DIY Remote Control Robot Kit

(The best Christmas Gift)

User Manual v1.0

www.DFRobot.com

Contents

Overall 3

Motor Specification 3

Microcontroller 3

Parts List 4

Basic Kit 4

Upgrade Components 4

Additional Parts Required 5

Project Ideas 6

Bluetooth Remote Control 6

IR Remote Control 6

Obstacle Avoidance 6

Assembly 7

Basic Platform 7

Electronics and Wiring 13

Power and switch 13
Motors 13
IR remote control 14
IR sensors 14
Servo 15
URM37V3.2 Ultrasonic sensor 15
7LEDs disc 16
Totals 17

Support 18

Arduino Code 18

Bluetooth Remote Control 18

IR Remote Control 23

Obstacle Avoidance 26

Standard PWM DC control 29

Overall

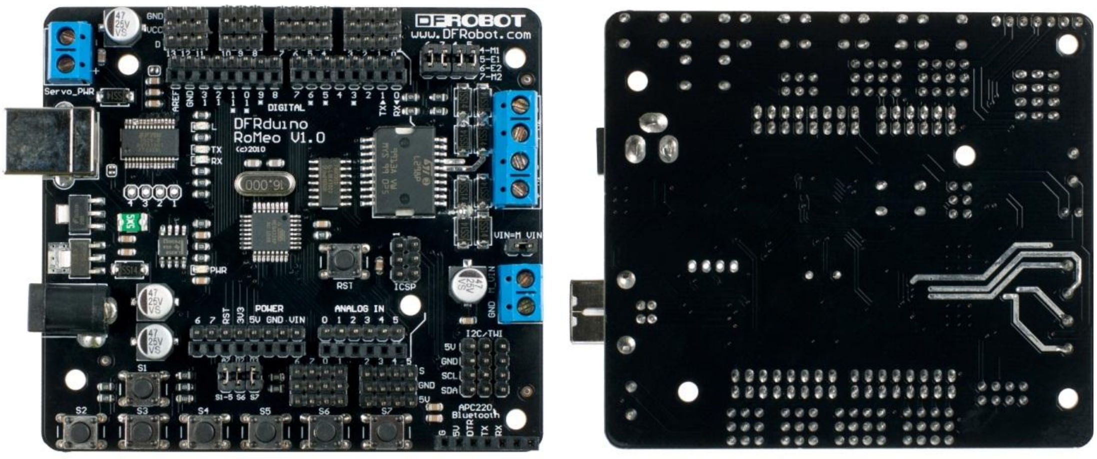
- 4WD Mobile platform Motors: 3-12V DC
- Speed: 90cm/s
- Dimensions: 200mm x 170mm x 105mm

Motor Specification

- • Gear Ratio 1:120
- • No-load speed(3V):100RPM
- • No-load speed(6V):200RPM
- • No-load current(3V):60mA
- • No-load current(6V):71mA
- • Stall current(3V):260mA
- • Stall current(6V):470mA
- • Torque (3V): 1.2Kg/cm
- • Torque (6V): 1.92Kg/cm
- • Size: 55mm x 48.3mm x 23mm
- • Weight:45g

Microcontroller

- Romeo-All in one Controller (Arduino Compatible Atmega 328)



- Atmega 328
- 14 Channels Digital I/O
- 6 PWM Channels (Pin11,Pin10,Pin9,Pin6,Pin5,Pin3)
- 8 Channels 10-bit Analog I/O
- USB interface

- Auto sensing/switching power input
- ICSP header for direct program download
- Serial Interface TTL Level
- Support AREF
- Support Male and Female Pin Header
- Integrated sockets for APC220 RF Module and DF-Bluetooth Module
- Six I2C Interface Pin Sets
- Two way Motor Driver with 2A maximum current
- 7 key inputs
- DC Supply：USB Powered or External 7V~12V DC
- DC Output：5V /3.3V DC and External Power Output
- Dimension：90x80mm
- Weight:60 gram

Parts List

Basic Kit

The following parts are included with the basic platform kit:

Product Code	Name	unit	Port	Port type
ROB0003	4WD Mobile Platform (Arduino Controller Supported)	1	/	/
DFR0004	Romeo-All in one Controller (Arduino Compatible Atmega 328)	1	14	digital
			8	Analog
SER0020	DF05BB Standard Servo (5kg)	1	1	digital
DFR0107	IR Kit For Arduino	1	1	digital
FIT0063	10 sets M3 * 10 hexagonal standoffs mounting kit	1	/	/

Upgrade Components

The following parts are included in the upgrade kit, along with all parts in the basic kit:

Product Code	Name	unit	Port	Port type
FIT0006	URM ultrasound mounting bracket	1	/	/

SEN0001	URM37 V3.2 Ultrasonic Sensor	1	2	Digital
SEN0019	Adjustable Infrared Sensor Switch	3	1 for each	Digital
DFR0106	Light Disc with 7 SMD RGB LED	2	3 for each	Digital (PWM)
TEL0026	DFRobot Bluetooth V3	1	1	Serial
SER0020	DF05BB Standard Servo (5kg)	1	1	Digital
FIT0004	Pan and Tilt Kit (Black Anodized) (no servos)	1	/	/

Additional Parts Required

In order to build the complete kit, you will need the following additional (basic) tools:
□ Wire cutter and wire stripper
□ Soldering Iron
□ Solder
□ Phillips Screw Driver
□ Pliers
□ 5xAA batteries (1.2V rechargeable or 1.5V alkaline)

Project Ideas

The basic robot is intended for you to create a small autonomous or remote controlled robot**.** Although the kit includes all essential parts to make an autonomous robot, it is intended to allow you to add your own electronics in order to satisfy your objectives.

Bluetooth Remote Control

We have created some sample code so that you can use your mobile phone with the Bluetooth to control the robot. (Sample code at the end of this document)

IR Remote Control

We have lots of IR controllers in the home for TVs, DVDs, Blu-rays, etc… And we have provided a sample code which you can modify to use your existing IR controls to control your robot.

Obstacle Avoidance

Giving a robot life is amazing. The great thing is that you can make your robot have its own personality. You can program it to react to obstacles in its own unique way. We provide some sample codes you can use to understand how the robot operates and observe its DFRobot personality.

Assembly

Although you can develop your own way to build your robot with the kit, this manual will give you a basic guide to assembling the robot.

Basic Platform

1. Before assembly the motor, you should solder the motor wires to the motor. Please leave a wire length of at least 15-20 cm in order to install to Romeo. Be careful when using a soldering iron, heat shrink tubing is recommended to protect soldered wires.(If you have condition you can use hot glue to fixed the wires) If you do not know how to use a soldering iron, please find professional help.

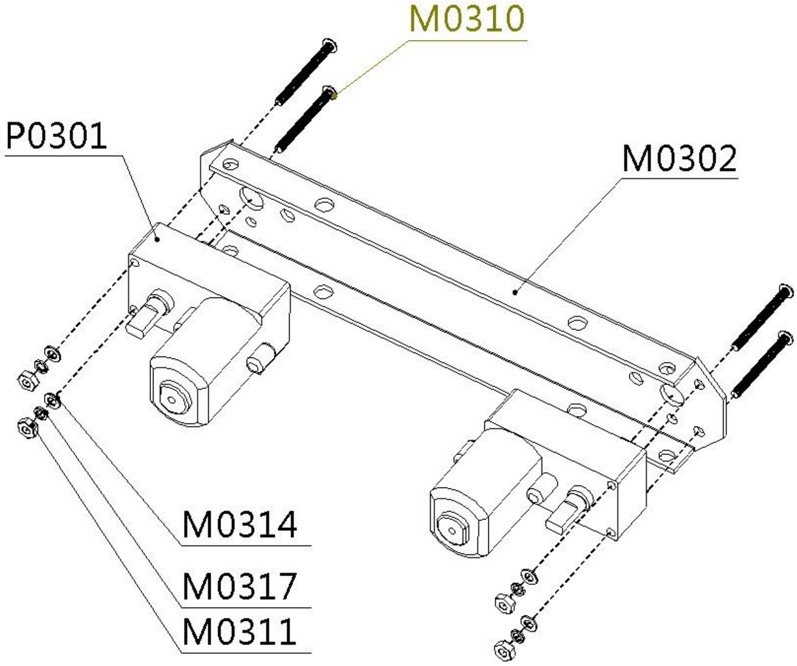


Figure 4-1

2. Figure 4-1; insert the two motors into the frame and using the given truss screws secure it. Note the direction of the motors, ensure the holes line up properly between the motor and truss. For details please see Figure 4-1 above.
3. All four motor are installed in the same way mentioned here.

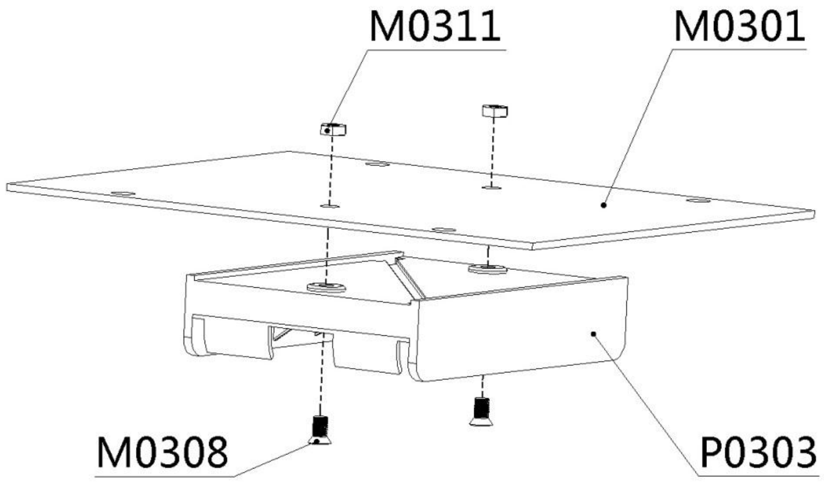


Figure 4-2

4. Figure 4-2; the battery holder is installed with countersunk screws on the fuselage’s lower plate.(tips: you may alternatively use the 7.4V Lipo 2200mAh Battery(FIT0137) at www.DFRobot.com to replace the battery holder).
5. The battery holder uses five AA batteries. You can use 1.2v rechargeable, or 1.5v alkaline batteries. We recommend using rechargeable batteries.

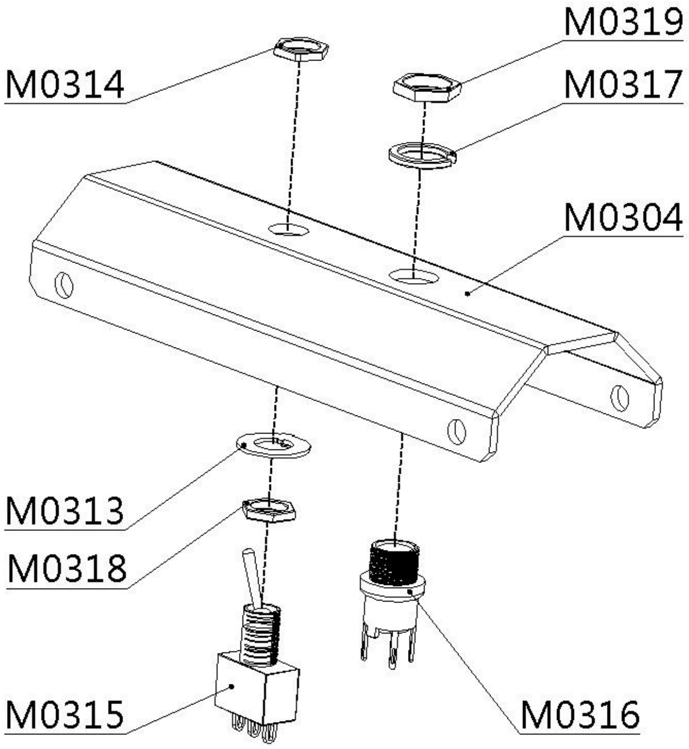


Figure 4-3

6. Figure 4-3; before assembly the power switch and power jack, you should solder the wires to the switch and jack. Please leave at least 10-15 cm of wire in order to install to battery holder and Romeo. Remember red for positive, black for negative.

Again, please be careful when using a soldering iron, heat shrink tubing is recommended to protect soldered parts. If you do not know how to use a soldering iron, please find professional help.

7. Please install the power switch and power jack on the bezel as pictured above in figure 4-3.

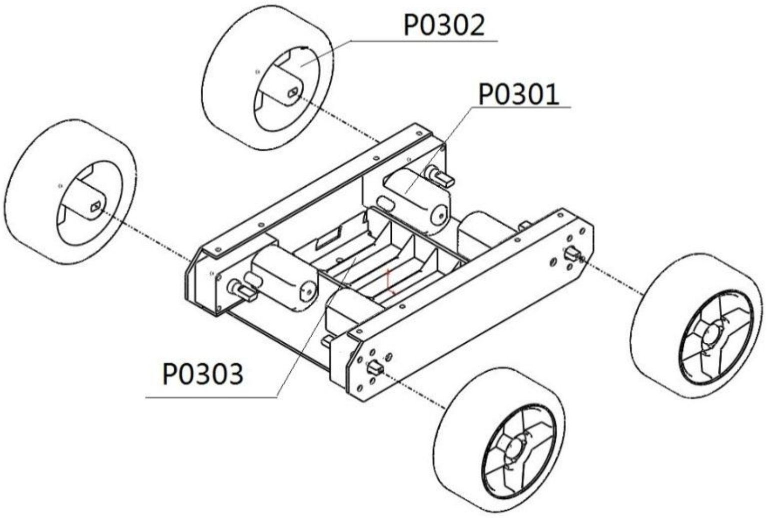


Figure 4-4

8. Figure 4-4; use the screws to connect the fuselage's lower plate and motor brackets.
9. Before installing any other parts, you must attach the wheels to the motors. Otherwise you may risk damage to the motor shafts.

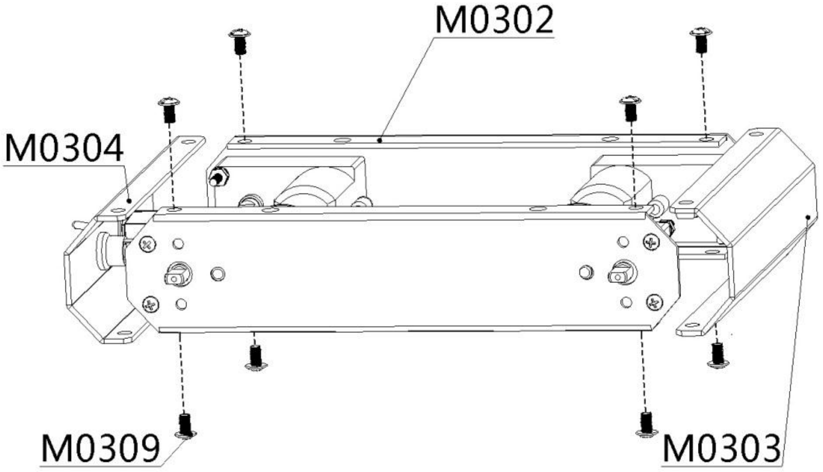


Figure 4-5

10. Figure 4-5; using the screw to fix the side truss to the fuselage's lower plate.

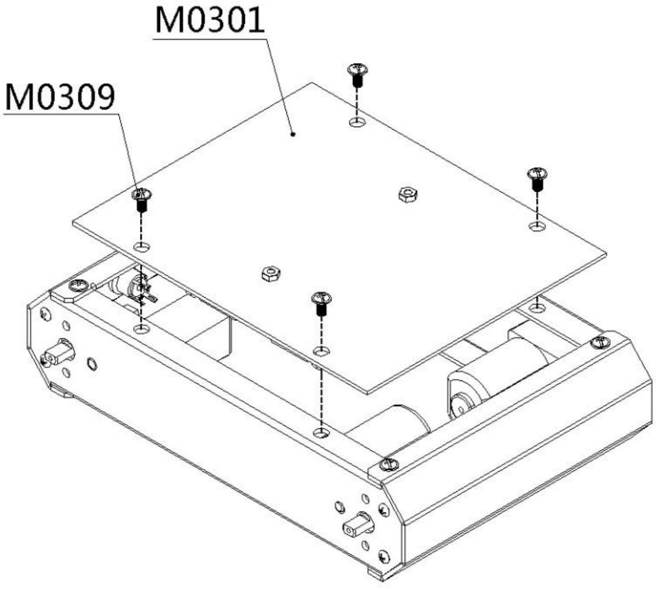


Figure 4-6

11. Figure 4-6; use flange screws to fix the fuselage's upper plate to the side truss.
12. When installing the fuselage upper plate, make sure the battery cables have been identified and pull the leads out so you can wire them to the Romeo.

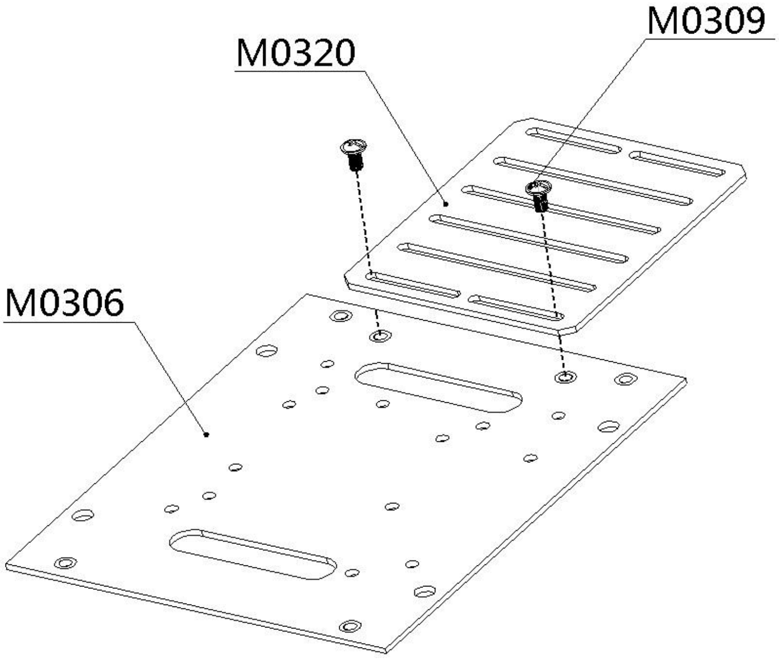


Figure 4-7

13. Figure 4-7; use flange screws to fix the sensor mounting grill to the bracket. This grill is an option to use if you have any other sensors to install.

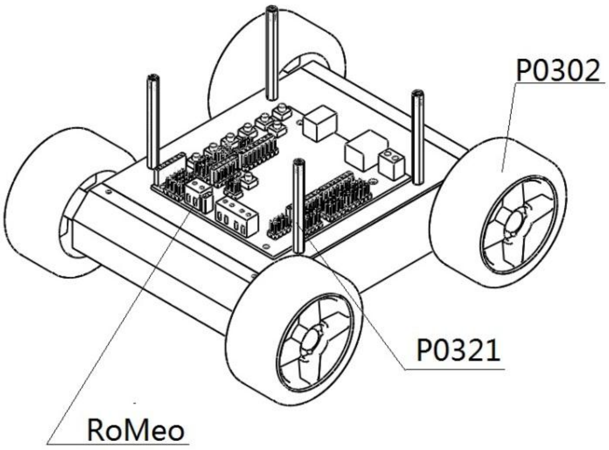


Figure 4-8

14. Figure 4-8; the copper stand-offs need to be mounted on the fuselage's upper plate in order to support the top plate.

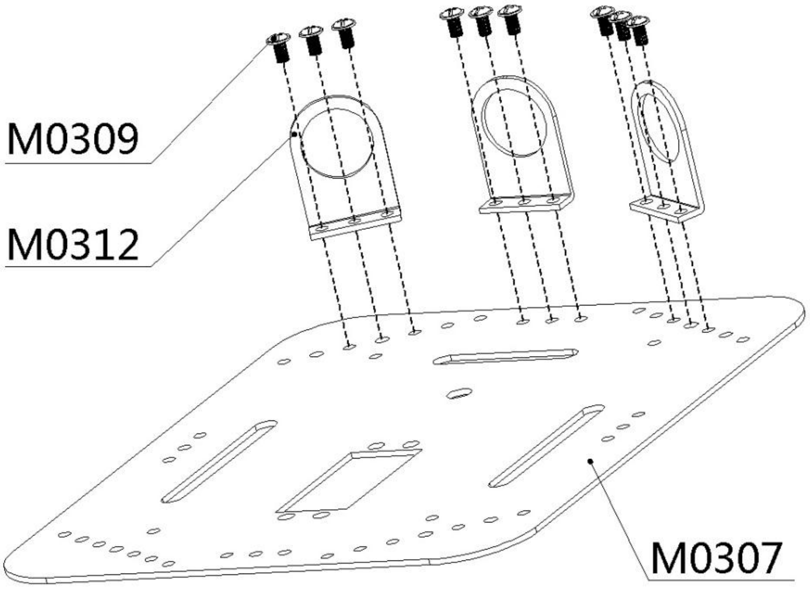


Figure 4-9

15. Figure 4-9; use flange screws to fix the IR sensor's mounting bracket on the top plate.

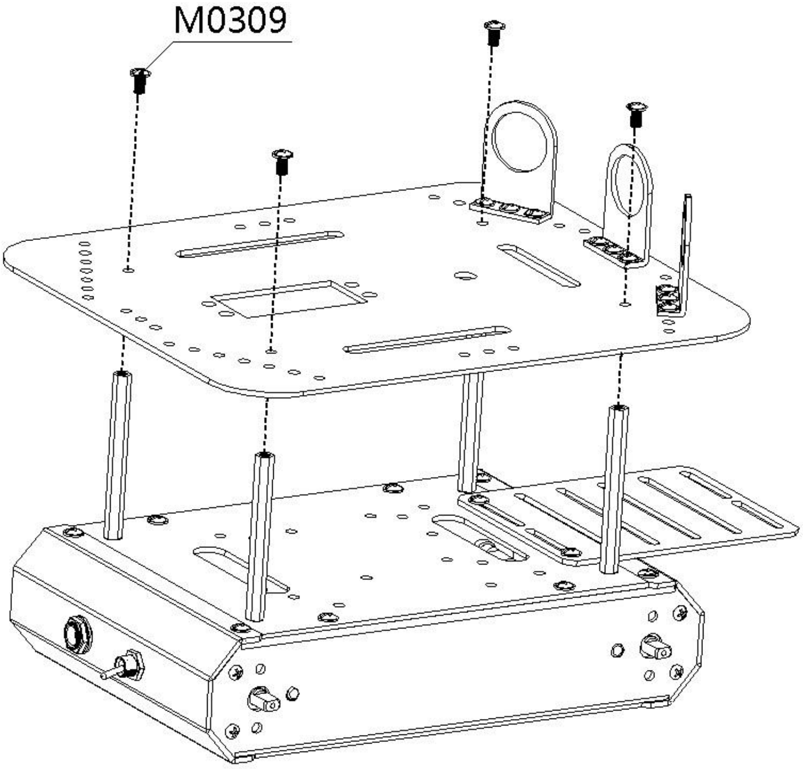


Figure 4-10

16. Figure 4-10; use flange screws to fix the top plate on to the Copper stand-offs.

Electronics and Wiring

It is recommended that you assemble and test each sub-section before assembling the entire robot. This will help you troubleshoot any problems as they arise.

Power and switch

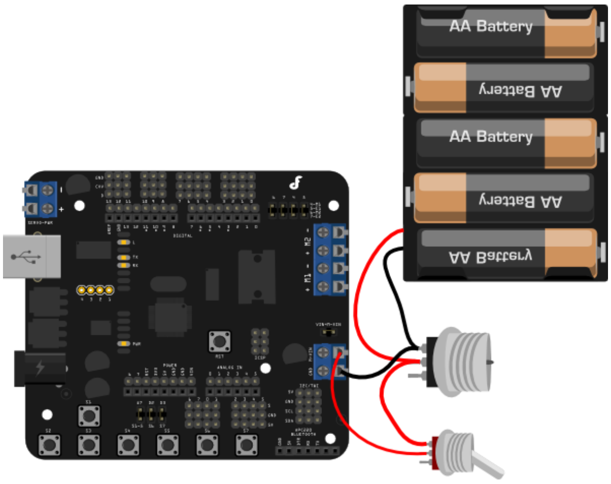


Figure 4-11

Please refer to Figure 4-11 to connect the power supply and switch. Remember that red for positive, and black for negative.

Motors

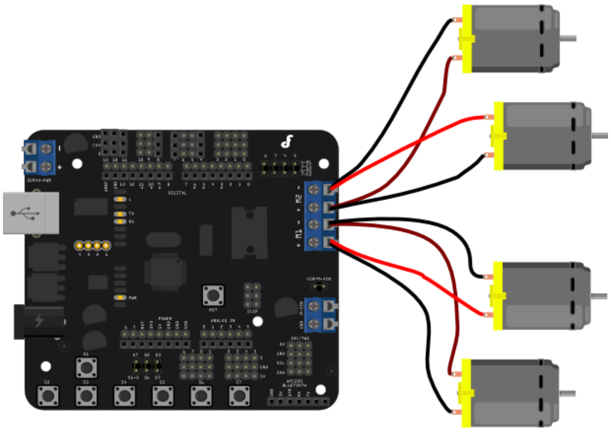


Figure 4-12

Please refer to Figure 4-12 to connect the two left motor to M1 and the two right motor to M2. When you connect the motors you should test the motors for each side to ensure they move in the right direction. If not, you should flip the wiring for only one of the motors.

IR remote control

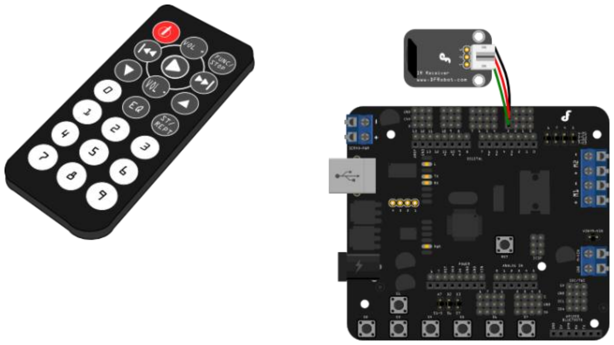


Figure 4-13

Please refer to Figure 4-13 to connect the IR receiver to digital pin 3, the green line is for data, the black line is for GND, and the red line is for Vcc.

IR sensors

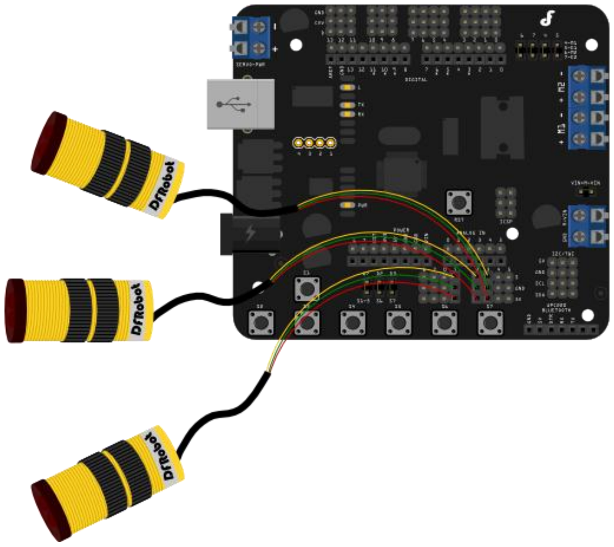


Figure 4-14

Please refer to Figure 4-14 to connect the IR sensors to analog pins 1, 2, and 3. You have to re-wire the connectors to fit the analog pins (Notice the wire order). If you don’t want to use the analog pins, you can just plug into digital pins but you should note that in this setup most of the digital pins are used to drive other sensors and the motors.

Servo

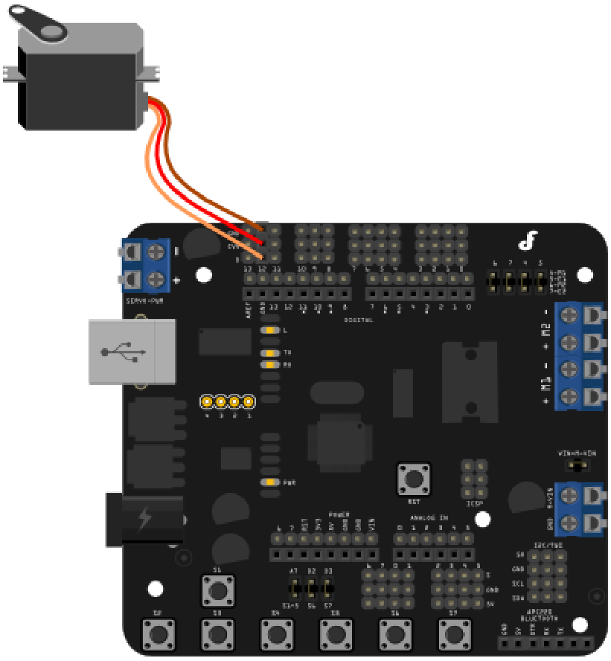


Figure 4-15

Please refer to Figure 4-15 to connect the servo to digital pin 12, the orange line is for the data, the brown line is for GND, the red line is for Vcc.

URM37V3.2 Ultrasonic sensor

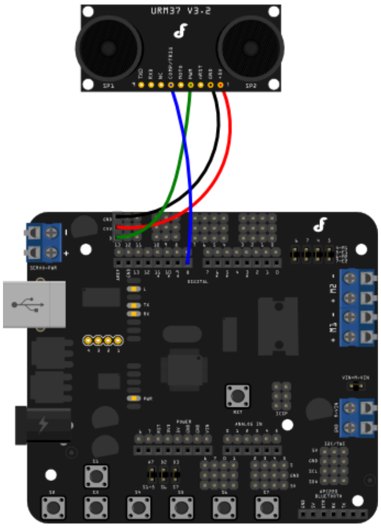


Figure 4-16

Please refer to Figure 4-16 to connect the URM sensor to digital pins 8 (blue), and 13(green), the red line is for 5V+, the black line is for GND.

7LEDs disc

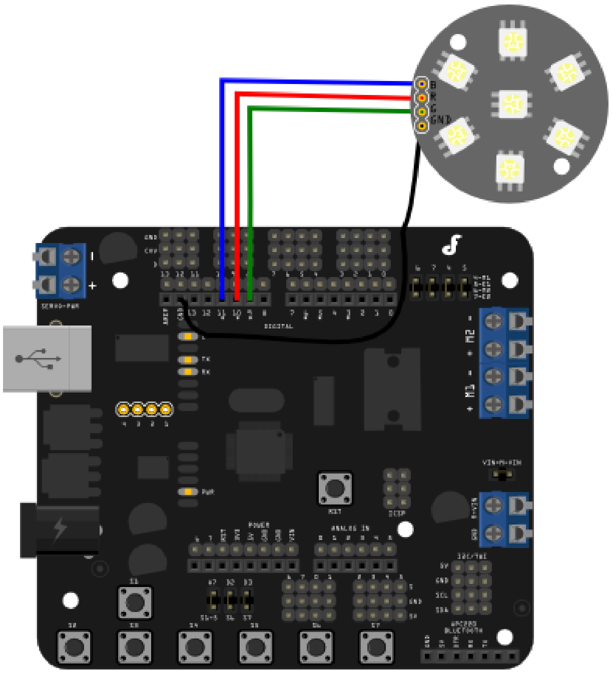


Figure 4-17

Please refer to Figure 4-17 to connect the 7LEDs disc to digital pins 9(green), 10(red), and 11(blue), and GND (black) to GND.

Totals

When all of the sub-section tests have passed, you should end up with something that looks just like the diagram below.

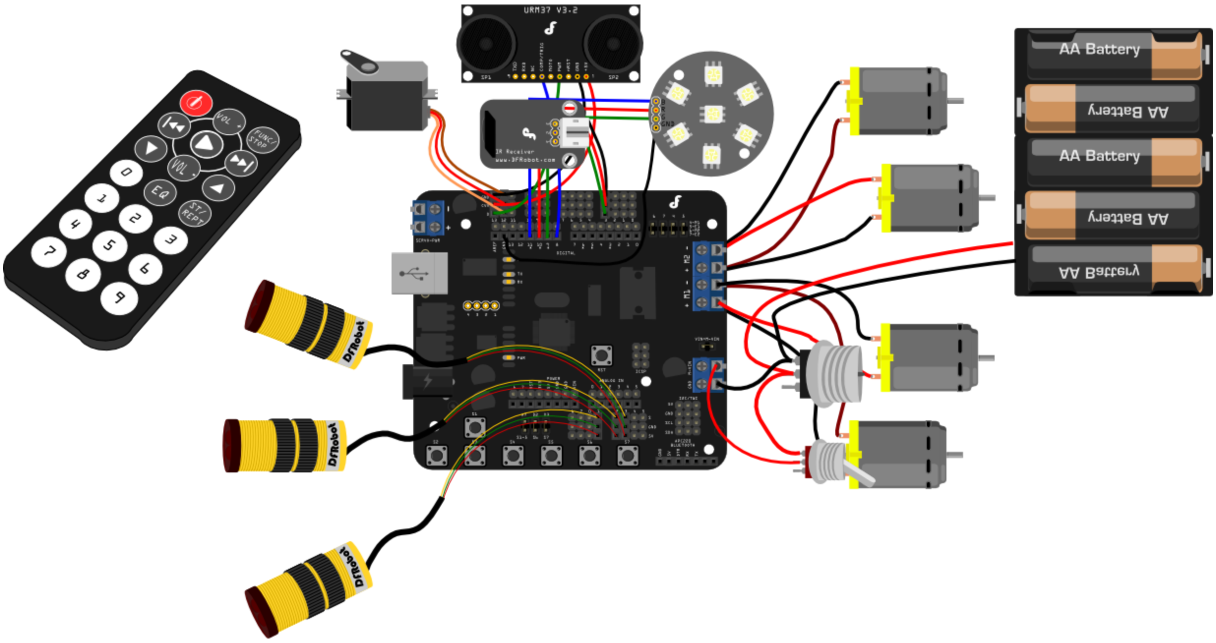


Figure 4-18

Support

Tech support: techsupport@dfrobot.com
Customer Service: sales@dfrobot.com

Arduino Code

The following demo code is compiled in the Arduino IDE (www.arduino.cc) and uses two subroutines. You are free to copy/paste the code into the Arduino compiler. Ensure the motor controller is connected properly before use.

Bluetooth Remote Control

```
//This code is just for Cellbots in android market.
#include <Metro.h>//version 0.1

#define LF 0
#define RT 1

#define FORW 1 // forward
#define BACK 0 // back
Metro MotorGap = Metro(30,true);
Metro printGap = Metro(100,true);

#define EN1 6 // right motor enable pins
#define IN1 7 // right motor direction of the pin
#define EN2 5 // left motor enable pins
#define IN2 4 // left motor direction of the pin


int lr = 0;
int rr = 0;
int left = 0;
int right = 0;

void setup(){
  int i;
  for(i = 4;i <= 7;i++)  pinMode(i, OUTPUT);

  Serial.begin(9600);      //Set Baud Rate
```

```
}


void loop(){

  AttachCmd();
  if(MotorGap.check()){
    if(left != 0 || right != 0)
      Motor_Control(rr,map(constrain(abs(right),10,45), 10,
45,150,255),lr,map(constrain(abs(left),10,45), 10, 45,150,255));
    else Motor_Control(0,0,0,0);
  }

}


int BluetoothCmd[15];
int BlankPos[5];
int BlankCount = 0;
int Endbyte = 0x0A;

void AttachCmd(){

  if(Serial.available()){
    unsigned long timer = millis();
    int i = 0;
    int RetryCounter = 0;
    int num = 0;
    boolean valid = true;

    for(int j = 0; j<10; j++)
      BluetoothCmd[j] = 0;

    byte firstByte = Serial.read();
    //Serial.println(input,BYTE);
    if(firstByte >= ’a’ && firstByte <= ’z’)
    {
      //Serial.println(”PassHeader”);
      switch(firstByte){
      case ’w’:
        BlankCount = 0;
        for(i = 0;i < 5; i++)  BlankPos[i] = 0;
        break;
      default:
        break;
      }
    }

    BluetoothCmd[0] = firstByte;
    i = 1;

    while(true){

      if(Serial.available()){

        BluetoothCmd[i] = Serial.read();

        //Serial.print(BluetoothCmd[i],HEX);
        //Serial.print(″\t″);

        if(BluetoothCmd[i] == 0x20){
          BlankPos[BlankCount] = i;
          BlankCount ++;
        }


        if(BluetoothCmd[i] == Endbyte){
          BlankPos[BlankCount] = i;
          break;
        }
        else if(i >= 14){
          Serial.flush();
          valid = false;
          break;
        }
        i++;
      }
      else     delayMicroseconds(100);
    }



    //Serial.println(″″);

    if(valid){//voice control
      if(BluetoothCmd[0] == ’s’ && BluetoothCmd[1] == 0x0A){

        lr = 0;
        rr = 0;
        left = 0;
        right = 0;

      }

      if(BluetoothCmd[0] == ’f’ && BluetoothCmd[1] == 0x0A){
        lr = 0;
        rr = 0;
        left = 240;
        right = 240;

      }

      if(BluetoothCmd[0] == ’b’ && BluetoothCmd[1] == 0x0A){

        lr = 1;
        rr = 1;
        left = 240;
        right = 240;
      }

      if(BluetoothCmd[0] == ’l’ && BluetoothCmd[1] == 0x0A){
        lr = 0;
        rr = 1;
        left = 240;
        right = 240;

      }
      if(BluetoothCmd[0] == ’r’ && BluetoothCmd[1] == 0x0A){
        lr = 1;
        rr = 0;
        left = 240;
        right = 240;

      }
      //voice control end

      if(BluetoothCmd[0] == ’w’ && BluetoothCmd[i] == 0x0A){
        char cmd[BlankPos[1] - BlankPos[0]];

        for(int j = 0; j < (BlankPos[1] - BlankPos[0]) ; j++){
          cmd[j] = BluetoothCmd[j + 1 + BlankPos[0]];
        }
        char Scmd[BlankPos[2] - BlankPos[1]];

        for(int j = 0; j < (BlankPos[2] - BlankPos[1]) ; j++){
          Scmd[j] = BluetoothCmd[j + 1 + BlankPos[1]];
        }



        Serial.print("cmd:\t");
        Serial.print(atoi(cmd));
        Serial.print("\t");
        Serial.println(atoi(Scmd));
        /*
        */
        left = atoi(cmd);
        right = atoi(Scmd);
        if(left < 0)  lr = 1;
        else lr = 0;

        if(right < 0)  rr = 1;
        else rr = 0;
      }
    }
    else  Serial.println("Invalid");
  }
  else  Serial.flush();

}



void Motor_Control(int M1_DIR, int M1_EN, int M2_DIR, int M2_EN) //
control the motor rotation
{
  ////////// M1 ////////////////////////
  if (M1_DIR == FORW) // M1 direction of the motor
    digitalWrite (IN1, FORW); // set high, set the direction of the
forward
  else
    digitalWrite (IN1, BACK); // set low, set the direction of the back
  if (M1_EN == 0) // M1 motor speed
```

```
    analogWrite (EN1, LOW); // set low, miniQ stop
  else
    analogWrite (EN1, M1_EN); // Otherwise, set the corresponding value

  /////////// M2 //////////////////////
  if (M2_DIR == FORW) // M2 motor direction
    digitalWrite (IN2, FORW); // set high, the direction of forward
  else
    digitalWrite (IN2, BACK); // set low, the direction of backward
  if (M2_EN == 0) // M2 motor speed
    analogWrite (EN2, LOW); // set low, to stop
  else
    analogWrite (EN2, M2_EN); // set the value for a given
}
```

IR Remote Control

```
/*

 4WD kit

 author:Lauren
 version:0.1
 date:2011.12.6

 Function:
 IR remote control 4wd robot



 */



int E1 = 5;      //M1 Speed Control
int E2 = 6;      //M2 Speed Control
int M1 = 4;     //M1 Direction Control
int M2 = 7;     //M1 Direction Control





#include <Metro.h>
```

```
/*
 * IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 */

#include <IRremote.h>

int RECV_PIN = 3;

Metro output = Metro(30,true);

IRrecv irrecv(RECV_PIN);

decode_results results;
void stop(void)                    //Stop
{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
void advance(char a,char b)          //Move forward
{
  analogWrite (E1,a);      //PWM Speed Control
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void back_off (char a,char b)          //Move backward
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             //Turn Left
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
```

```
void turn_R (char a,char b)             //Turn Right
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

void setup()
{
  unsigned char i;
  for (i = 4; i <= 7; i ++) // settings control two of the four pins for
the output motor
      pinMode (i, OUTPUT);
  Serial.begin(57600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }

  if(output.check()){
    switch(results.value){
    case 0xFD807F:
      Serial.println(″forward″);
      advance (240,240);


      break;

    case 0xFDA05F:
      Serial.println(″back″);
      back_off (240,240);

      break;

    case 0xFD906F:
      Serial.println(″stop″);
      stop();

      break;
    case 0xFD20DF:
      Serial.println(″left″);
      turn_L (240,200);

      break;


    case 0xFD609F:
      Serial.println(″right″);
      turn_R (240,200);

      break;


    default:
      break;
    }
  }
}
```

Obstacle Avoidance

```
int E1 = 5;     //M1 Speed Control
int E2 = 6;     //M2 Speed Control
int M1 = 4;    //M1 Direction Control
int M2 = 7;    //M1 Direction Control

void stop(void)                    //Stop
{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
void advance(char a,char b)          //Move forward
{
  analogWrite (E1,a);      //PWM Speed Control
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
```

```
}
void back_off (char a,char b)          //Move backward
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             //Turn Left
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void turn_R (char a,char b)             //Turn Right
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}


void setup()
{
  Serial.begin(9600);
  pinMode( A2 , INPUT);
  pinMode( A3 , INPUT);
  pinMode( A1 , INPUT);
  int i;
  for(i=6;i<=9;i++)
    pinMode(i, OUTPUT);

}

void loop()
{
  if (( !(  digitalRead( A2) ) || ( ( !(   digitalRead( A3) ) &&
( !(     digitalRead(A2)  )  && !(   digitalRead(  A1)  )  )  )  ||
( !( digitalRead( A3) ) && !( digitalRead( A1) ) ) ) ))//read sensor
from IR distence switch
  {
    back_off(240,240);
    delay(1000);
    Serial.print( ″back″ );

    Serial.println(″″);
    if (random( 1024 )>512)
    {
      turn_R(240,240);
      delay(1000);
      Serial.print( ″random right″ );
      Serial.println(″″);
    }
    else
    {
      turn_L(240,240);
      delay(1000);
      Serial.print( ″random left″ );
      Serial.println(″″);
    }


  }
  else
  {
    if (( !(  digitalRead( A3) ) || ( !(  digitalRead(  A2) )
&& !( digitalRead( A3) ) ) ))//read sensor from IR distence switch
    {
      back_off(240,240);
      delay(1000);
      turn_L(240,240);
      delay(1000);
      Serial.print( ″back and  left″ );
      Serial.println(″″);
    }
    else
    {
      if (( !(  digitalRead( A1) ) || ( !(  digitalRead( A1) )
&& !( digitalRead( A2) ) ) ))//read sensor from IR distence switch
      {
        back_off(240,240);
        delay(1000);
        turn_R(240,240);
        delay(1000);
        Serial.print( ″back and right″ );
        Serial.println(″″);
      }
      else
      {
        advance(240,240);
        delay(1000);

        Serial.print( ″go″ );
        Serial.println(″″);
      }
    }
  }
}
```

Standard PWM DC control

```
int E1 = 5;     //M1 Speed Control
int E2 = 6;     //M2 Speed Control
int M1 = 4;    //M1 Direction Control
int M2 = 7;    //M1 Direction Control

///For previous Romeo, please use these pins.
//int E1 = 6;     //M1 Speed Control
//int E2 = 9;     //M2 Speed Control
//int M1 = 7;    //M1 Direction Control
//int M2 = 8;    //M1 Direction Control


void stop(void)                    //Stop
        {
          digitalWrite(E1,LOW);
          digitalWrite(E2,LOW);
        }
void advance(char a,char b)          //Move forward
        {
          analogWrite (E1,a);      //PWM Speed Control
          digitalWrite(M1,HIGH);
          analogWrite (E2,b);
          digitalWrite(M2,HIGH);
        }
void back_off (char a,char b)          //Move backward
        {
          analogWrite (E1,a);
          digitalWrite(M1,LOW);
          analogWrite (E2,b);
          digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             //Turn Left
        {
          analogWrite (E1,a);
          digitalWrite(M1,LOW);
          analogWrite (E2,b);
          digitalWrite(M2,HIGH);
        }
void turn_R (char a,char b)             //Turn Right
        {
          analogWrite (E1,a);
          digitalWrite(M1,HIGH);
          analogWrite (E2,b);
          digitalWrite(M2,LOW);
        }
void setup(void)
{
    int i;
    for(i=6;i<=9;i++)
    pinMode(i, OUTPUT);
    Serial.begin(19200);      //Set Baud Rate
}
void loop(void)
{
     char val = Serial.read();
     if(val!=-1)
       {
          switch(val)
           {
             case ’w’://Move Forward
                     advance (100,100);   //PWM Speed Control
                     break;
             case ’s’://Move Backward
                     back_off (100,100);
                     break;
             case ’a’://Turn Left
                     turn_L (100,100);
                     break;
             case ’d’://Turn Right
                     turn_R (100,100);
                     break;
            }
          delay(40);
       }
      else stop();
}
```

Release Date	Version	Comments
Dec 7	1.0	First Release