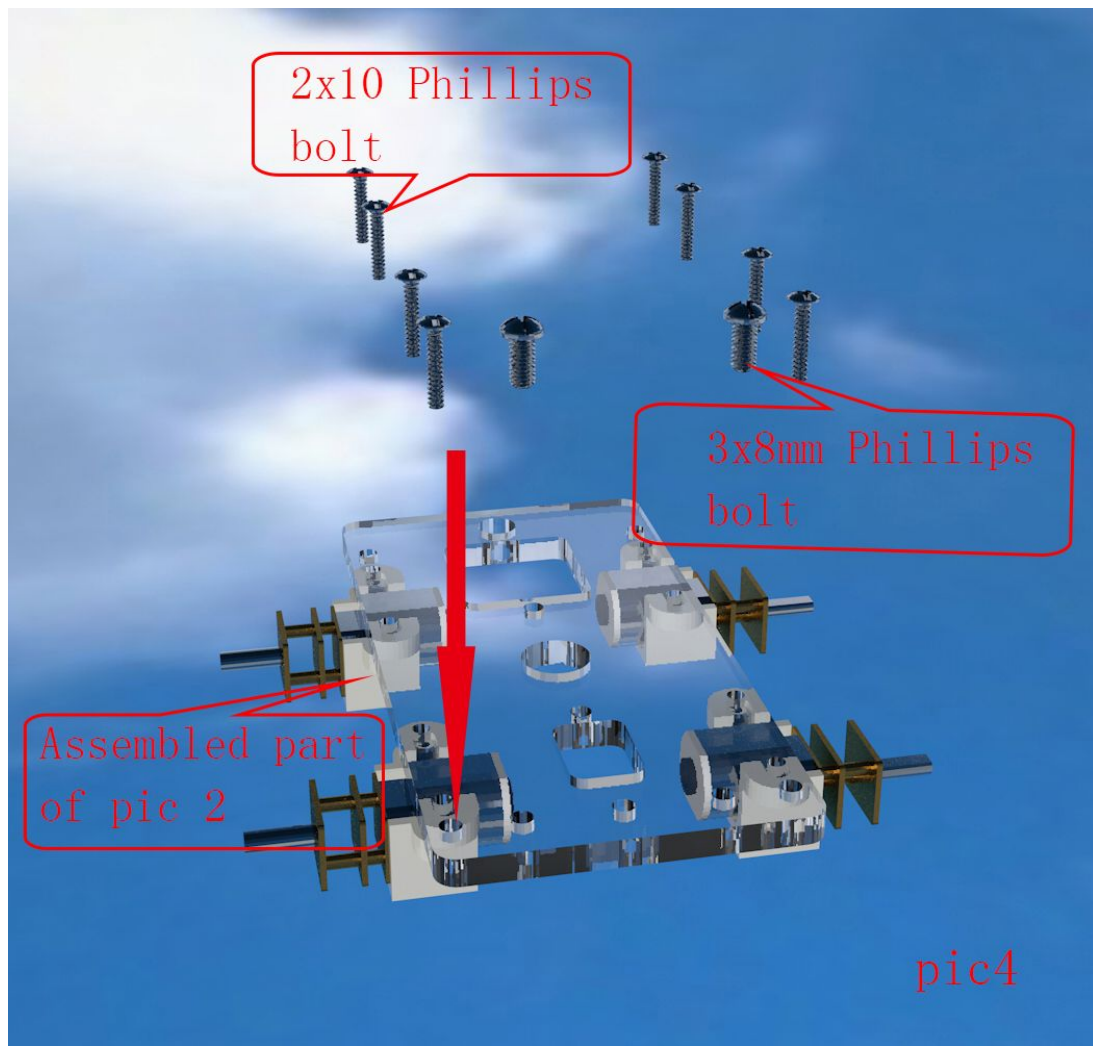


# keystudio

---



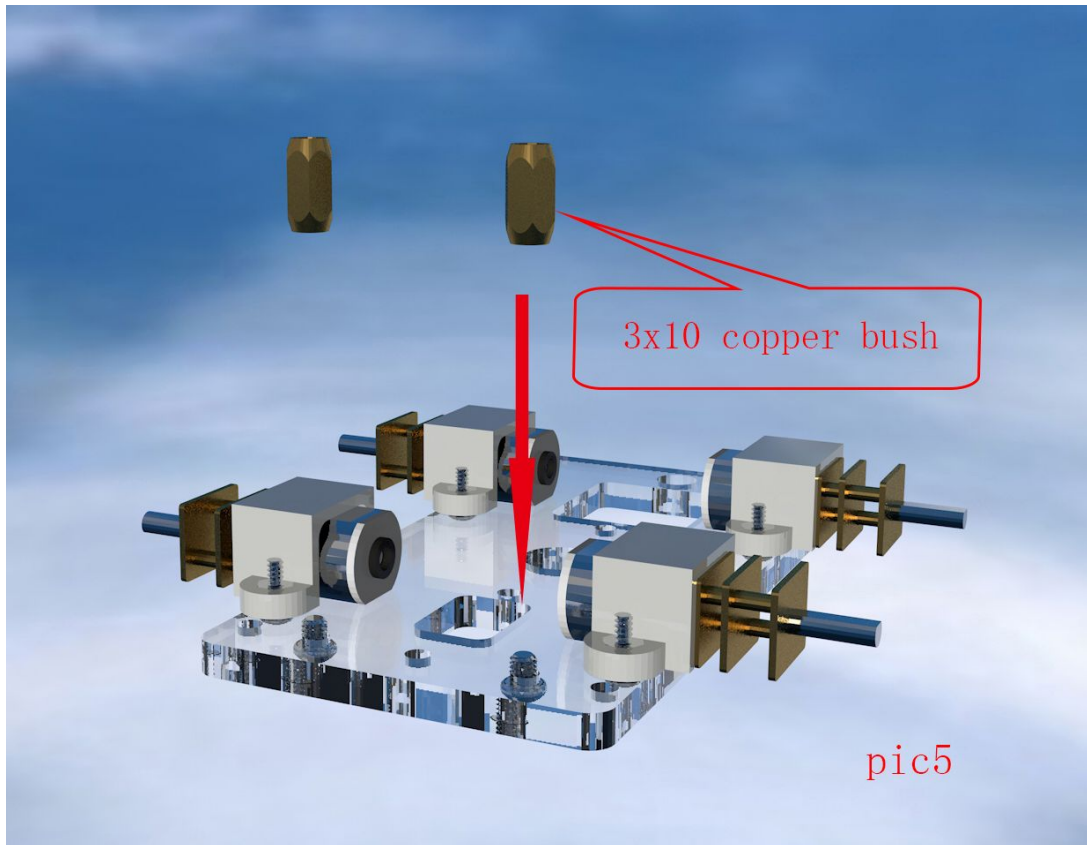
# keystudio



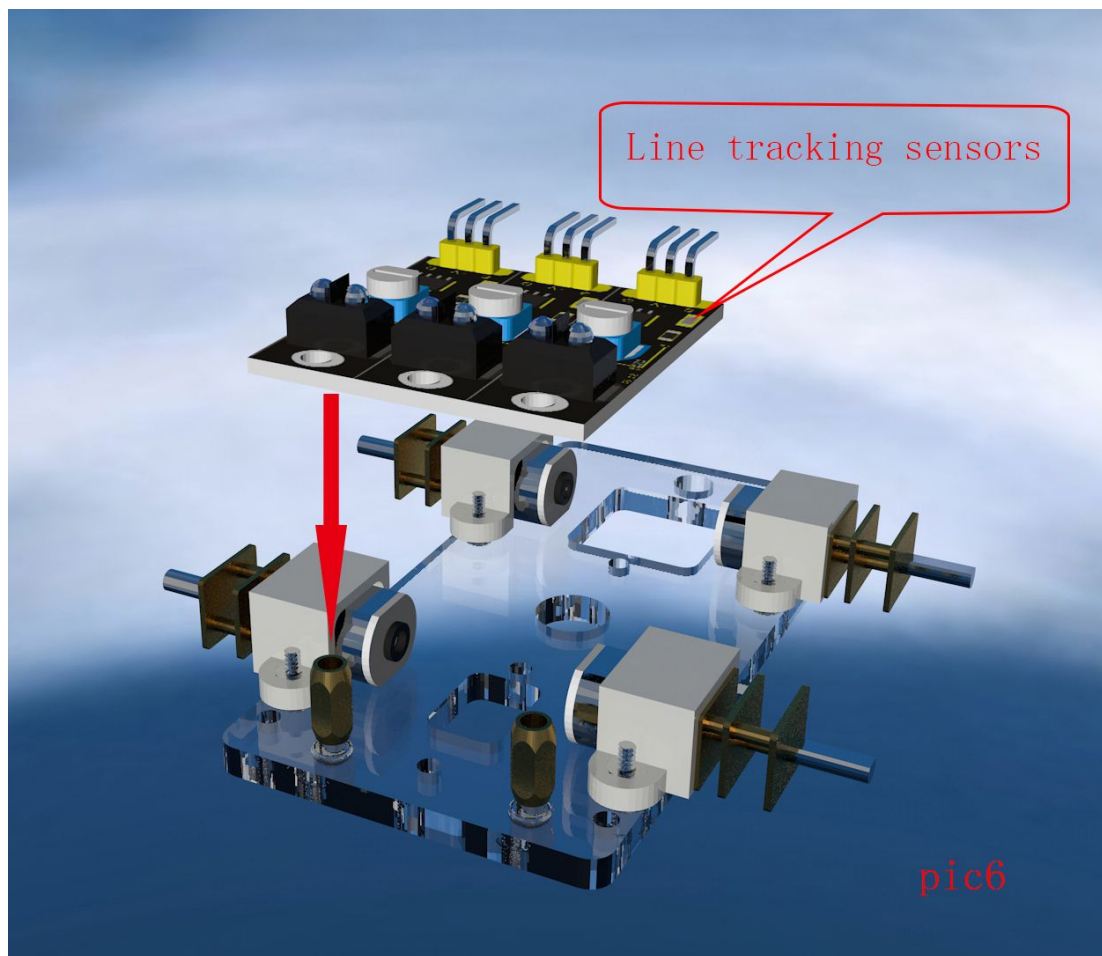


# keystudio

---

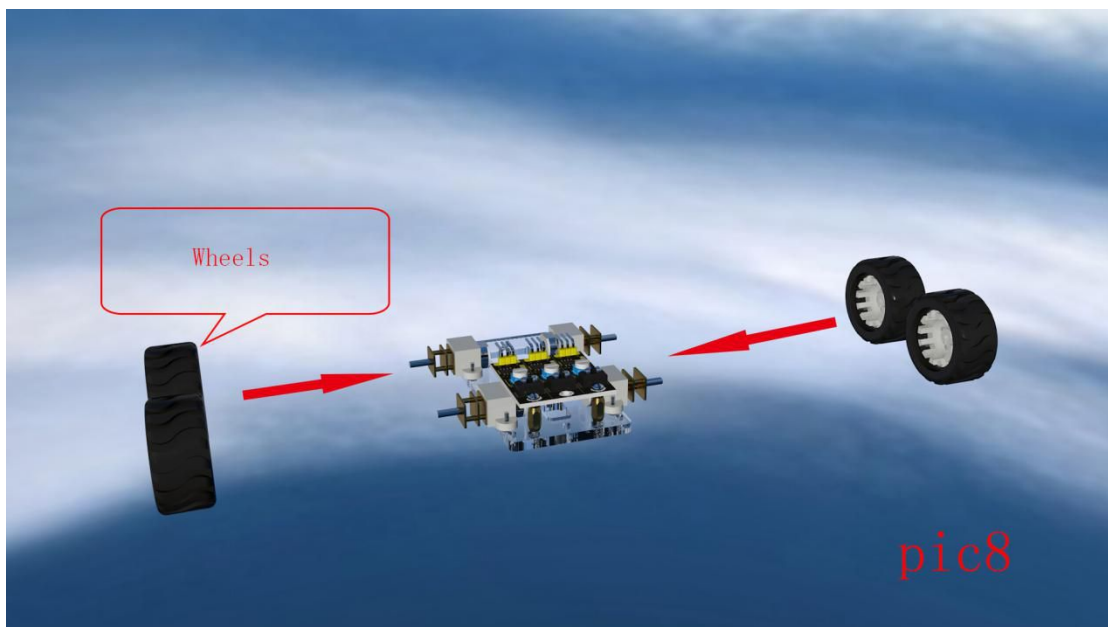
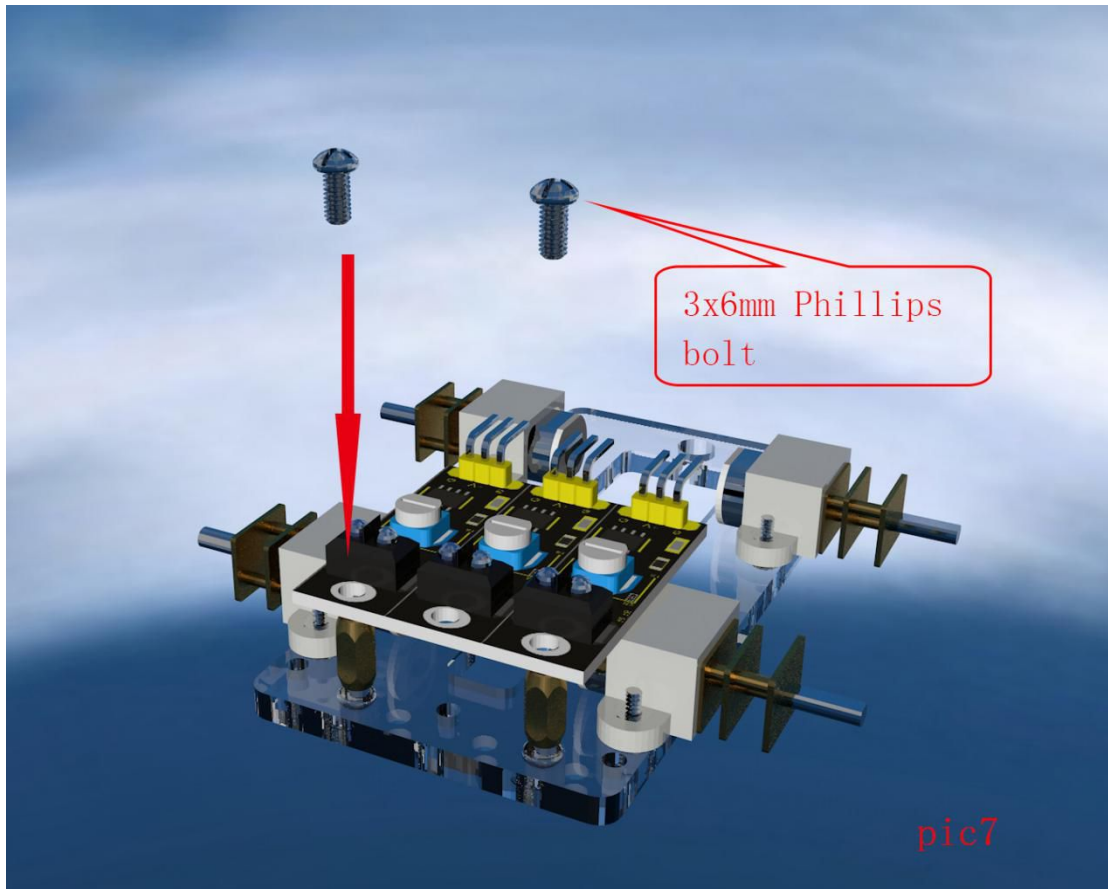


# keystudio



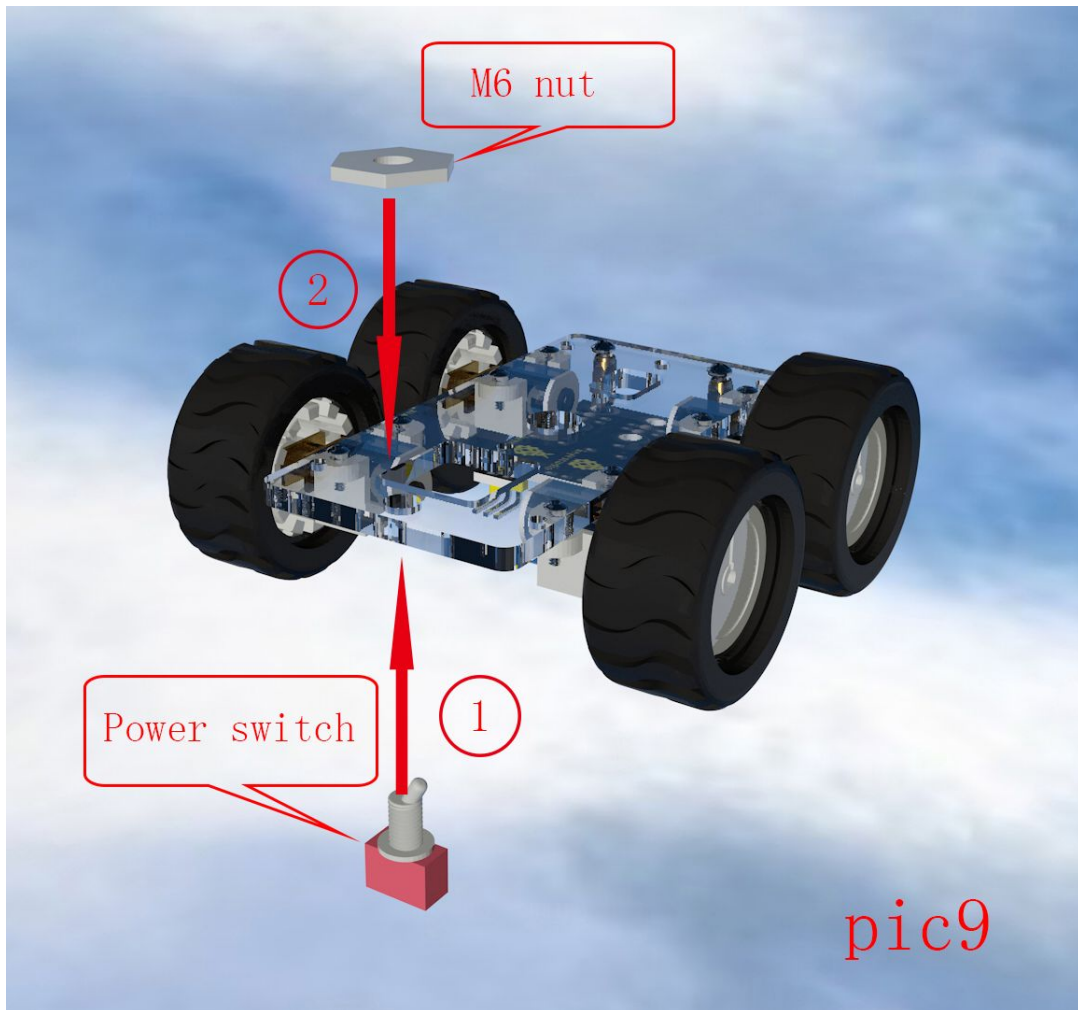


# keystudio



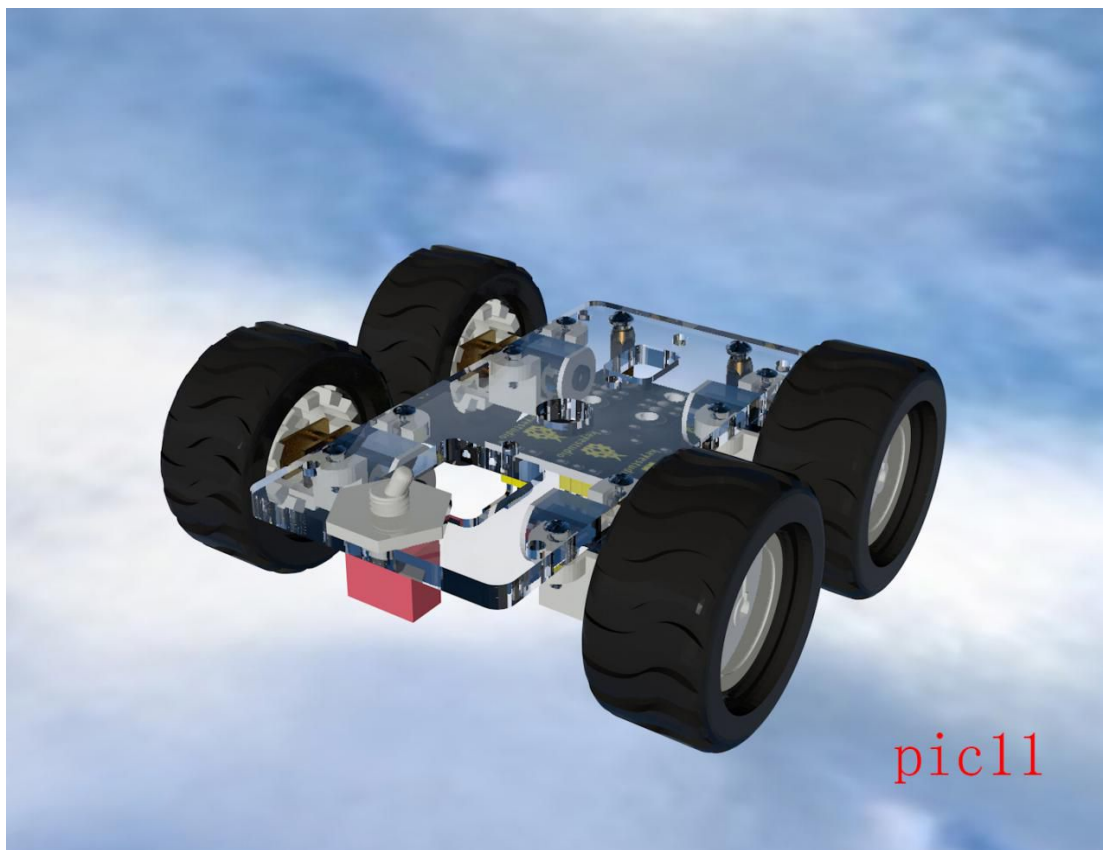
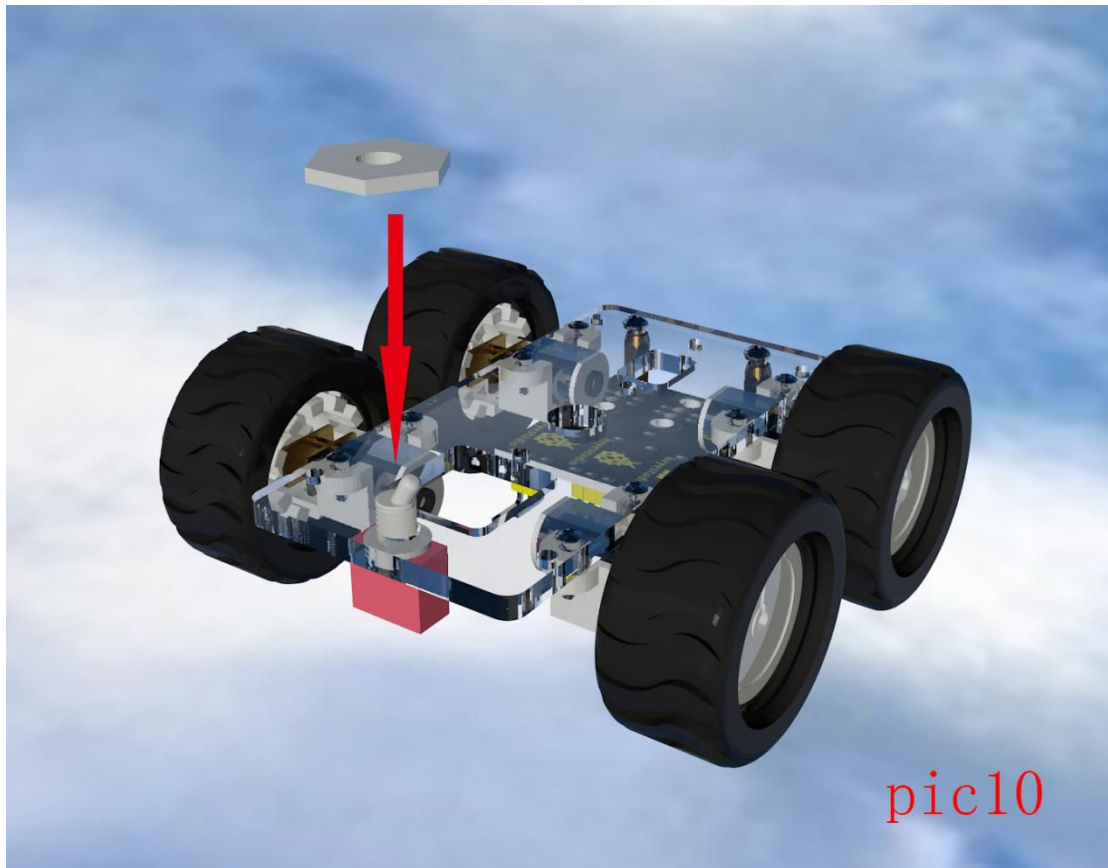
# keystudio

---

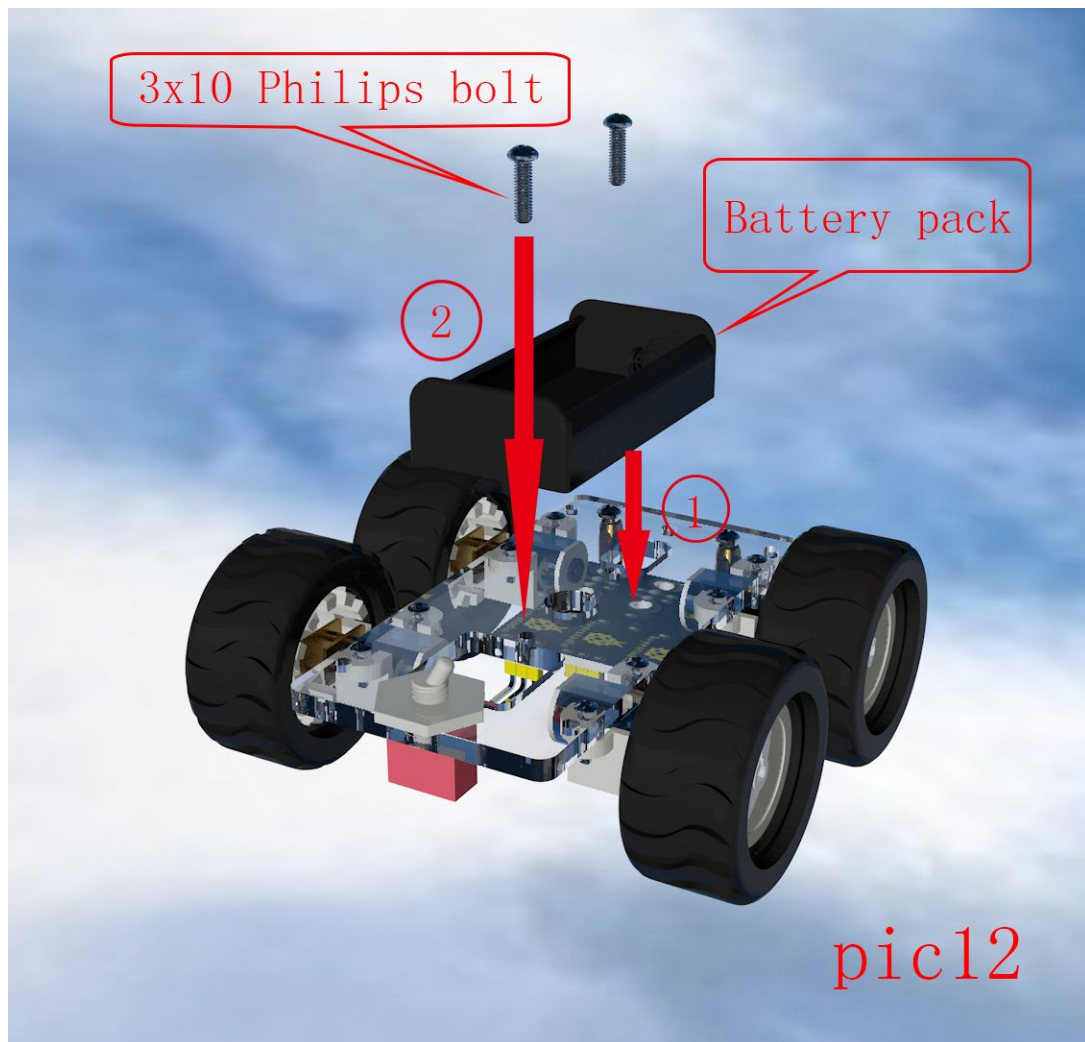


# keystudio

---

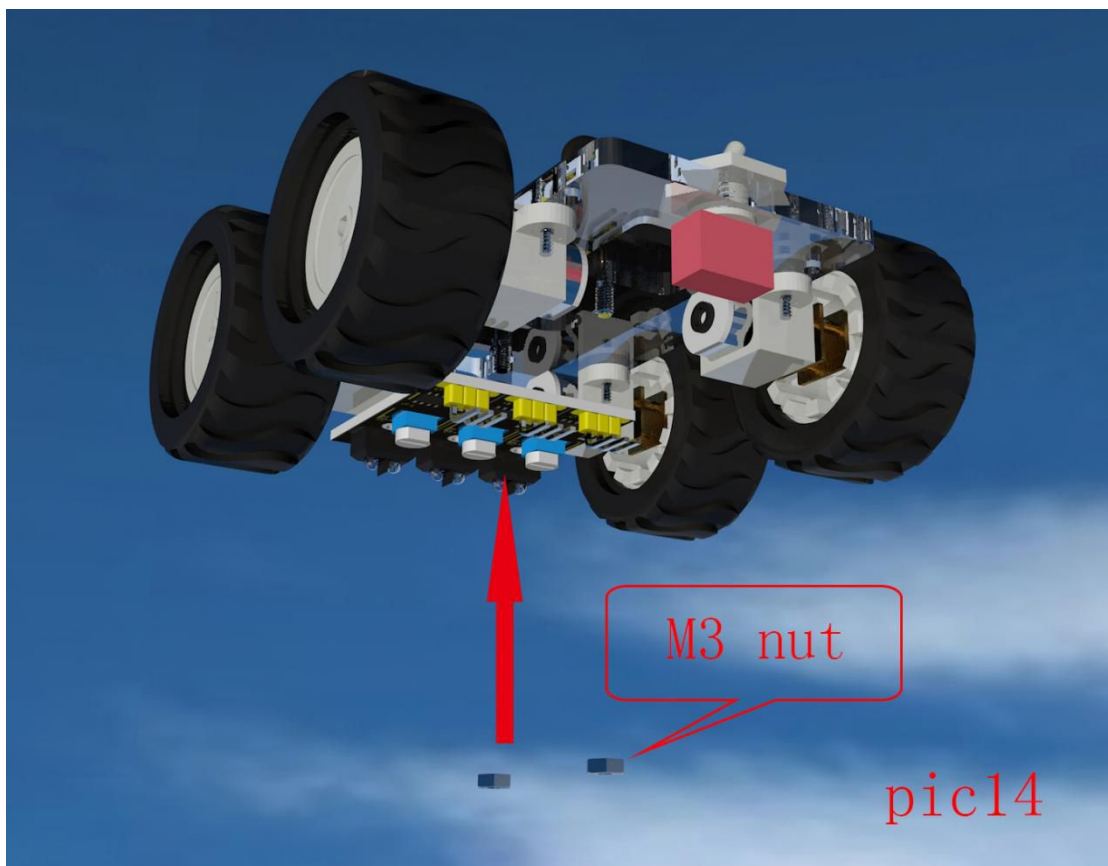
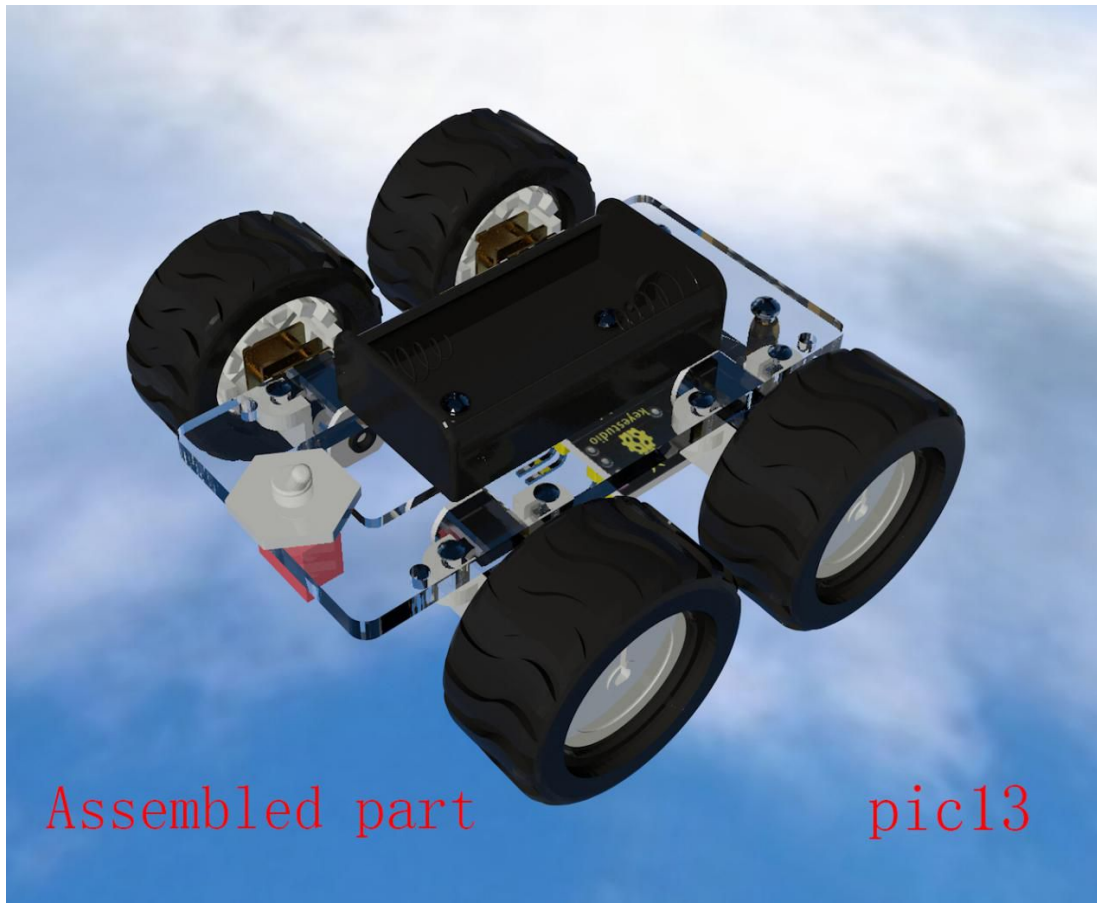


# keystudio

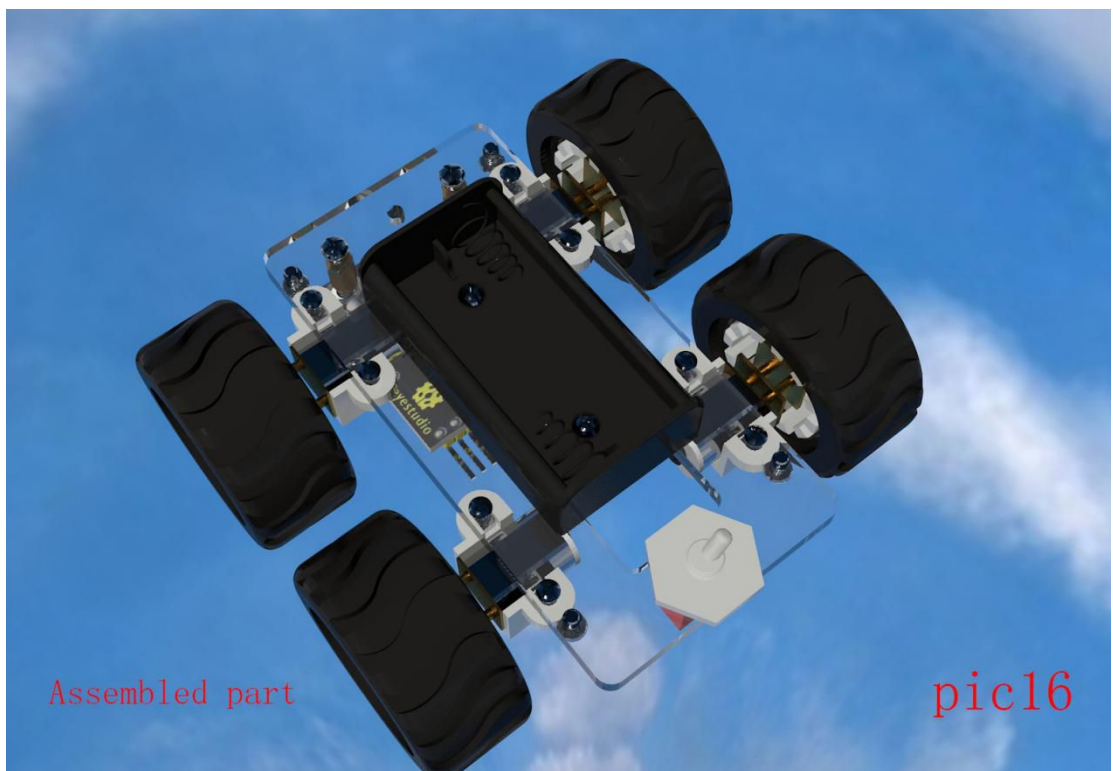
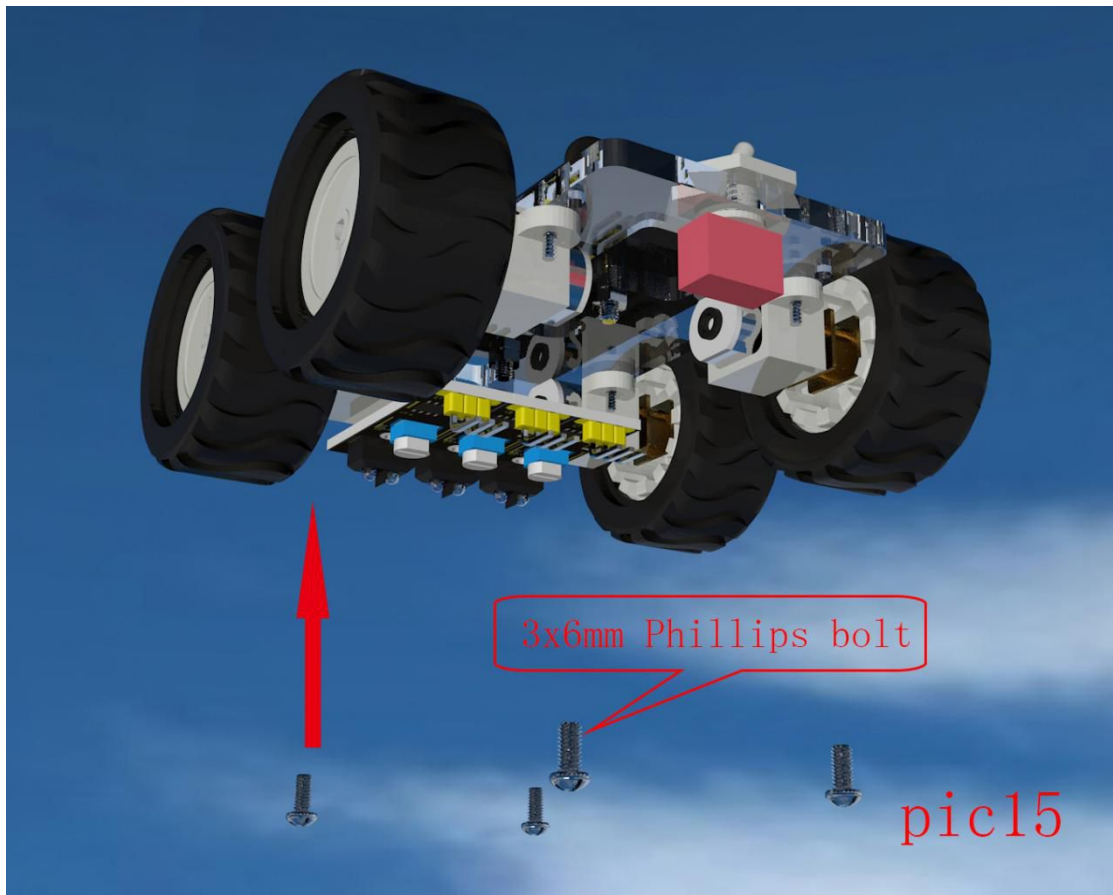




# keystudio



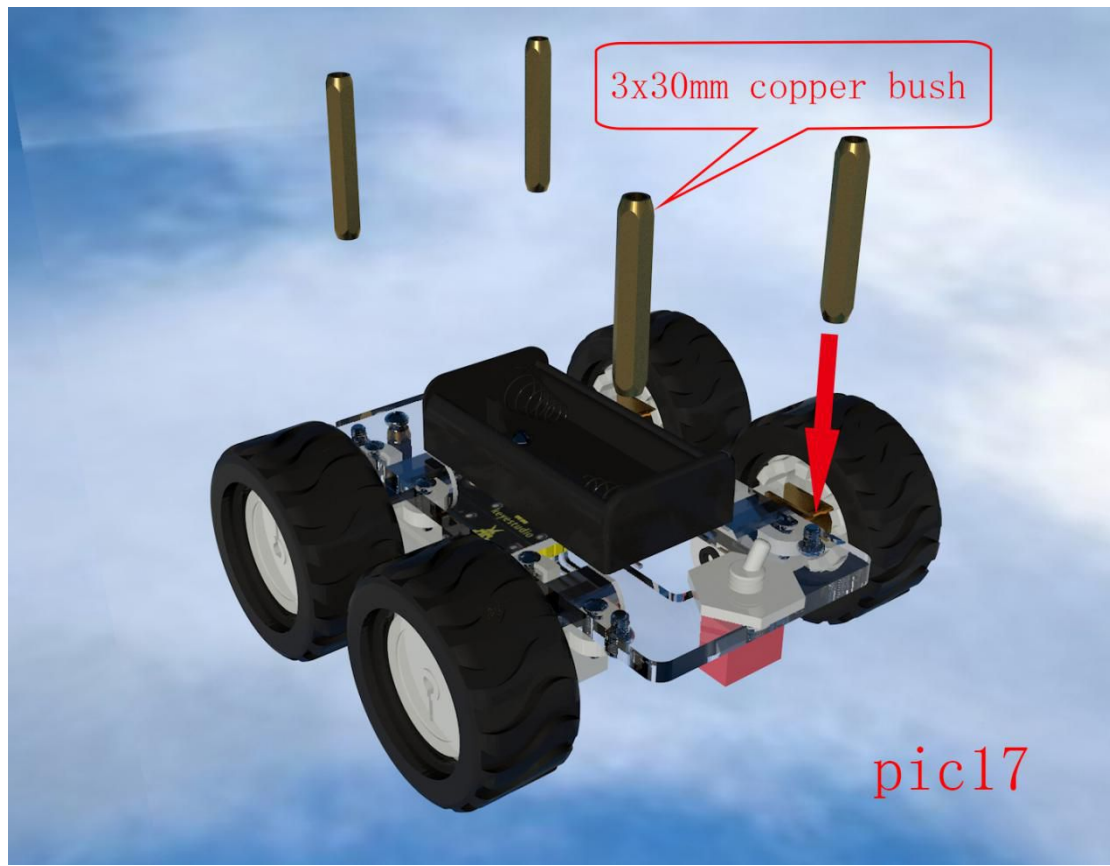
# keystudio



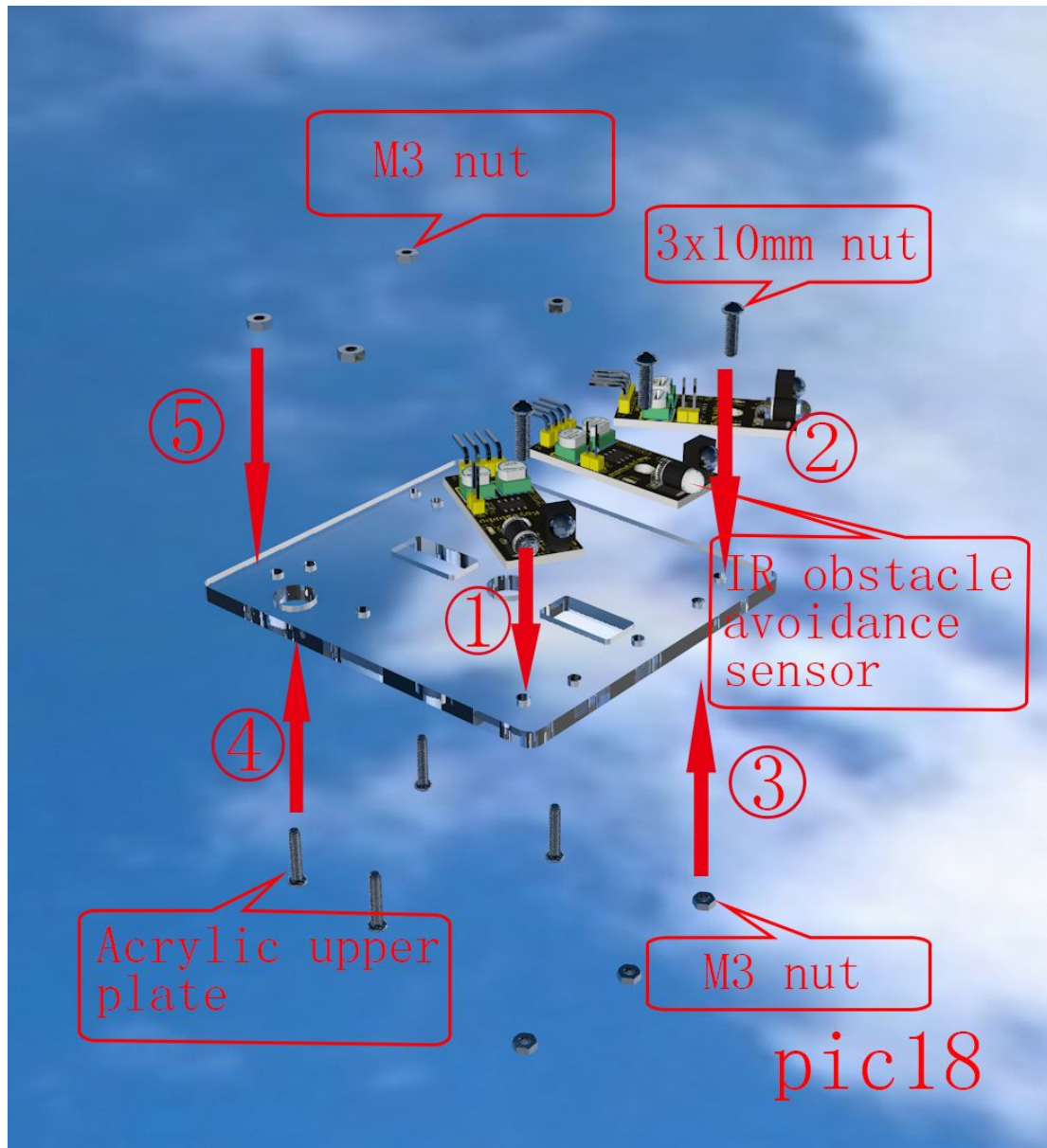


# keystudio

---



# keystudio



# keystudio

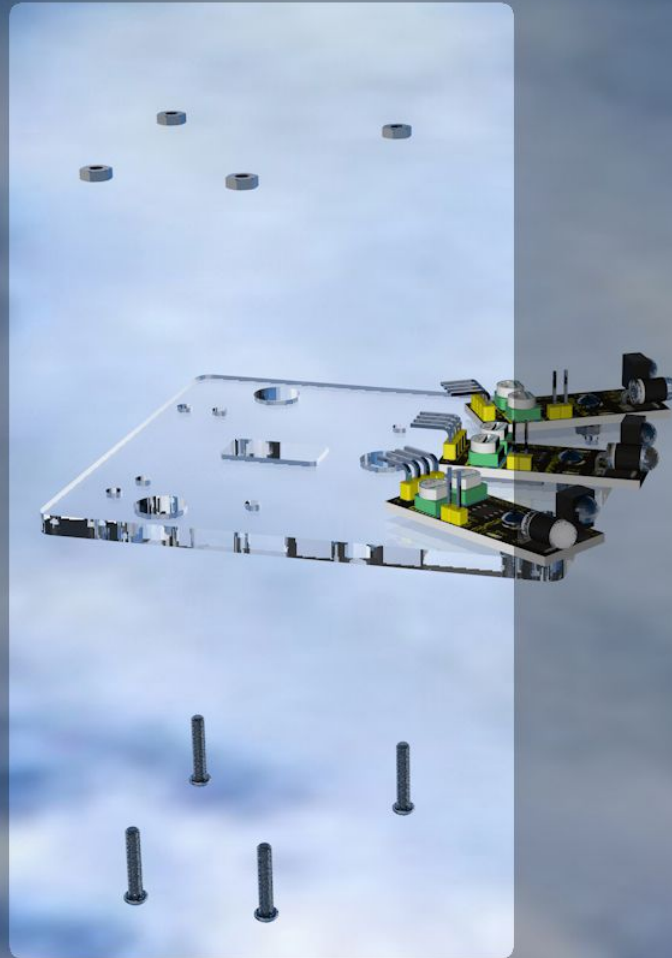
---



# keystudio

---

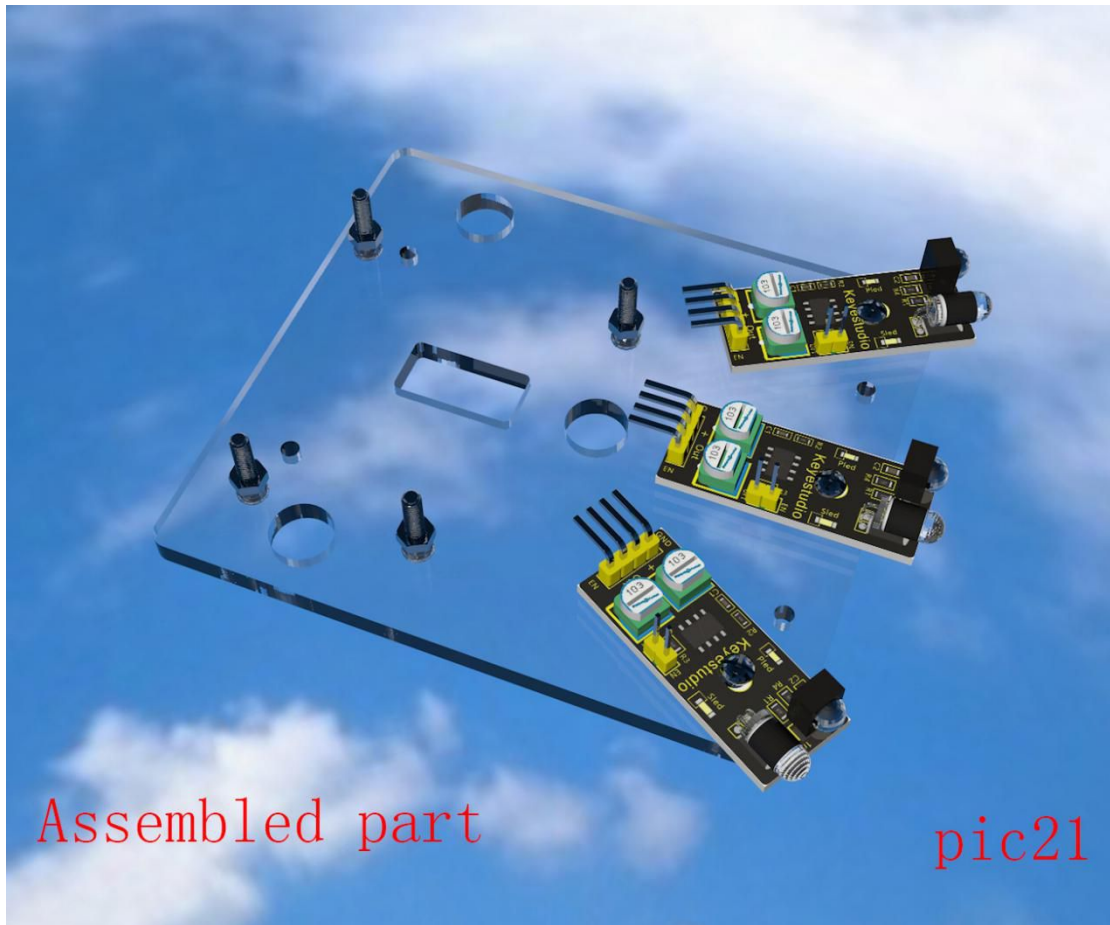
Assembled part of pic 19



pic20

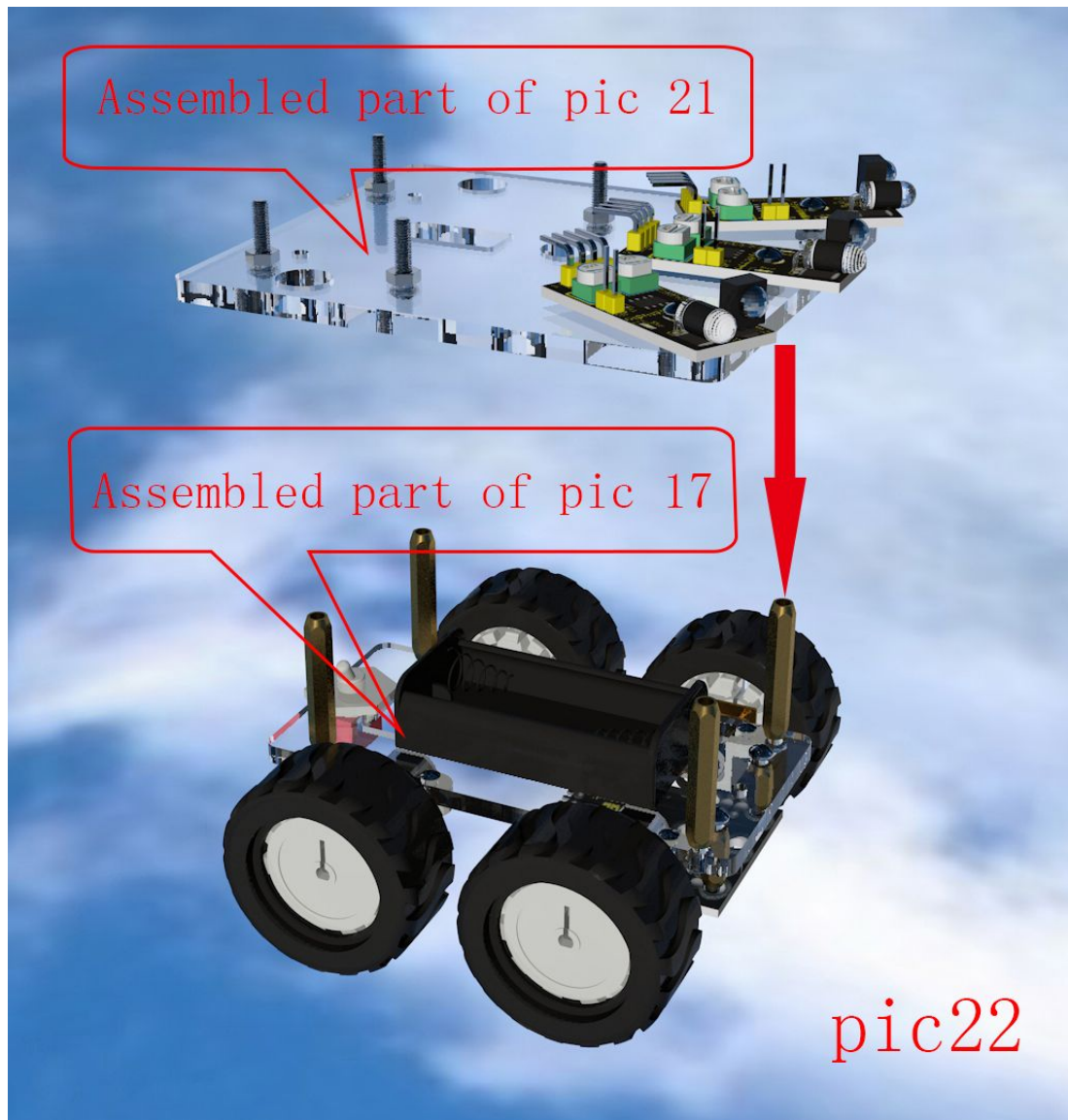
# keystudio

---





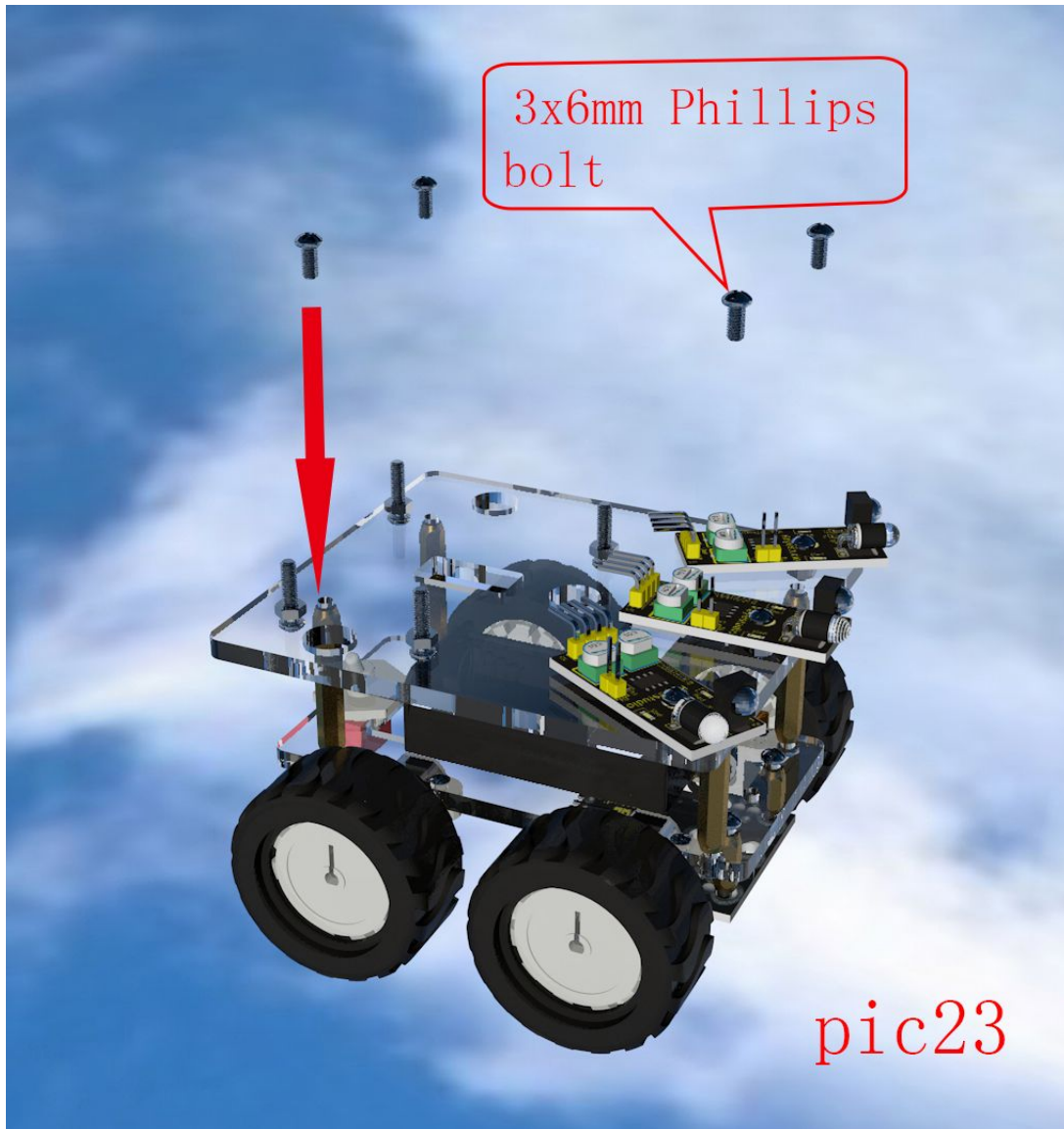
# keystudio





# keystudio

---



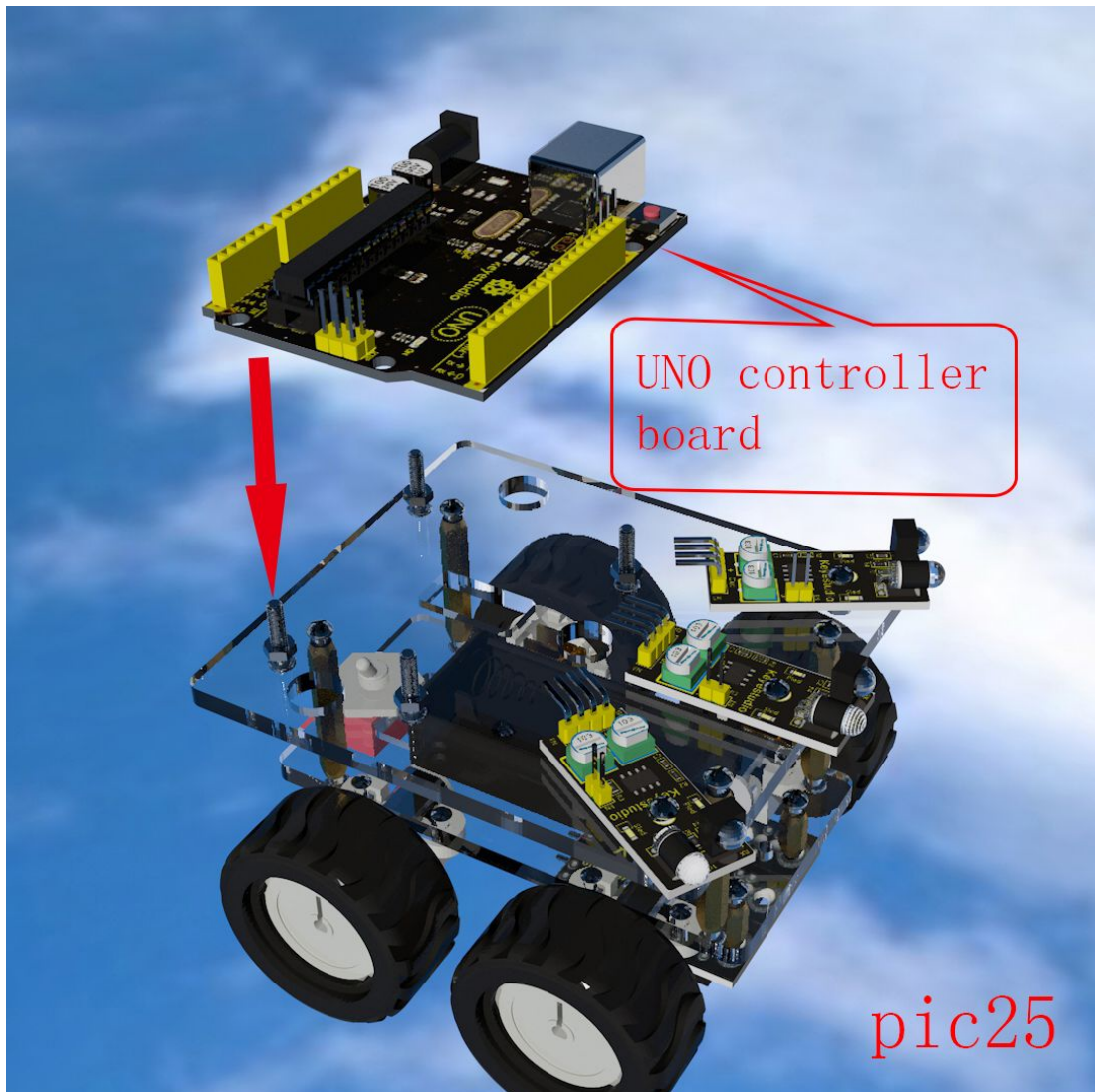
# keystudio

---



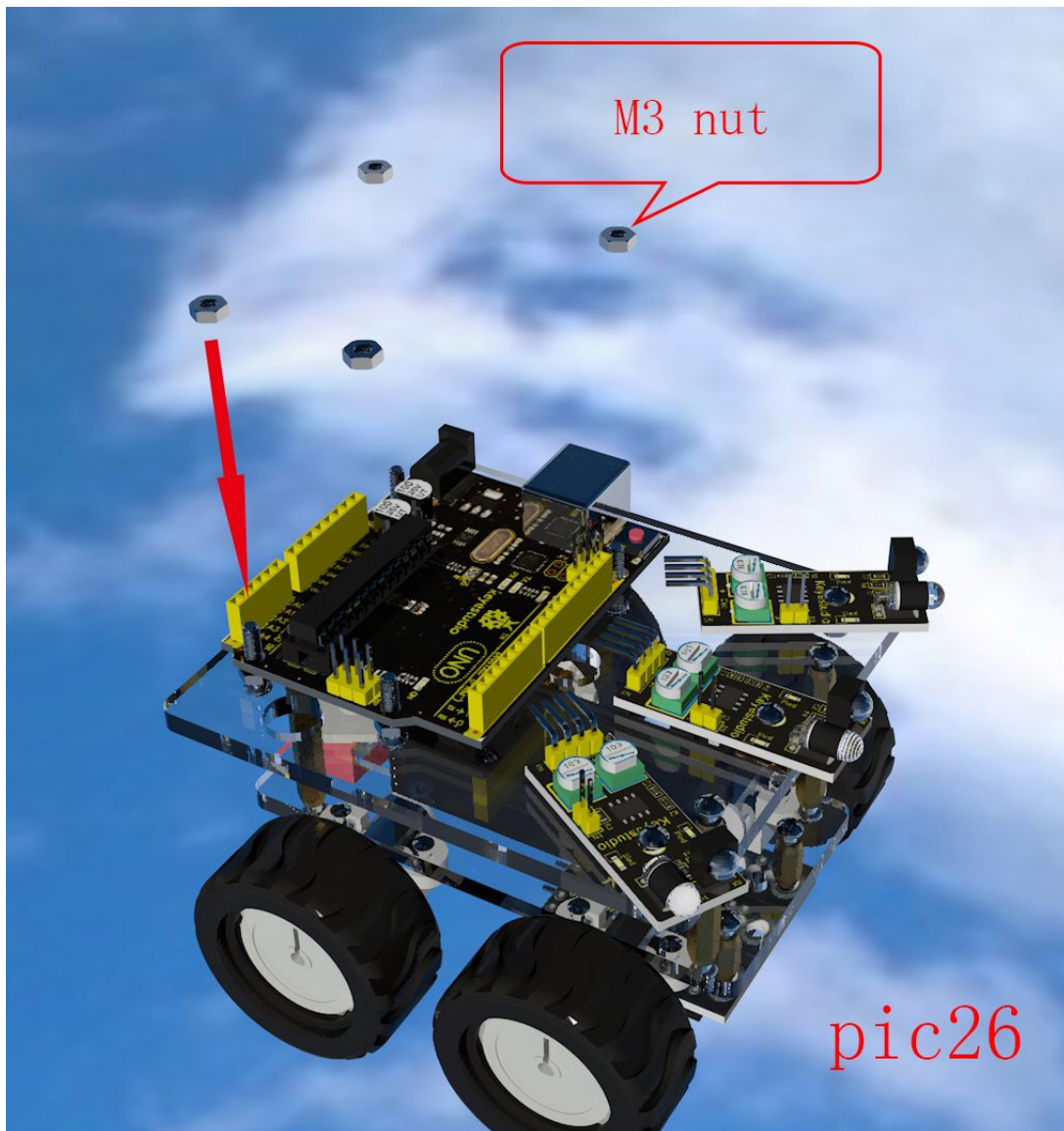
# keystudio

---



# keystudio

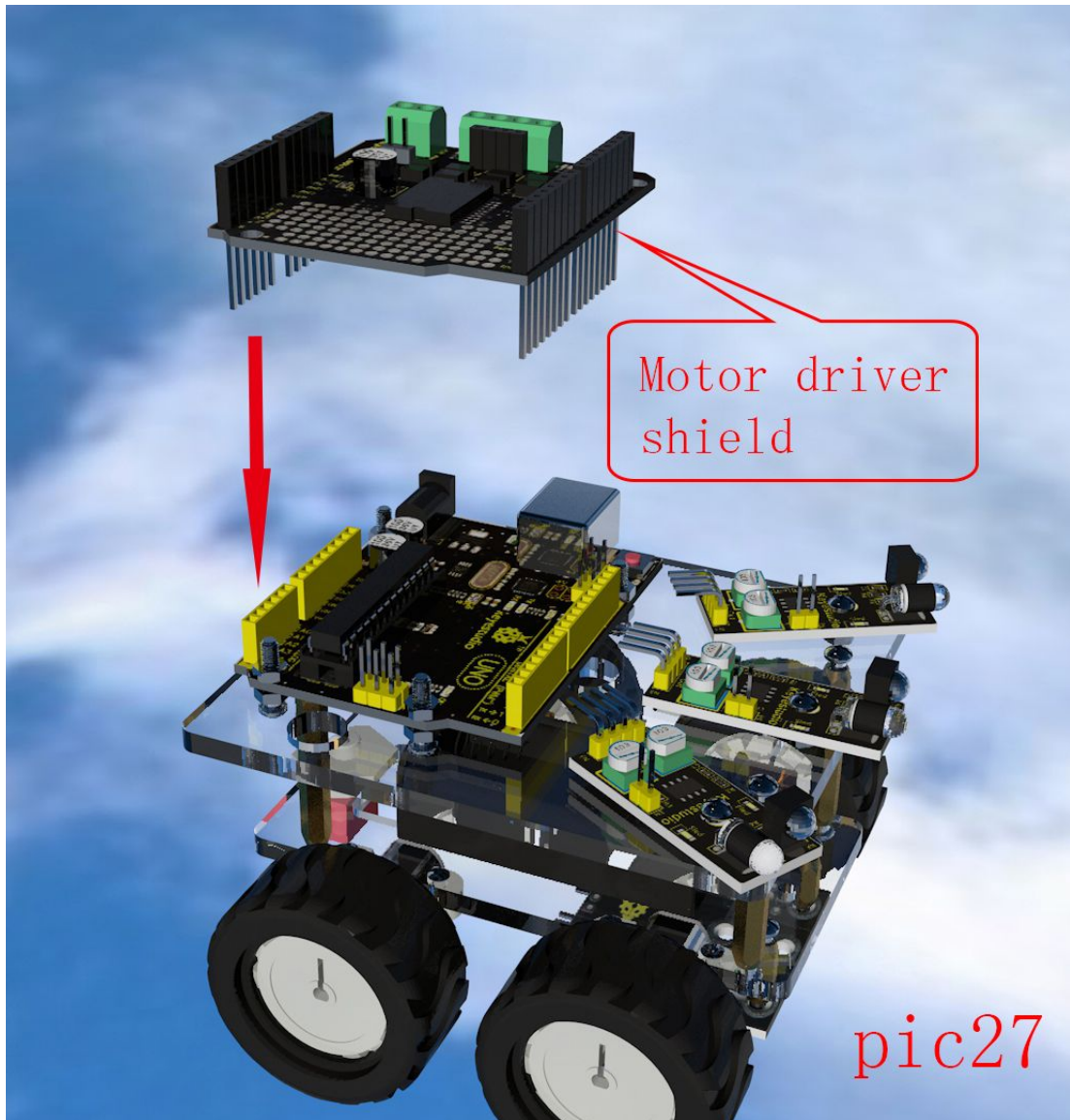
---





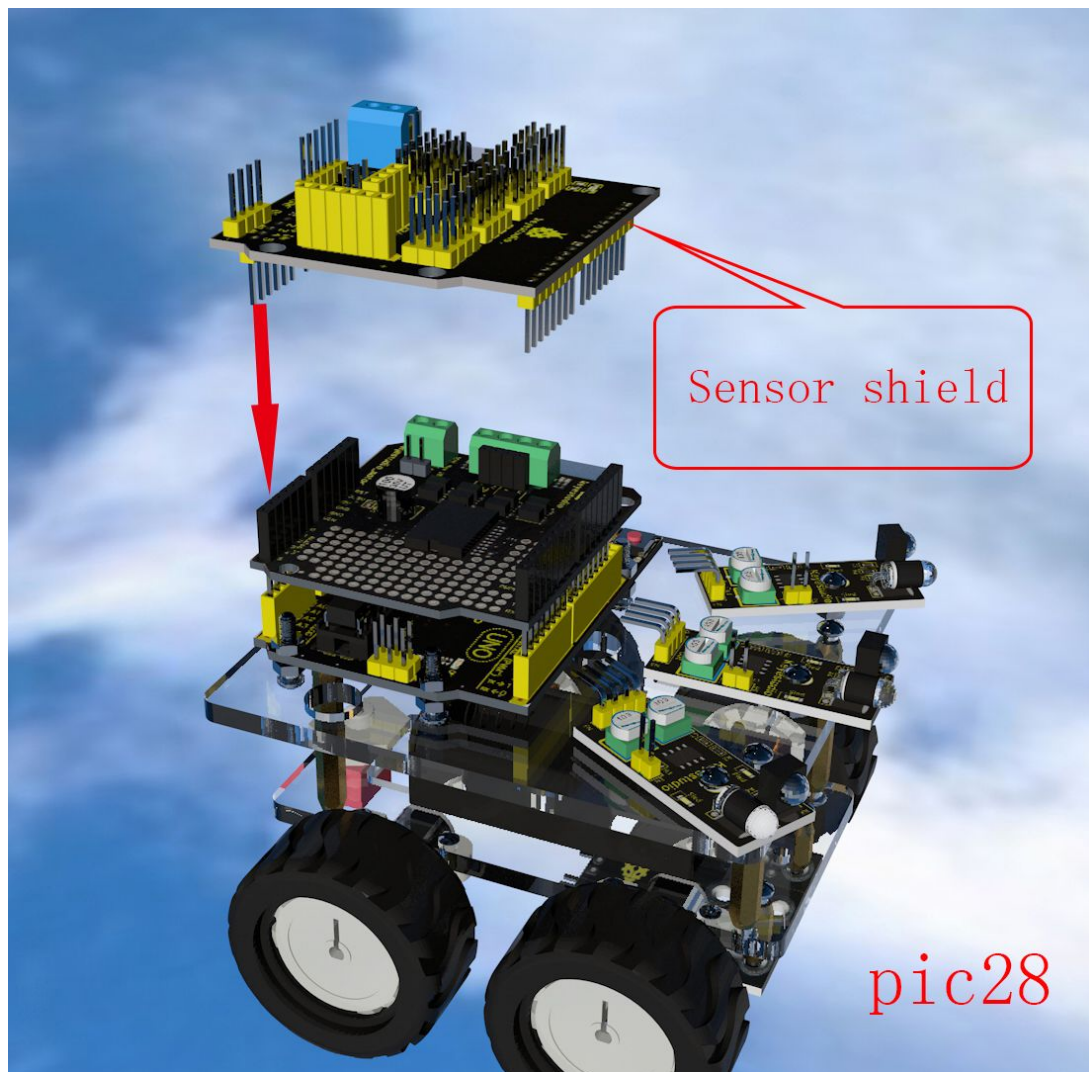
# keystudio

---



# keystudio

---





# keystudio

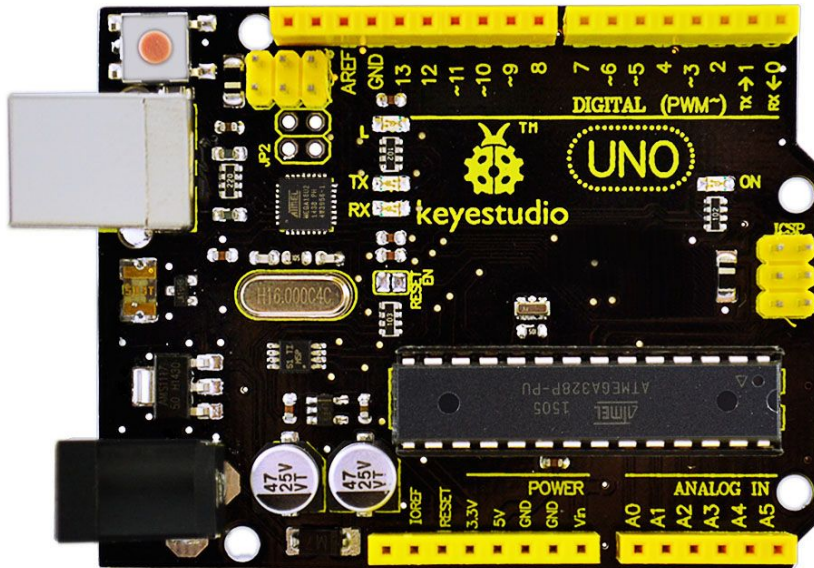
---



# keystudio

---

## 5. Application of Arduino



### Introduction

What's Arduino?

Arduino is an open-source hardware project platform. This platform includes a circuit board with simple I/O function and program development environment software. It can be used to develop interactive products. For example, it can read signals of multiple switches and sensors, and control light, servo motor and other various physical devices. It's widely applied in robot field.

### Arduino installation and program upload:

First, download the Arduino development software, click below hyperlink:

[arduino-1.5.6-r2-windows.rar](#)

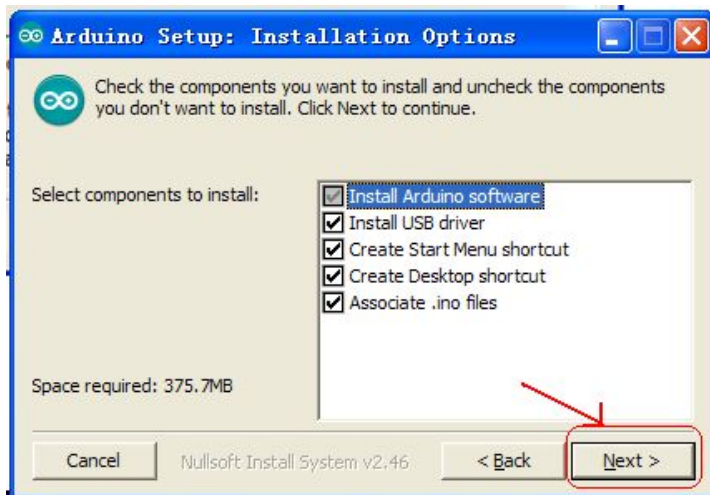
Downloaded file is a arduino-1.5.6-r2-windows.zip compressed folder, unzip it to your hard drive.

Double click Arduino-1.5.6 .exe. Click "I agree";

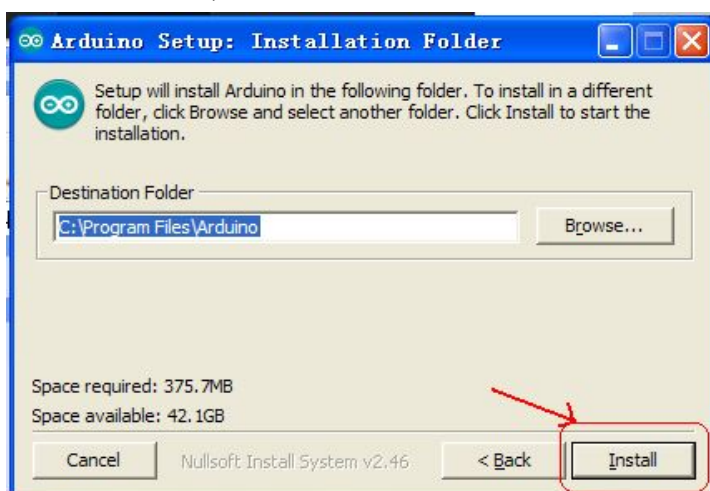
# keyestudio



Click "Next";

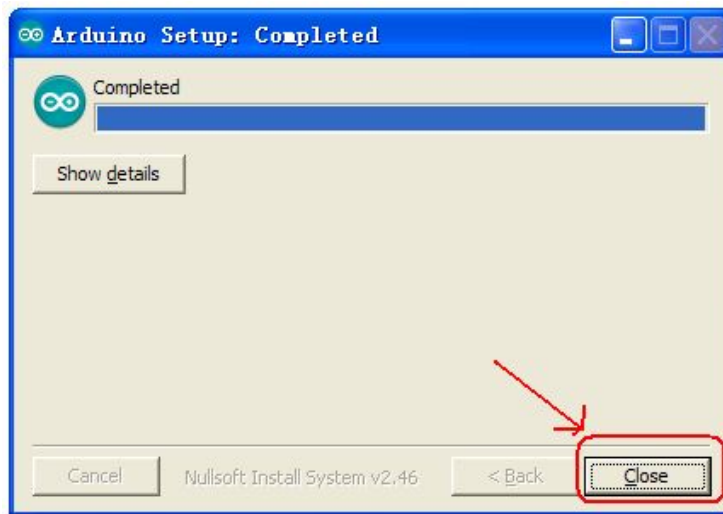


And then "Install";

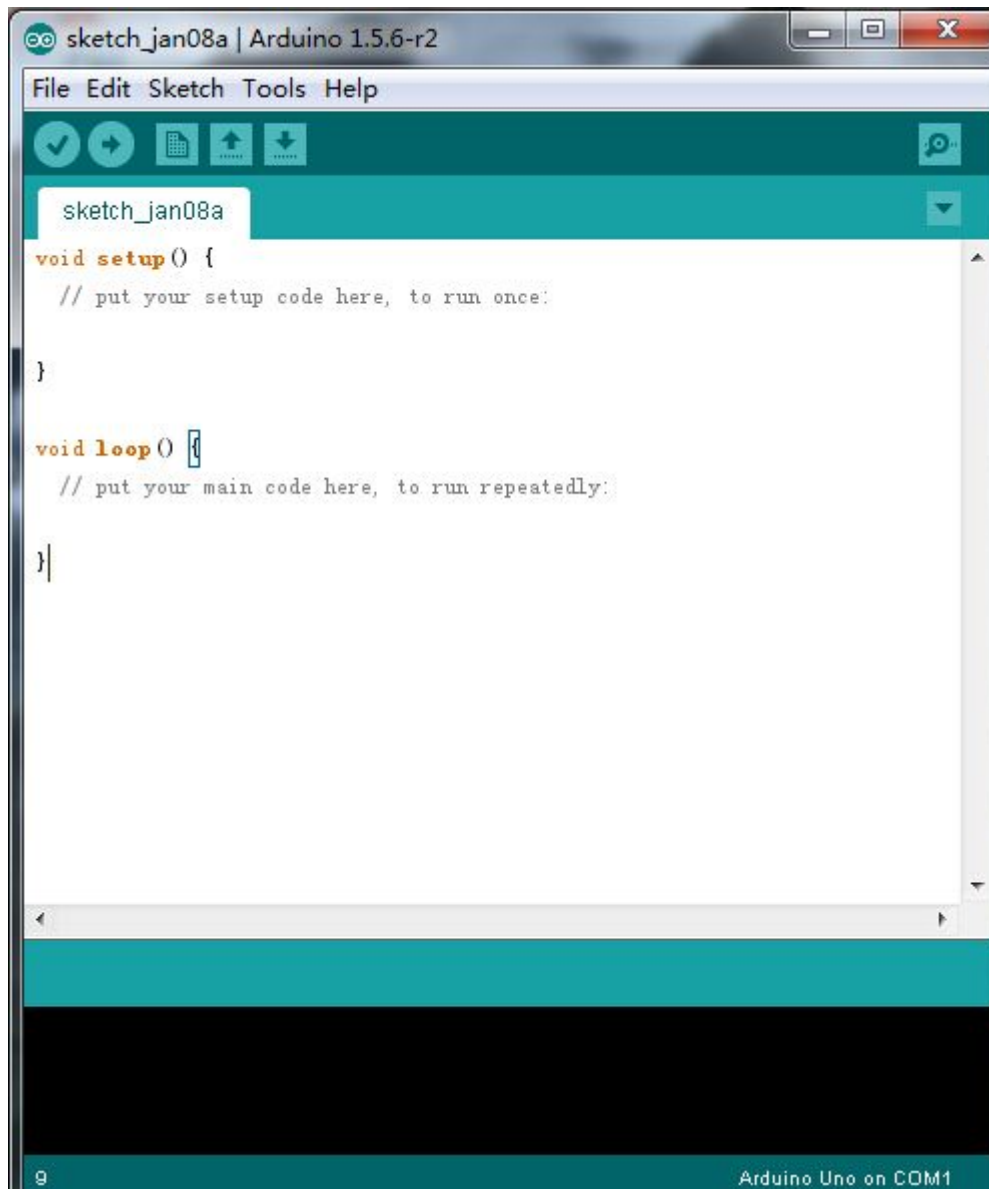


Wait for the installation to be completed, click close.

# keystudio



Below is how Arduino 1.5.6 looks like.

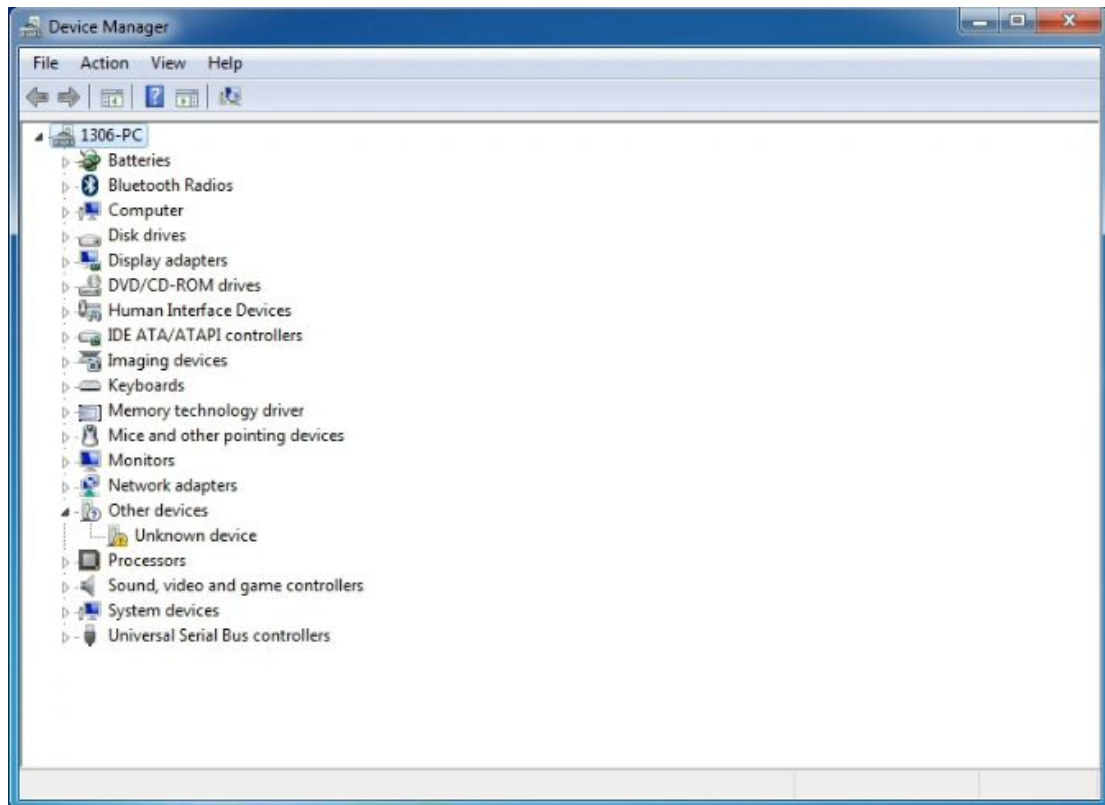


# keystudio

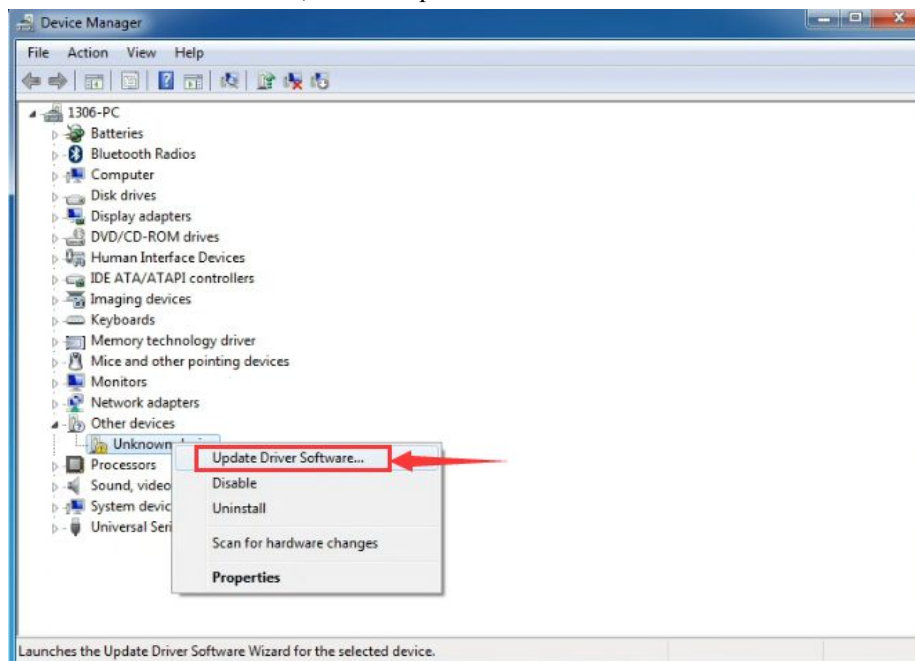
Next, let's install Arduino driver.

For different operating system, there may be slight difference in installation method. Below is an example in WIN 7.

When you connect Arduino Uno to your computer the first time, right click "Computer" → "Properties" → "Device manager", you can see "Unknown devices".



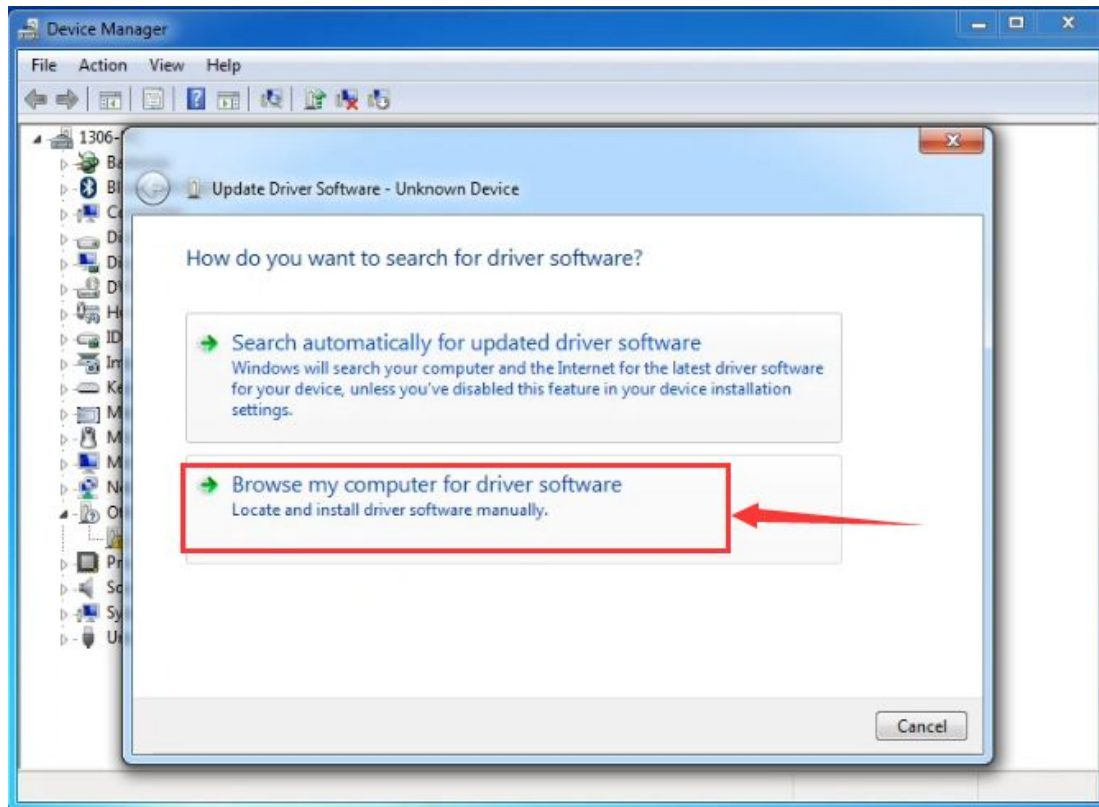
Click "Unknown devices", select "Update Driver software".



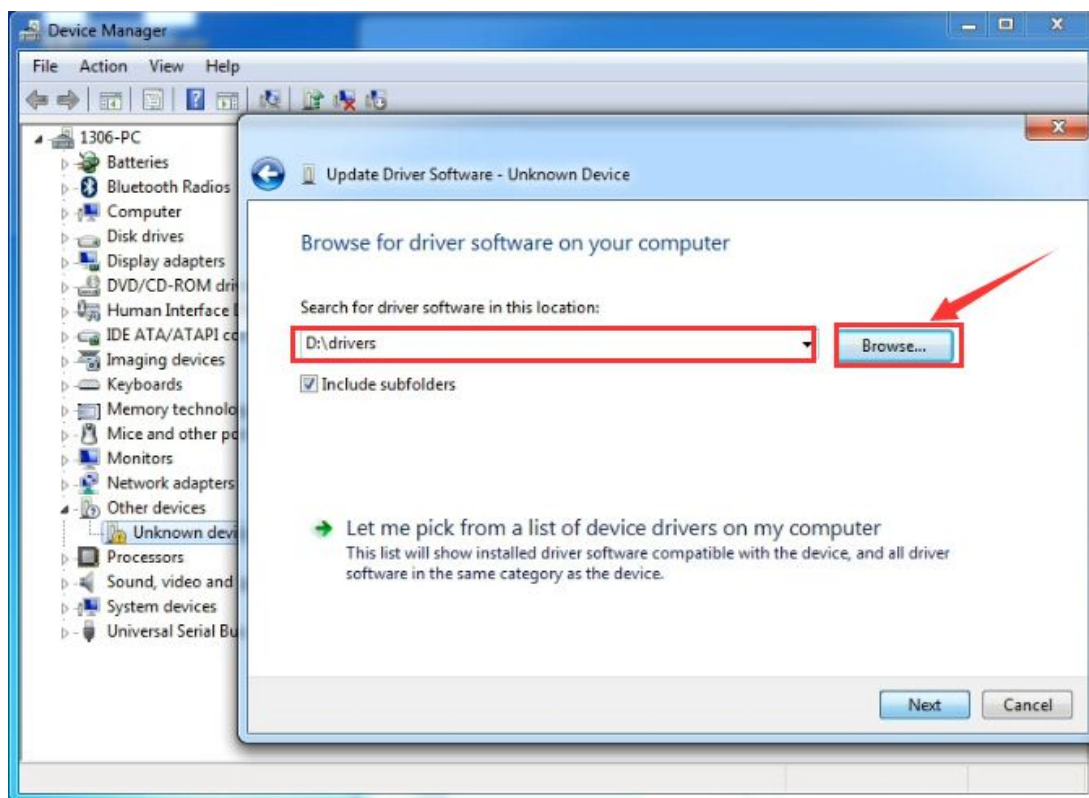


# keystudio

In this page, click “Browse my computer for driver software”.



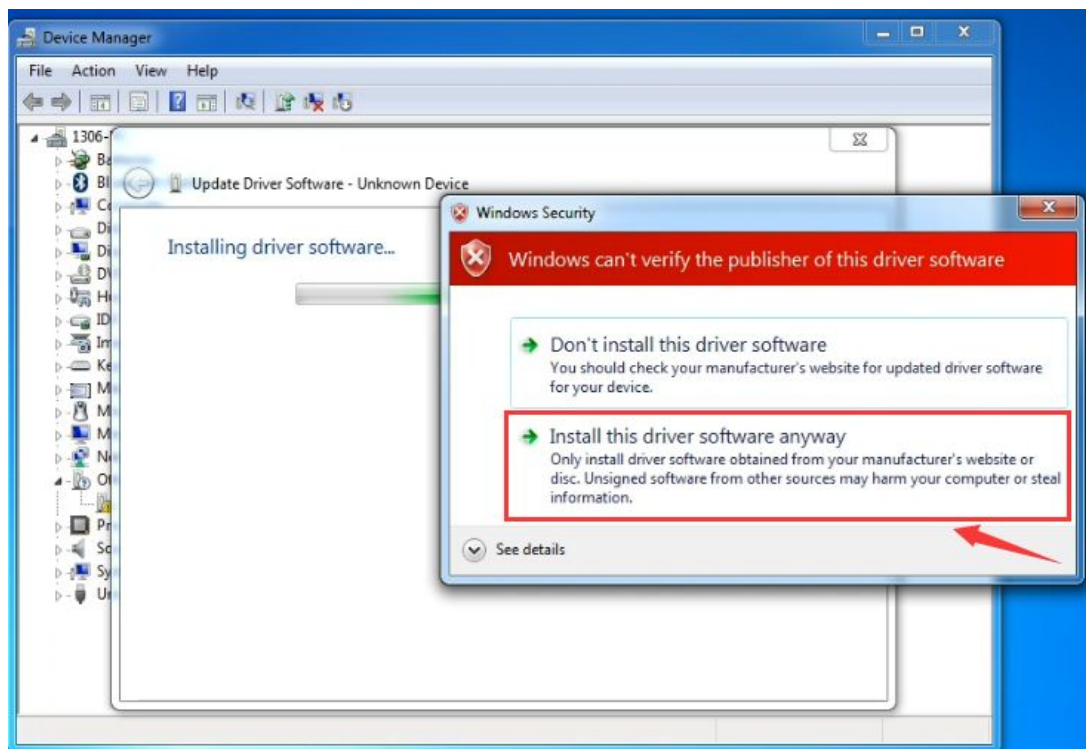
Find the “drivers” file.



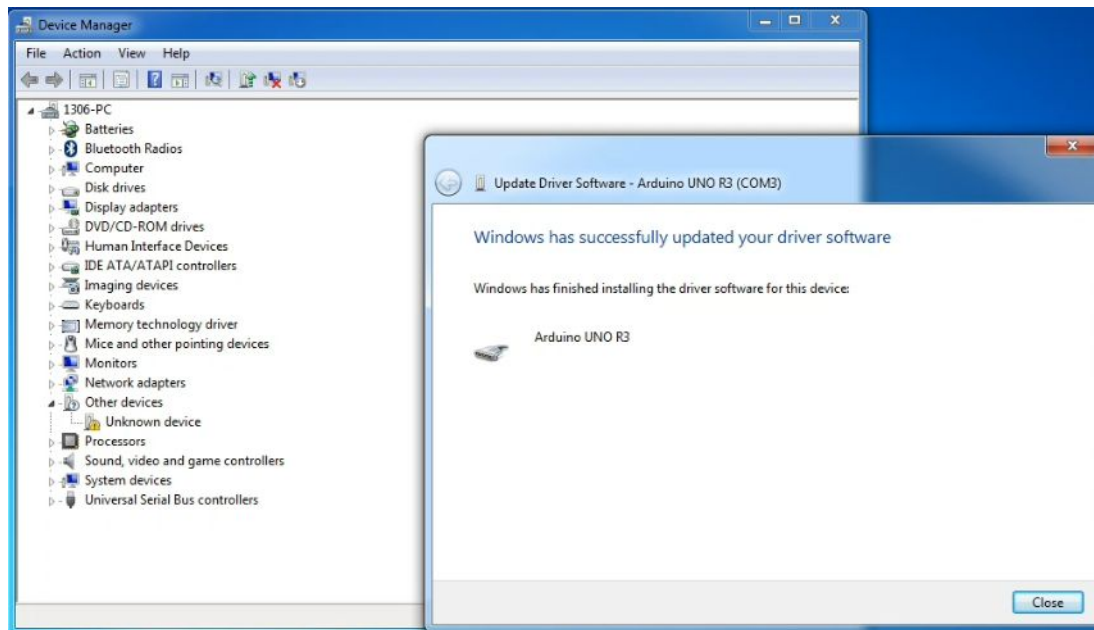
Click “Next”; select “Install this driver software anyway” to begin the installation.



# keystudio

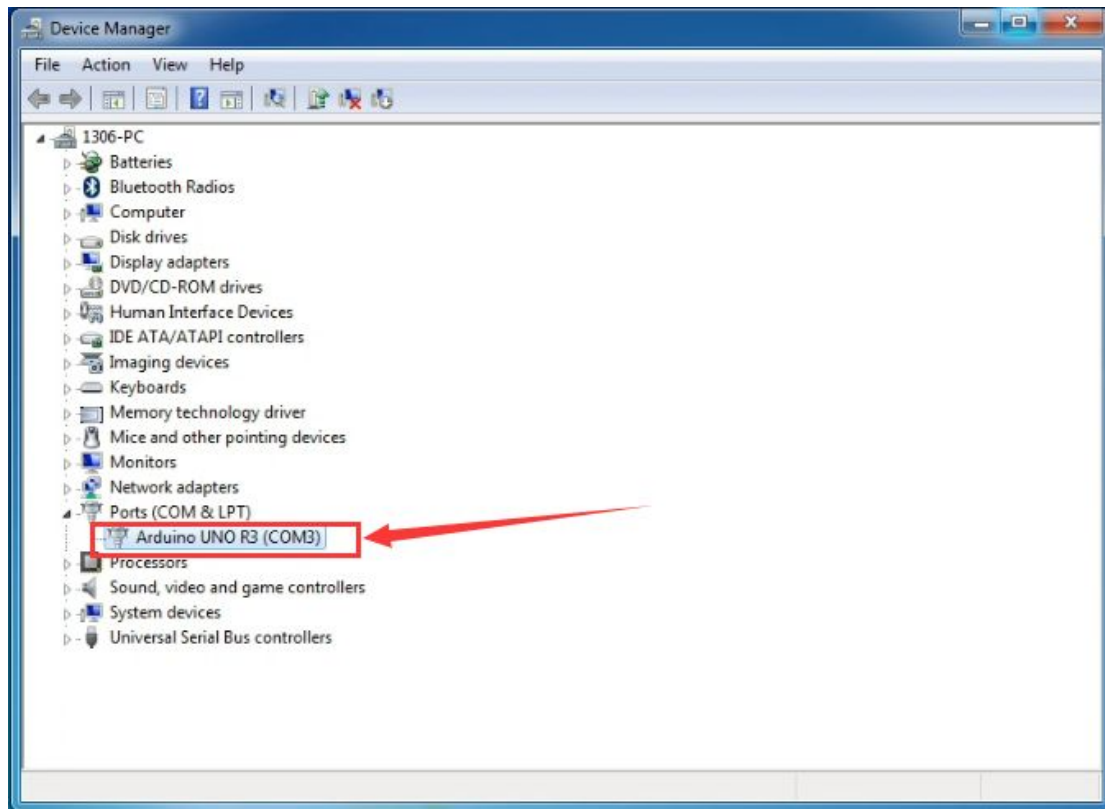


Installation completed; click "Close".



After driver is installed, go to "Device manager" again. Right click "Computer" → "Properties" → "Device manager", you can see UNO device as below figure shows, also the Com port info.

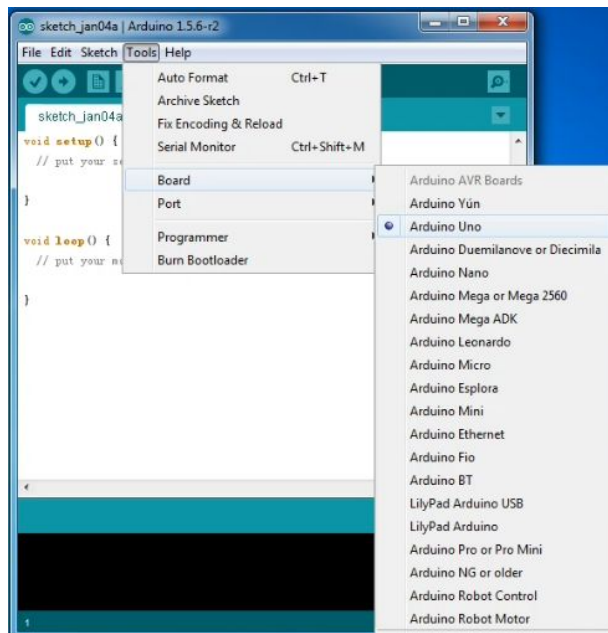
# keystudio



Following is a sketch uploading example called “Hello World! ”.

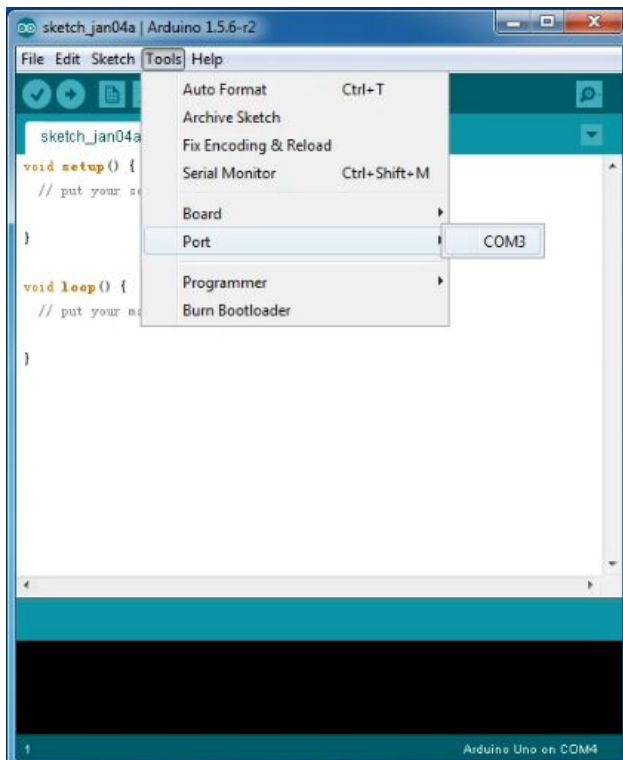
First, open Arduino IDE. In this example sketch, we program Arduino to display “Hello World! ” in serial monitor when it receives a specific character string “R”; also the on-board D13 LED will blink once each time it receives “R”.

First, set up board; In “Tools”, select “Arduino Uno”.



Next, set up COM port; In “Tools”, select “COM3”.


# keystudio



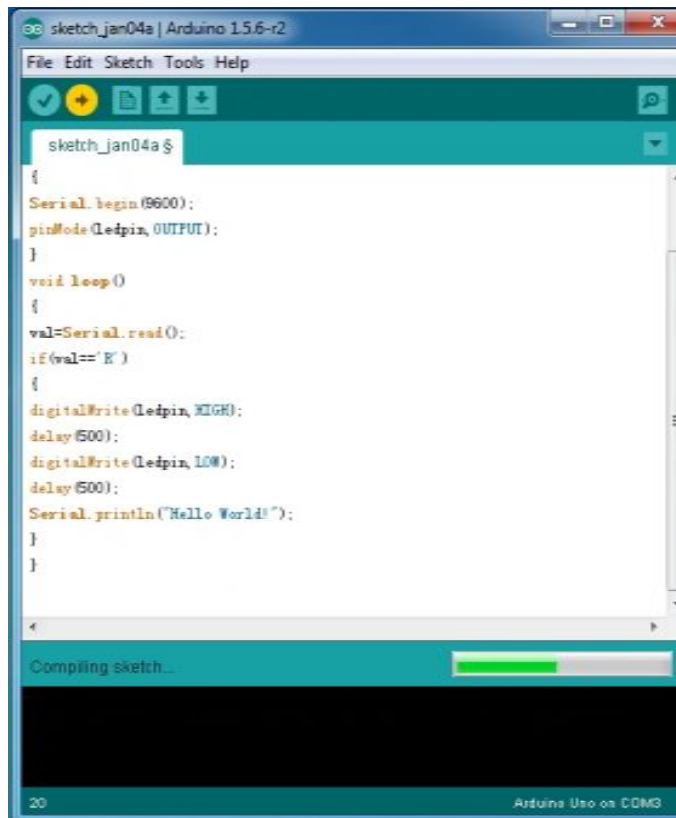
After selection, you can see indicated area is the same with settings in “Device manager”.




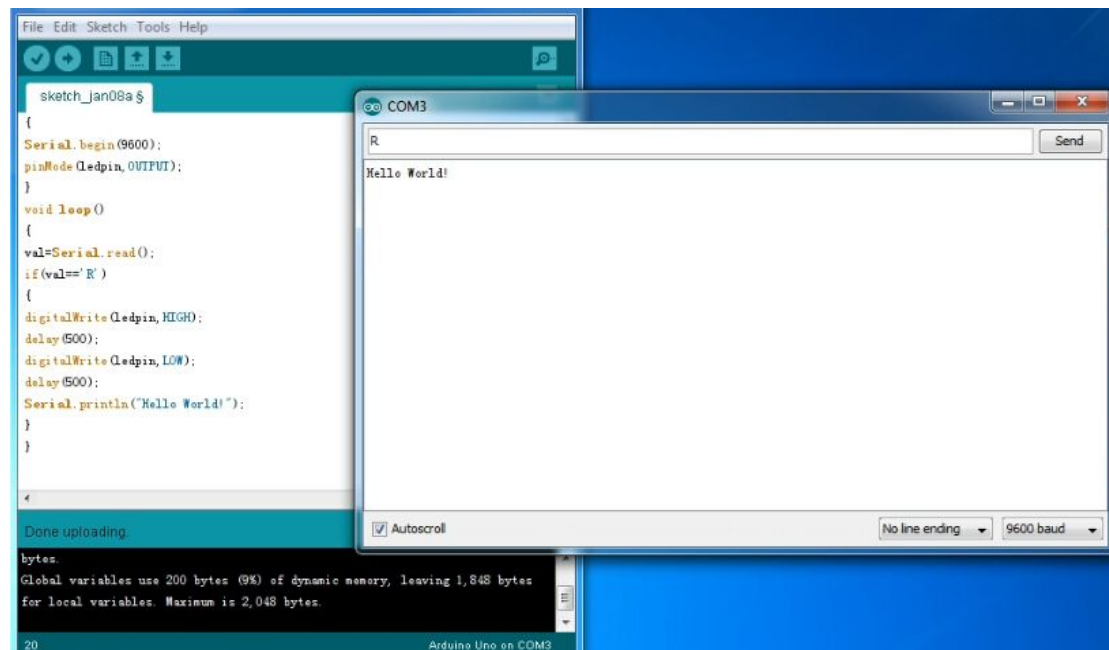
# keystudio

Copy the example sketch and paste it to the IDE; click “Verify ” to check compiling mistakes;

click “Upload ” to upload the program to the board.



After uploading is done, open “serial monitor ”; enter “R”; click “Send”, the serial monitor will display “Hello World!” and the D13 LED will blink once.



Congratulations! Your first sketch uploading is a success!

# keystudio

---

## 6. Project details

### Project 1: Line-tracking smart car

#### Introduction

This project is a simple line-tracking smart car system based on Arduino. The controller part is a UNO board. An infrared photoelectric sensor is used to detect black line, and feedback the signal to UNO. UNO will analyze the signal to determine and control the motors movement to adjust car moving direction. Therefore the smart car can automatically move towards a black line.

#### Working principle

1. For color black, it has characteristic of low reflectivity to light. When the surface color is not black, most of the red light the sensor sends will be reflected. The sensor outputs low level signal 0.
2. If there is black line on a surface, when the sensor is above it, color black has low reflectivity; reflected infrared light is not enough to reach the sensor's action level, so the sensor output is still 1.
3. To detect the black line, we only need a MCU to determine whether the sensor output is 0 or 1.
4. The MCU will control the car moving action according to the signal received.

#### Main hardware introduction



In the line tracking sensor, TCRT5000 infrared tubes are applied. The working principle is based on the infrared's different reflectivity of different color. The strength of the reflected signal is then converted into current signal. For this line tracking module, it's high level signal when detecting black line, low level when detecting white line. Detection height is 0-3cm.

Note: you can adjust the potentiometer knob in the circuit to adjust the sensitivity of this line tracking module.

#### Performance parameter:

1. Detection height: 3cm when detecting black line on white background; the detection height varies with background color; highest in white background.



# keystudio

---

2. Power supply voltage: 2.5V to 12V, no more than 12V. (Note: it is better to use low voltage power supply, high power supply voltage will shorten service life of the sensor. 5V power supply is preferred.)
3. Working current: 18~20ma under 5V power supply. After testing, when the working current is 18~20ma, the sensor is at its best performance in anti-jamming capability.
4. When it detects black line, the signal end outputs high level signal 1; when no black line detected, the signal end outputs low level 0.
5. The sensor outputs TTL level signal; can be directly connected to the IO port of 3.3V or 5V MCU.

## Usage:

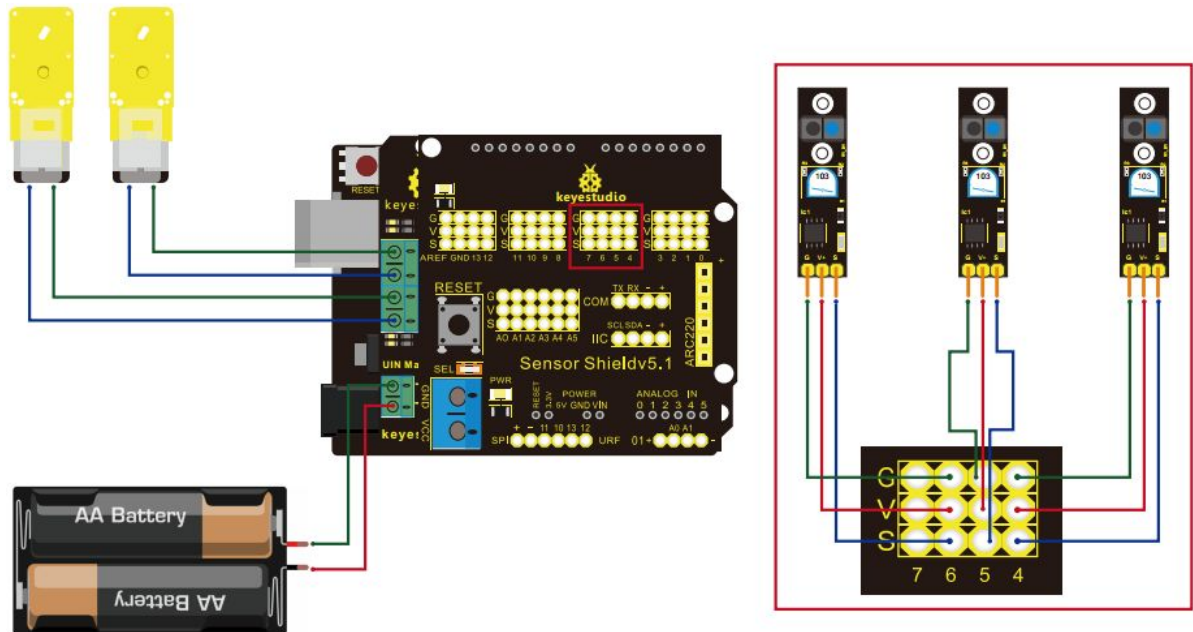
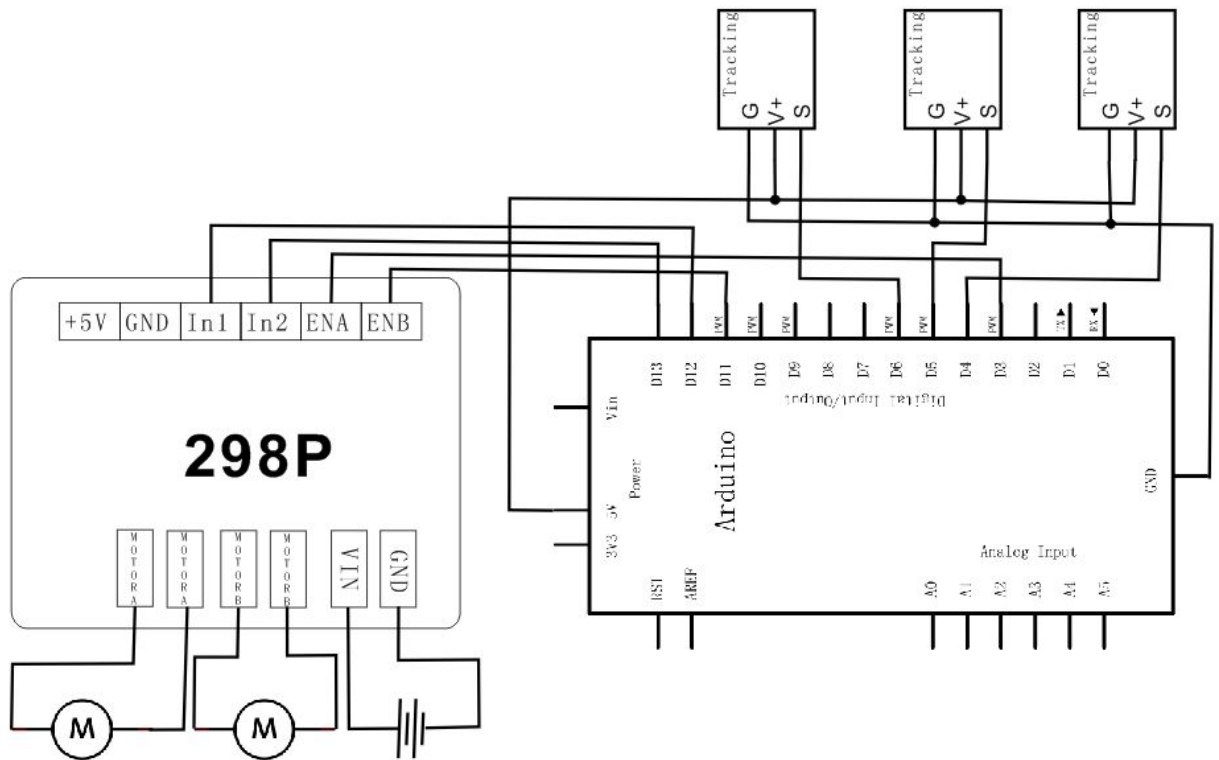
1. The sensor has 3 pins, namely G, +V and S. G and +V are for power supply. S is signal output pin.
2. When it detects black line, the signal end outputs high level signal 1; when no black line detected, the signal end outputs low level 0.
3. From the output 0 or 1, the sensor can determine whether the color is black or not.

## Module test program:

```
///Arduino Sample Code
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.println(digitalRead(3)); // print the data from the sensor
  delay(500);
}
```

## Schematic and connection diagram

# keystudio



## Line tracking car program

```
#define SensorLeft 4 // left sensor input pin
#define SensorMiddle 5 // middle sensor input pin
#define SensorRight 6 // right sensor input pin
unsigned char SL; // left sensor status
unsigned char SM; // middle sensor status
unsigned char SR; // right sensor status
```

# keystudio

---

```
#define E1 3 // speed control pin, ENA pin on motor driver shield
#define E2 11 // speed control pin, ENA pin on motor driver shield
#define M1 12 // motor direction control, IN1 pin on motor driver shield
#define M2 13 // motor direction control, IN2 pin on motor driver shield
void Sensor_IO_Config()
{
    pinMode(SensorLeft,INPUT);
    pinMode(SensorMiddle,INPUT);
    pinMode(SensorRight,INPUT);
}
void Sensor_Scan(void)
{
    SL = digitalRead(SensorLeft);
    SM = digitalRead(SensorMiddle);
    SR = digitalRead(SensorRight);
}
void M_Control_IO_config(void)// initialization function of motor driver shield IO
{
    pinMode(M1,OUTPUT);
    pinMode(M2,OUTPUT);
    pinMode(E1,OUTPUT);
    pinMode(E2,OUTPUT);
}

void advance(void) // move forward
{
    digitalWrite(M1,LOW); // wheel on the right moves forward
    digitalWrite(M2, LOW); // wheel on the left moves forward
    analogWrite(E1,150);
    analogWrite(E2, 150);
}
void turnR(void) // turn right
{
    digitalWrite(M1,LOW); // wheel on the left moves forward
    digitalWrite(M2,HIGH); // wheel on the right moves backward
    analogWrite(E1,150);
    analogWrite(E2, 150);
}
void turnL(void) // turn left
{
    digitalWrite(M1,HIGH); // wheel on the left moves backward
    digitalWrite(M2, LOW); // wheel on the right moves forward
    analogWrite(E1,150);
```

# keystudio

---

```
    analogWrite(E2, 150);
}
void stopp(void)           // stop
{
    digitalWrite(M1,LOW);
    digitalWrite(M2, LOW);
    analogWrite(E1, 0);
    analogWrite(E2, 0); // both right &left wheel stop
}
void back(void)           // move backwards
{
    digitalWrite(M1,HIGH); // both right &left wheel moves backwards
    digitalWrite(M2, HIGH);
    analogWrite(E1,150);
    analogWrite(E2, 150);
}

void setup()
{
    Sensor_IO_Config();
    M_Control_IO_config(); // initialization of motor controller module IO
    stopp();
}
void loop()
{
    Sensor_Scan();
    if((SL==1&&SM==1&&SR==1)||((SL==1&&SM==1&&SR==0)||((SL==0&&SM==1&&SR==1)||((SL
==0&&SM==1&&SR==0))advance());
    if((SL==0&&SM==0&&SR==0)||((SL==0&&SM==0&&SR==1))turnL();
    if(SL==1&&SM==0&&SR==0)turnR();
}
*****
```

When the car enters line tracking mode, it begins constantly scanning I/O port of the MCU that connects the detecting sensor. When it picks up a signal of the I/O port , it will enter the judgment processing; firstly determine which sensor detects the black line.

## Project 2: Obstacle-avoidance smart car

### Introduction

This project is a simple obstacle avoidance smart car system based on Arduino. The controller part is a UNO board. An obstacle avoidance sensor is used to detect whether there is obstacle ahead,

# keystudio

---

and feedback the signal to UNO. UNO will analyze the signal to determine and control the motors movement to adjust car moving direction. Therefore the smart car can automatically avoid obstacles.

## Working principle

1. Obstacle avoidance sensor is applied here. The sensor has a pair of infrared transmitting and receiving tube. When infrared ray launched by the transmitting tube encounters an obstacle (its reflector), the infrared ray is reflected to the receiving tube. The reflecting light strength is then converted into current signal.
2. When it detects an obstacle ahead, the sensor will output low level 0; when there is no obstacle detected, the sensor outputs high level signal 1.
3. To detect an obstacle, we only need a MCU to determine whether the sensor output is 0 or 1.
4. The MCU will control the car moving action according to the signal received.

## Main hardware introduction



Infrared obstacle avoidance sensor is equipped with a pair of infrared transmitting and receiving tube. When infrared ray launched by the transmitting tube encounters an obstacle (its reflector), the infrared ray is reflected to the receiving tube. The reflecting light strength is then converted into current signal. When it detects an obstacle, it outputs low level signal 0; when no obstacle is detected, it outputs high level signal 1.

Note: you can adjust the potentiometer knob in the circuit to adjust the sensitivity of this line tracking module.

## Performance parameter:

1. When it detects an obstacle, it outputs low level signal 0; when no obstacle is detected, it outputs high level signal 1. The sensor can be directly connected to the IO port of 3.3V or 5V MCU.
2. Detection distance is 2-40cm, long distance and high precision. (Note: for infrared sensor, the detection distance varies with difference in object color because different color has different reflectivity of light, the darker the object, the shorter the detection distance. The distance 2-40cm



# keystudio

---

is calculated when the sensor is facing a white wall.)

3. Quick response, suitable for smart car obstacle avoidance, black and white line tracking, fall prevention, liquid level detection, etc.
4. Wide power supply range of 3-6V, suitable for using with 3.3-5V micro-controller.
5. Equipped with EN pin, when this pin output is "1", the sensor is not operating; when the output is "0", the sensor is. When the wire jumper on the module is put on, the EN pin is connected to GND and the output is "0". If you want to use the EN pin, just take away the wire jumper.
6. Frequency adjusting potentiometer is used to adjust the carrier frequency of the infrared transmitting tube, because the integrated receiver is the most sensitive when in 38 KHZ carrier frequency. Adjust the 502 pot's clockwise to the upmost (that is when the transmitting tube is the brightest); then face the sensor to a white wall and adjust the 103 pot's until the detection distance is the furthest. Now the sensor is working at its best state.

## Usage:

1. The sensor has 4 pins, namely GND, +, OUT, and EN. + and GND is for power supply; OUT is signal output pin. When in actual use, you only need to +, GND and OUT.
2. When an obstacle is detected, the signal pin outputs low level signal 0, when no obstacle detected, signal end outputs high level signal 1.
3. By determining whether the sensor output is 0 or 1, it can determine whether there is an obstacle ahead.

## Module test program:

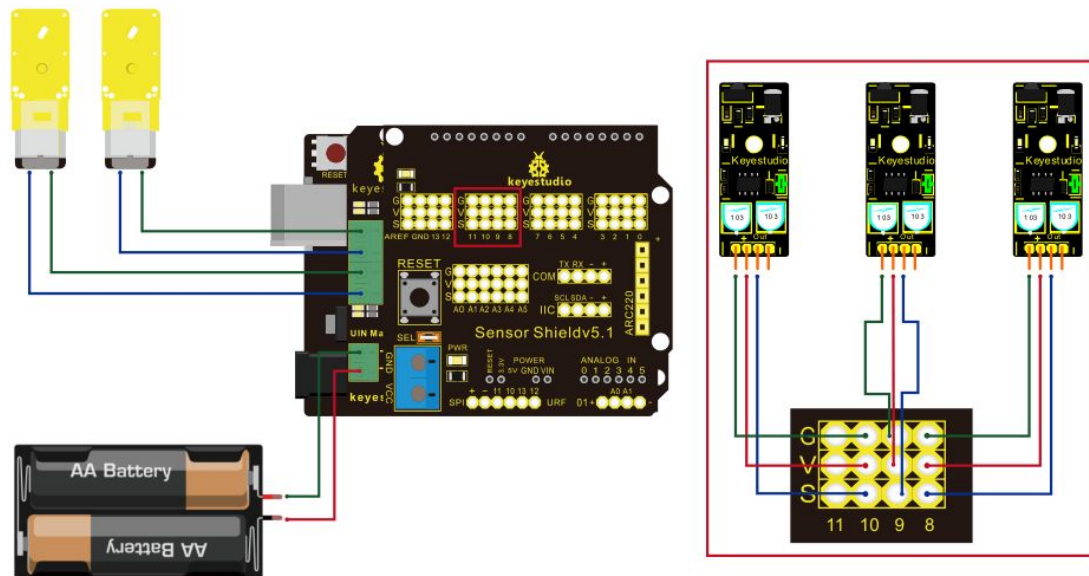
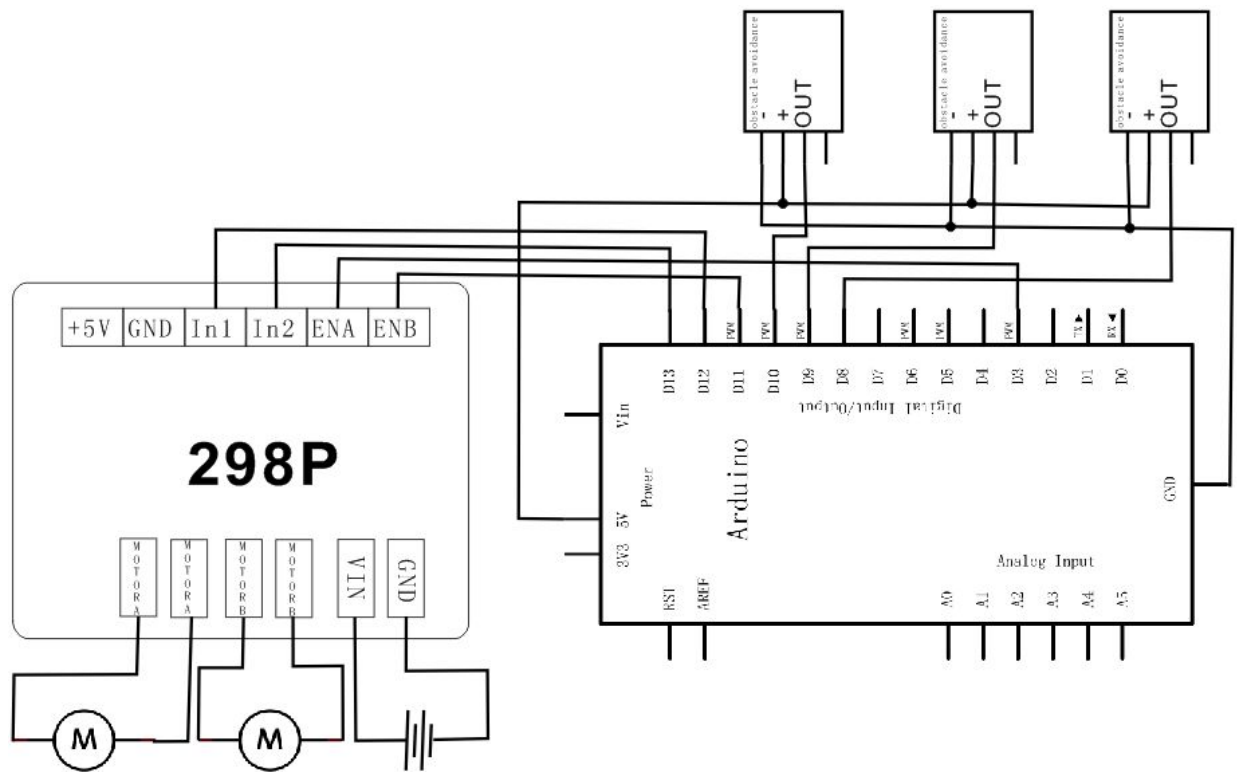
```
const int sensorPin = 2;    // the number of the sensor pin
const int ledPin = 13;     // the number of the LED pin
int sensorState = 0;       // variable for reading the sensor status

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(sensorPin, INPUT); }

void loop(){
  // read the state of the sensor value:
  sensorState = digitalRead(sensorPin);
  // if it is, the sensorState is HIGH:
  if (sensorState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

## Schematic and connection diagram

# keystudio



## Obstacle avoidance car program

```
#define IRSensorLeft    10 // left sensor input pin
#define IRSensorMiddle  9  // middle sensor input pin
#define IRSensorRight   8  // right sensor input pin
unsigned char IRSL;      // left sensor status
unsigned char IRSM;     // middle sensor status
unsigned char IRSR;     // right sensor status
#define E1    3
```

# keystudio

---

```
#define E2  11
#define M1  12
#define M2  13
void Sensor_IO_Config()
{
    pinMode(IRSensorLeft,INPUT);
    pinMode(IRSensorMiddle,INPUT);
    pinMode(IRSensorRight,INPUT);
}
void Sensor_Scan(void)
{
    IRSL = digitalRead(IRSensorLeft);
    IRSM = digitalRead(IRSensorMiddle);
    IRSR = digitalRead(IRSensorRight);
}
void M_Control_IO_config(void)// initialization function of motor driver shield IO
{
    pinMode(M1,OUTPUT);
    pinMode(M2,OUTPUT);
    pinMode(E1,OUTPUT);
    pinMode(E2,OUTPUT);
}

void advance(void)    // move forward
{
    digitalWrite(M1,LOW);    // right wheel moves forward
    digitalWrite(M2, LOW); // left wheel moves forward
    analogWrite(E1,150);
    analogWrite(E2, 150);
}
void turnR(void)      // turn right
{
    digitalWrite(M1,LOW); // left wheel moves forward
    digitalWrite(M2,HIGH); // right wheel moves backward
    analogWrite(E1,150);
    analogWrite(E2, 150);
}
void turnL(void)      // turn left
{
    digitalWrite(M1,HIGH); // left wheel moves backwards
    digitalWrite(M2, LOW); // right wheel moves forward
    analogWrite(E1,150);
    analogWrite(E2, 150);
}
```

# keystudio

---

```
}
void stopp(void)           // stop
{
    digitalWrite(M1,LOW);
    digitalWrite(M2, LOW);
    analogWrite(E1, 0);
    analogWrite(E2, 0); // both left and right wheels stop
}
void back(void)           // move backward
{
    digitalWrite(M1,HIGH); // both left and right wheels move forward
    digitalWrite(M2, HIGH);
    analogWrite(E1,150);
    analogWrite(E2, 150);
}

void setup()
{
    Sensor_IO_Config();
    M_Control_IO_config(); // initialization of motor driver shield IO
    stopp();
}
unsigned char old_IRSL,old_IRSM,old_IRSR;
void loop()
{
    Sensor_Scan();
    if(IRSL==1&&IRSM==1&&IRSR==1)advance();
    if((IRSL==0&&IRSM==0&&IRSR==1)||((IRSL==0&&IRSM==1&&IRSR==1)||((IRSL==1&&IRSM==0&&IRSR==1))turnL();
    if((IRSL==0&&IRSM==0&&IRSR==0)||((IRSL==1&&IRSM==0&&IRSR==0)||((IRSL==1&&IRSM==1&&IRSR==0))turnR();
}
}
```

## Project 3: Bluetooth smart car

### Introduction

This project is a smart car system based on Bluetooth communication, including software and hardware design. The controller part is a UNO board. A Bluetooth module is used to receive the Bluetooth signal from the cellphone and pass on to the UNO. UNO will analyze the signal to determine and control the motors movement to adjust car moving direction. Therefore the smart car can be controlled by cellphone.



# keystudio

---

## Working principle

1. The UNO is connected to a Bluetooth module; the module communicates with cell phone through a Bluetooth APP.
2. The Bluetooth APP on the cell phone will pass information of “U”“D”“L”“R”“S” to the Bluetooth module.
3. The Bluetooth module will pass the information to the UNO, so the UNO can determine car movement according to the information received.
4. When the UNO receives a “U”, the car goes straight forward; when it receives a “D”, the car goes backward; “L” for turning left; “R” for turning right; and “S” for stop.

## Main hardware introduction



### Bluetooth communication:

The term “Bluetooth” is derived from a Denmark king’s name in 10th century. The king is named Harald Blatand, while “Blatand” in English means Bluetooth.

The so-called bluetooth technology is actually a type of short distance wireless transmission technology. We can use “Bluetooth” to effectively simplify the communication between terminal devices such as laptop and mobile phone. It can also simplify communication between these devices and the Internet, thus improving the speed and efficiency of data transmission between them, widening the application scope of wireless communication.

### Performance parameter:

1. The Bluetooth module used here is a HC-06 slave module. It has 4 pins namely VCC, GND, TXD and RXD.
2. With an LED indicator to indicate connection status, quick flashing means no Bluetooth connected; steadily on means Bluetooth is connected and the channel is open.
3. Module bottom is equipped with anti-reverse diode, with 3.3V LDO; input voltage between 3.6~6V. When the Bluetooth is not paired, the current is around 30mA; when Bluetooth is paired, current is around 10mA. The input voltage should not be over 7V.
4. Interface level is 3.3V, can be directly connected to various MCU (51, AVR, PIC, ARM,

# keystudio

---

MSP430 etc.), also 5V MCU, no need for MAX232, and can not be used with MAX232.

5. 10m effective distance in open area (power class is CLASS 2). The distance can be further, but the connection quality can not be ensured.
6. Can be used as full duplex com after it's paired, no need to have knowledge of any Bluetooth protocols; supports 8 data bit, 1 stop bit; can set up odd-even check communication format, which is the most commonly used communication format, does not support other formats.
7. Compact design (40mm\*15.5mm), factory SMD production ensures quality; with transparent heat shrinkable film for dust-proof and anti-static.
8. Supports standard baud rate between 4800bps~1382400bps.
9. Module size: 40mm\*15mm

## Usage:

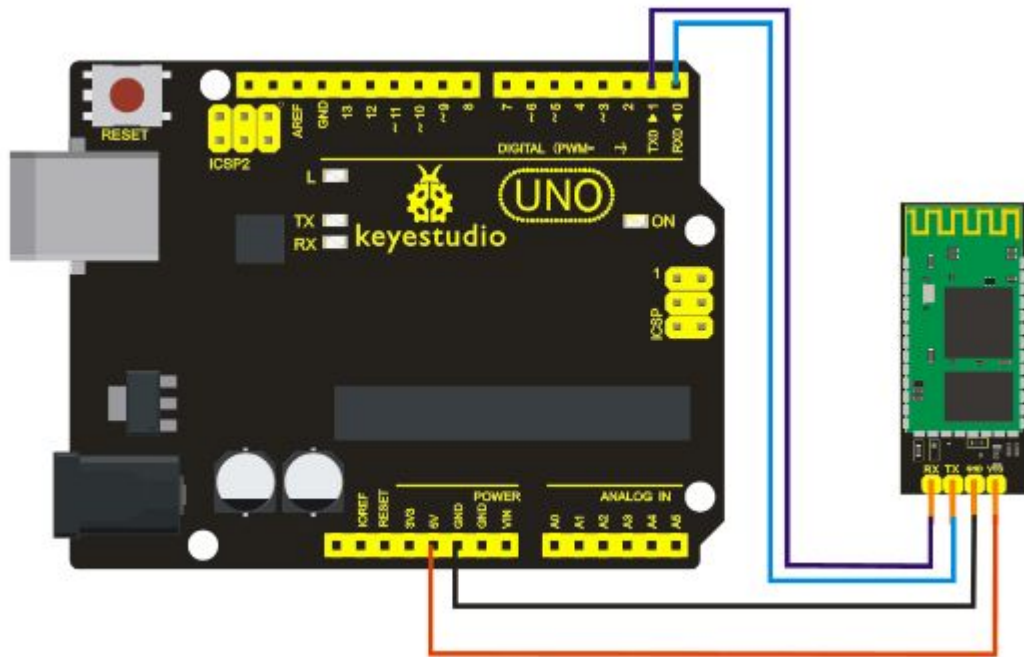
**\*Download the Bluetooth APP [Here](#).**

1. The sensor has 4 pins, GND, VCC, RX and TX. Connect main board +5V to Bluetooth VCC, main board GND to Bluetooth GND, main board TX to Bluetooth RX and RX to Bluetooth TX.
2. Remember to open the Bluetooth on your phone; when you open the Bluetooth APP, it will remind you.
3. Pair up Bluetooth device on your phone, search and pair.
4. Pair up device, PIN No. is 1234.
5. Open Bluetooth APP and pair up Bluetooth device. After it's paired, the Bluetooth module can communicate with cell phone.

## Module test:

Since this is your first try with Bluetooth module, we'll do a simple test of having Arduino to successfully communicate with PC. First is the wire connection. Connect the module to the main board as stated in **usage** (refer to below figure). When the Bluetooth module is successfully connected to PC, the power indicator of the module will blink and connection indicator the green one will be on.

# keyestudio

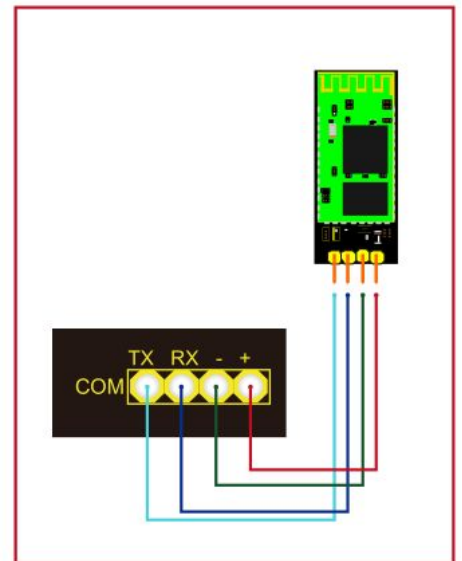
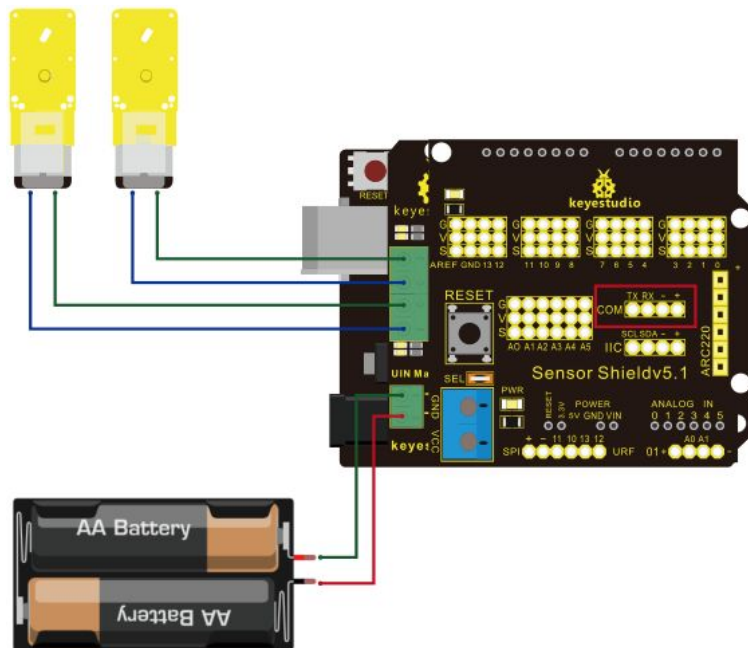
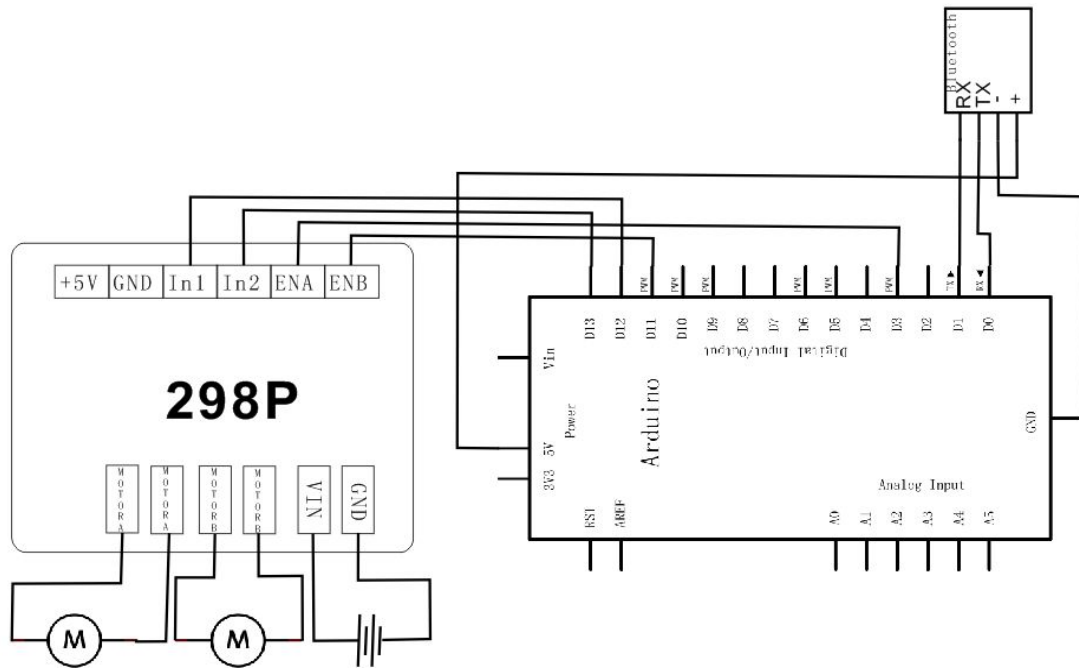


Now, let's move on to the program. I'll enter "r" and after Arduino receives my command "r", the pin13 LED on the main board will blink and serial monitor will print "keys". The program is as follows:

```
char val;
int ledpin=13;
void setup()
{
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
}
void loop()
{
  val=Serial.read();
  if(val=='r')
  {
    digitalWrite(ledpin, HIGH);
    delay((500);
    digitalWrite(ledpin, LOW);
    delay(500);
    Serial.println("keys");
  }
}
```

**Schematic and connection diagram**

# keystudio



## Bluetooth smart car program

Arduino Bluetooth remote control programmable smart car

```
//*****
```

```
#define E1 3
#define E2 11
#define M1 12
#define M2 13
int val;
```



# keystudio

---

```
void M_Control_IO_config(void)// initialization function of motor driver shield IO
{
  pinMode(M1,OUTPUT); //
  pinMode(M2,OUTPUT); //
  pinMode(E1,OUTPUT); //
  pinMode(E2,OUTPUT); //
}

void advance(void) // move forward
{
  digitalWrite(M1,LOW); // right wheel moves forward
  digitalWrite(M2, LOW); // left wheel moves forward
  analogWrite(E1,150);
  analogWrite(E2, 150);
}

void turnR(void) // turn right
{
  digitalWrite(M1,LOW); // left wheel moves forward
  digitalWrite(M2,HIGH); // right wheel moves forward
  analogWrite(E1,150);
  analogWrite(E2, 150);
}

void turnL(void) // turn left
{
  digitalWrite(M1,HIGH); // left wheel moves backward
  digitalWrite(M2, LOW); // right wheel moves forward
  analogWrite(E1,150);
  analogWrite(E2, 150);
}

void stopp(void) // stop
{
  digitalWrite(M1,LOW);
  digitalWrite(M2, LOW);
  analogWrite(E1, 0);
  analogWrite(E2, 0); // both left and right wheel stop
}

void back(void) // move backward
{
  digitalWrite(M1,HIGH); // both left and right wheel move backward
  digitalWrite(M2, HIGH);
  analogWrite(E1,150);
  analogWrite(E2, 150);
}
```

# keystudio

---

```
void setup()
{
  Serial.begin(9600);
  M_Control_IO_config();      // initialization of motor driver shield IO
  stopp();
}
void loop()
{
  val=Serial.read();
  if(val=='U')advance();
  if(val=='D')back();
  if(val=='L')turnL();
  if(val=='R')turnR();
  if(val=='S')stopp();
}
```