

What You'll Learn

In this project, you'll create an Insect Robot (as shown in Figure-1 below) that walks forward on four legs. Actually it's not really an insect because it only has four legs, insects does have six of them, right? However, the robot got the name because of his shape with the thin wire legs and the Infrared sensor.

We will briefly learn micro servo, Funduino Nano and infrared sensor and build a body for the insect by binding two servos together and shaping legs for it from a wire clothes hanger. Arduino will turn the two servos into one at a time, which moves each pair of metal legs like real legs so the insect can crawl forward. We will also strap battery on the body so our insect can behave autonomously.

Once you've learned the techniques in this chapter, you can easily extend your Insect Robot with new tentacles and sensors and new code. This easy-to-assemble kit is designed to enable creative engineering and robotics learning for age 6 and above. It is the best way to start building robot on Arduino.



Figure-1

List of Components and Tools

- ◆ Funduino nano V3.0 Atmel ATmega328 Mini-USB Board*1
- ◆ Micro Servo*2
- ◆ SHARP GP2Y0A41SK0F IR Sensor*1
- ◆ 3.7V/ 600mAh LiPo Battery*1
- ◆ Steel Wire 200mm x 1mm*1
- ◆ ABS Sheet (50x50mm)*1
- ◆ Double Sided Foam Tape(L/W/H:40x30x3mm)*1
- ◆ Cable Tie 1.8x100mm*5
- ◆ Cable Tie 4x200mm*2
- ◆ sharp-nose pliers*1
- ◆ 2 mm cross screwdriver*1
- ◆ Scissors*1

For the programming, a Micro USB data line and a computer are needed. When the assembly is done, you need to use the Arduino IDE to burn in the code.

Components Introduction

Funduino Nano

The Funduino Nano is an ultra-small simple I/O platform based on open source code, it has a big advantage in size compared to the old USB version. Funduino could be used to develop works which need to operate independently and interactive effective of electronic appliances, also can develop works connected with the PC, in collaboration with Flash, Processing, the Max/Msp, the PD and other software.

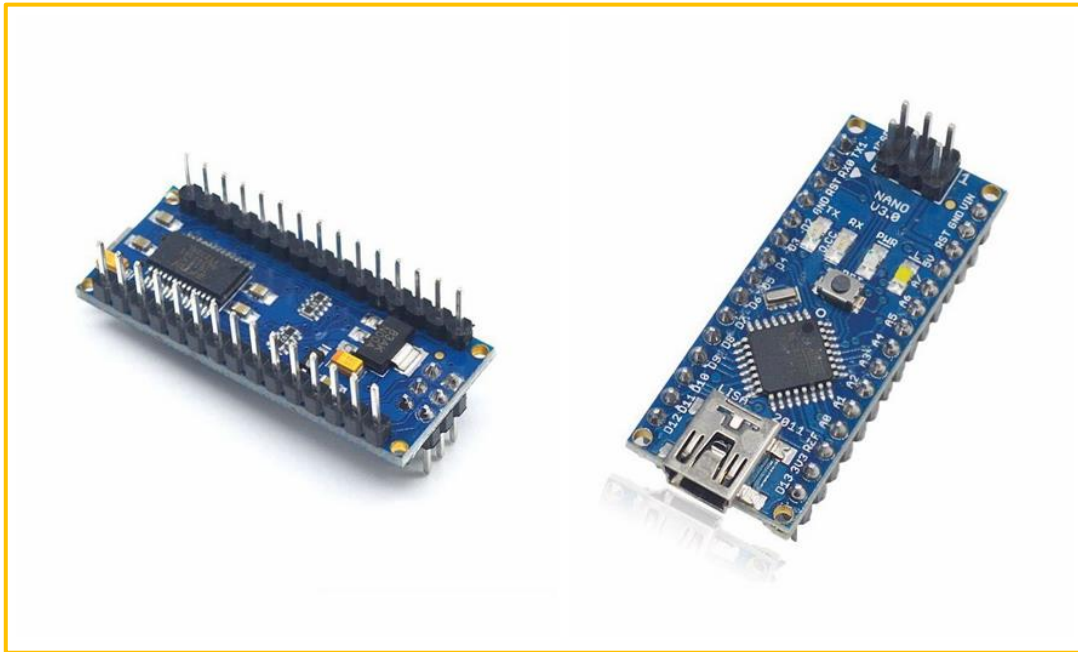


Figure-2

Servo Motors

Servo motors come in different sizes and prices and are based on different technologies. In this context, we are talking about micro servos, the kind used in R/C toys, for example. Servo motors have a servo controller that directs the position of the motor wherever we want. The red wire is power, the gray is GND and the orange is the data wire.

The motor itself is a DC (direct current) motor with gears. Servo motors usually rotate rather slowly and with a relatively strong torque. You can buy micro servo motors with either limited rotation or continuous rotation. Limited rotation models work for most purposes, and you can control their movement quite precisely by degrees of rotation. In continuous rotation servos, you can control only speed and direction.



Figure-3

SHARP GP2Y0A41SK0F

GP2Y0A41SK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IR-LED (infrared emitting diode) and signal processing circuit. The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method. This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

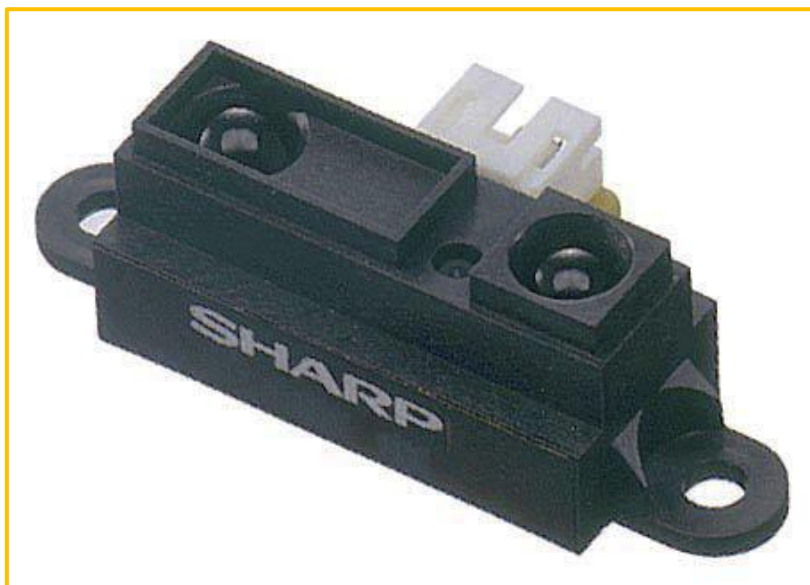


Figure-4

Assembling the Frame

Now it's time to assemble the frame for the robot. The frame of the walker consists of two connected servo motors.

Figure-5. Gluing the two servo motor together with double sided foam tape.

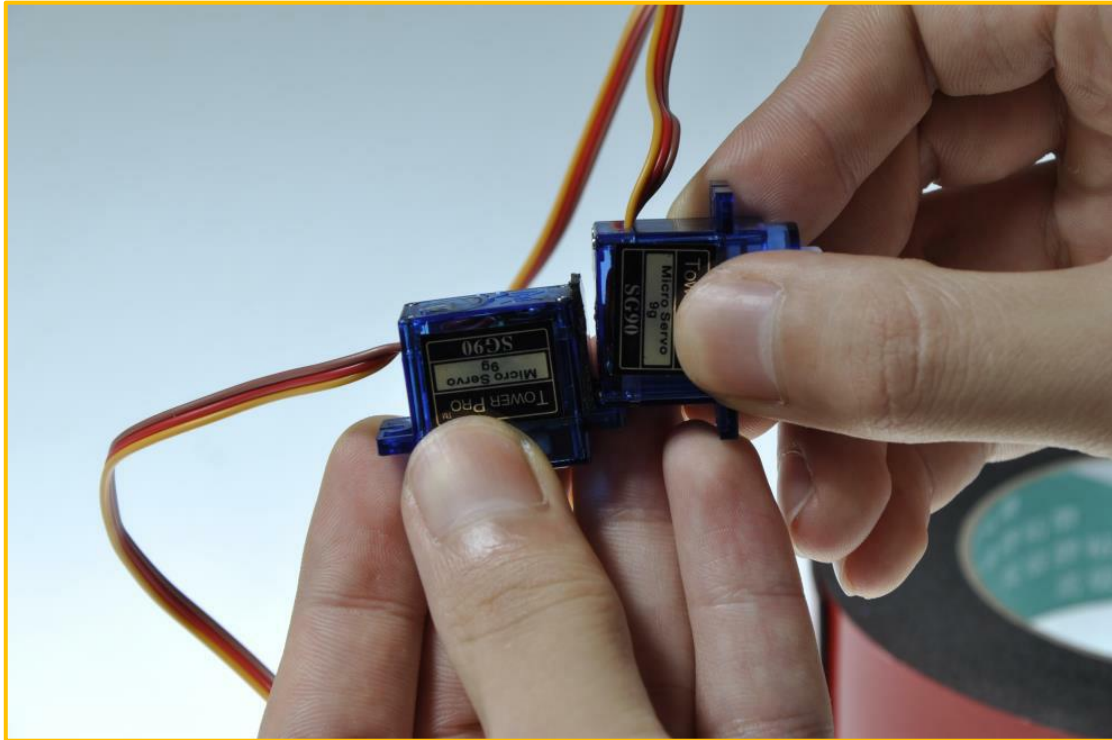


Figure-5

Figure-6. Tying the servo motor tightly with cable tie and cut out the excess.

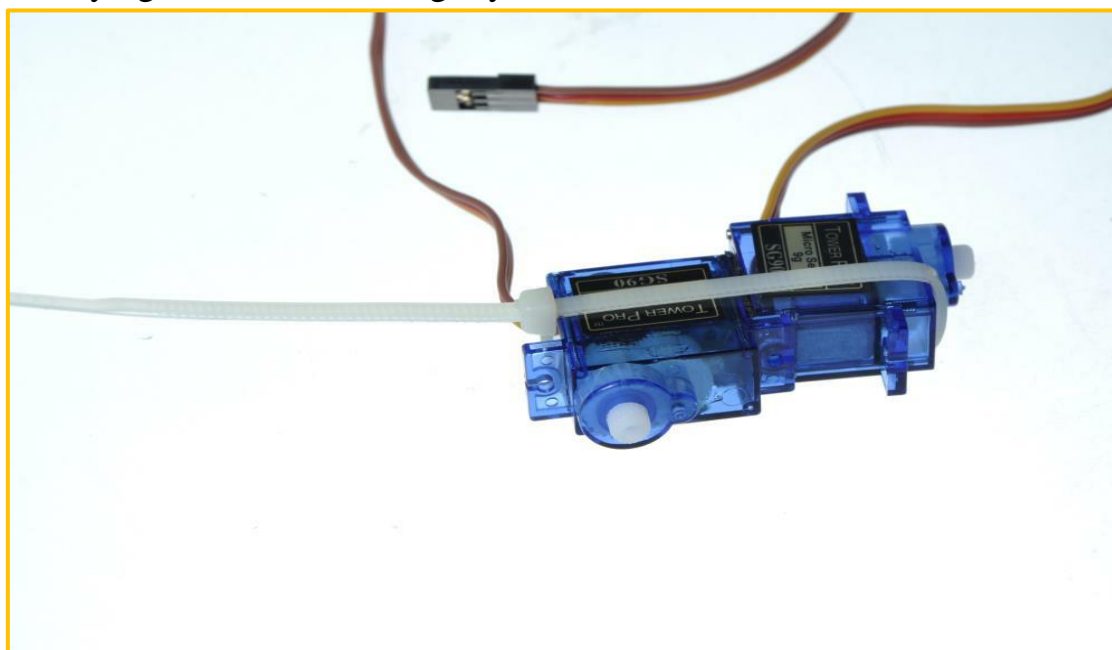


Figure-6

Bending the wire pair into the V type as shown in Figure-7, then bending 1cm out of the V end to 90 degrees, as shown in Figure-8.

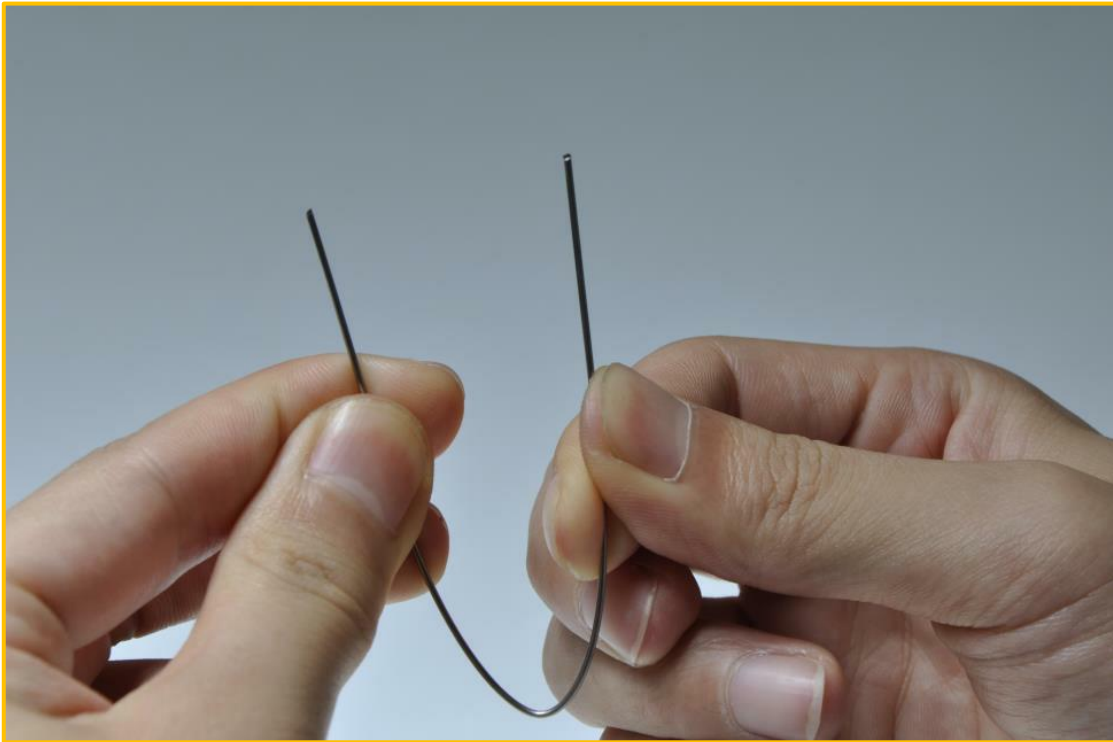


Figure-7



Figure-7



Figure-8

Figure-9. Passing the two feet of the wire through the second holes in the wheel.

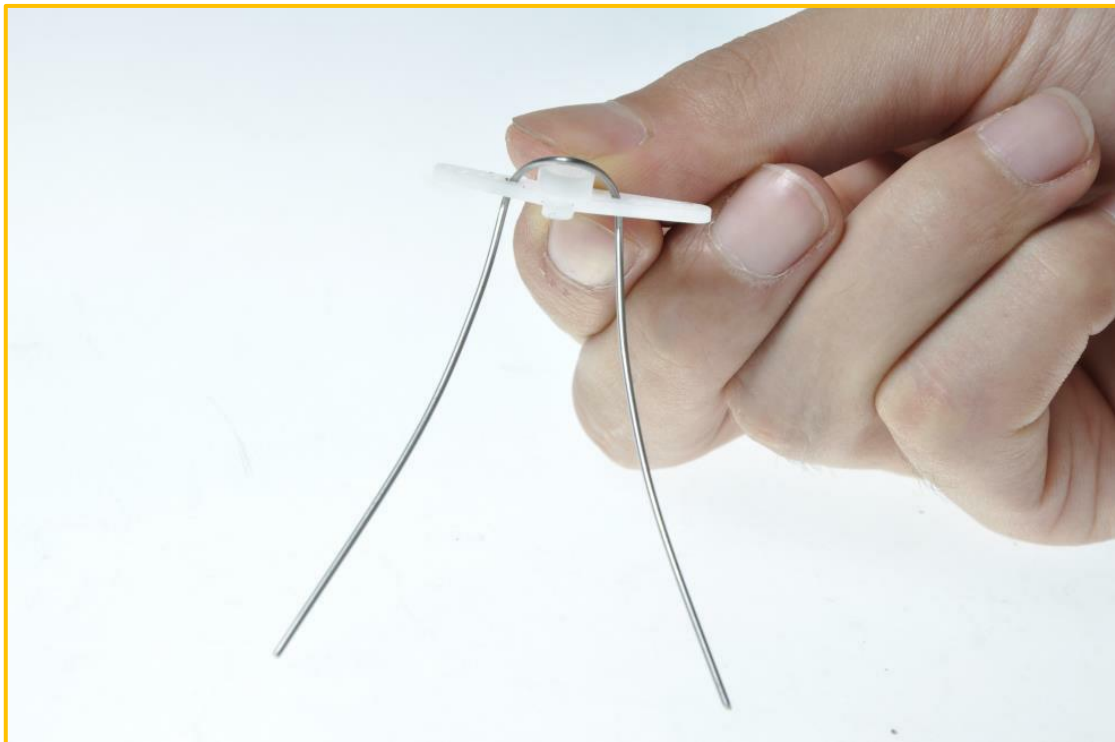


Figure-9

Figure-10. Folding the feet of the wire to the other end of the sharp corner with a pair of pliers. Note: Sharp-nose pliers is recommended, otherwise the white support will break easily.

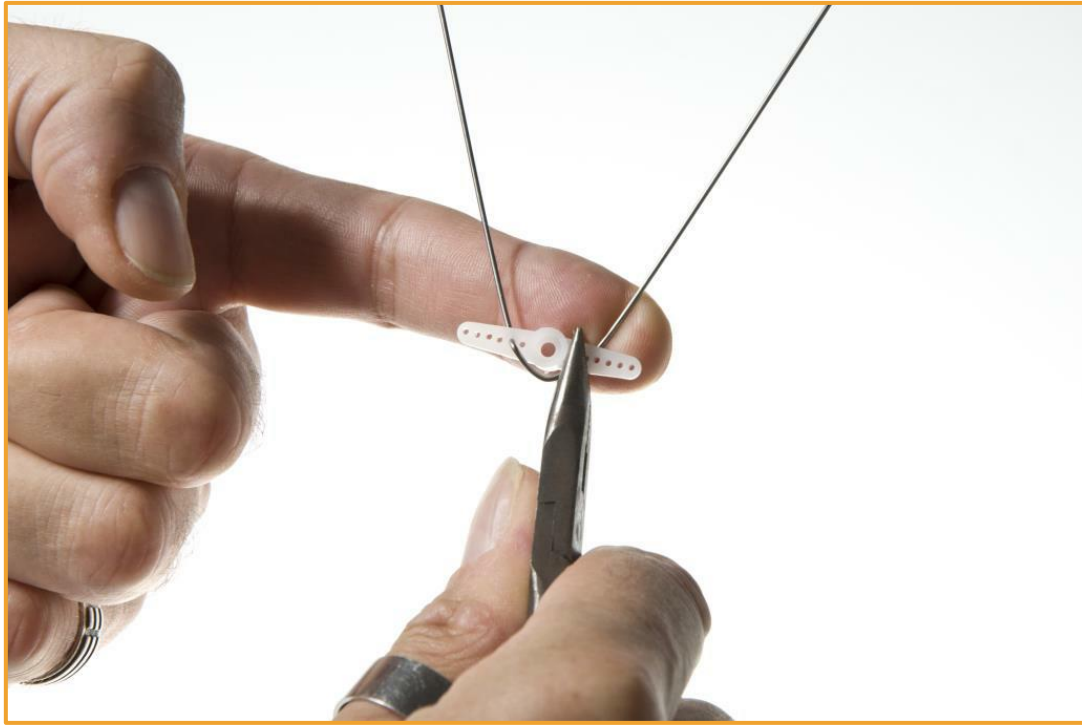


Figure-10

Through the above steps, you have completed the general look of the mechanical legs, Finally, the front and back legs bent, we can refer to the size shown in the figure on the wire bending, The front leg is completed as shown in Figure-11, and hind legs as shown in Figure -12.

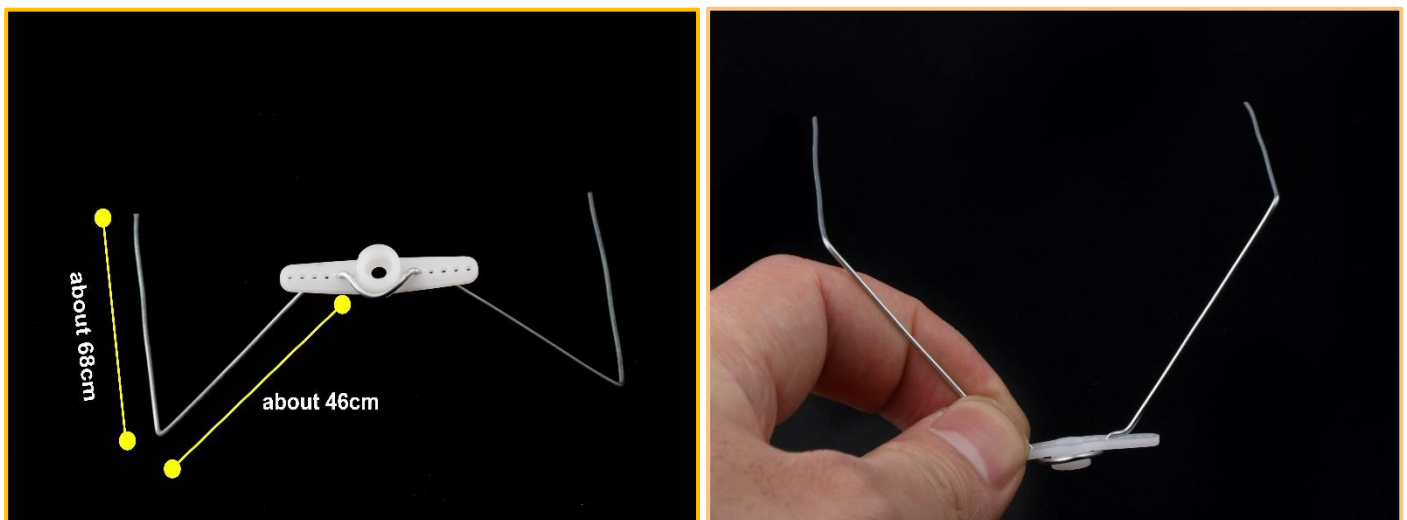


Figure-11

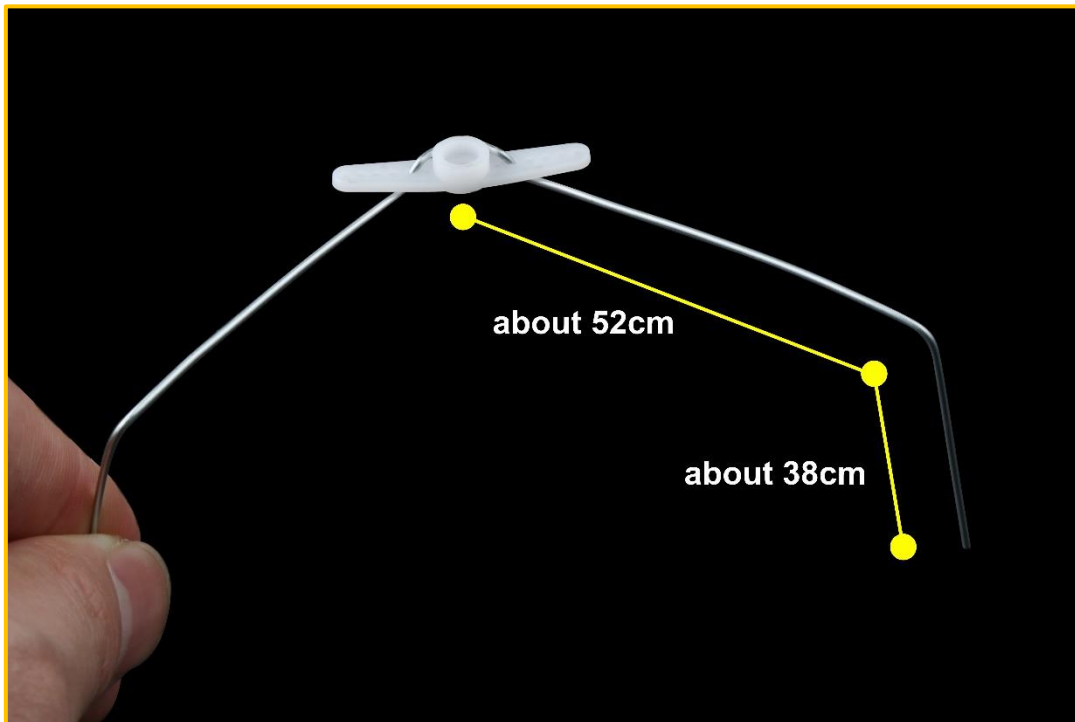


Figure-12

Figure-13. Put "shoes" on the four legs of the robot and insert the wire into the slip-resistant particles (to prevent slipping during walking).

Note: The insertion process will be more difficult, you can first use a sharp object to slip a small hole in the non-slip particles (be careful not to tie the hand), so the wire will be easy to insert non-slip particles.

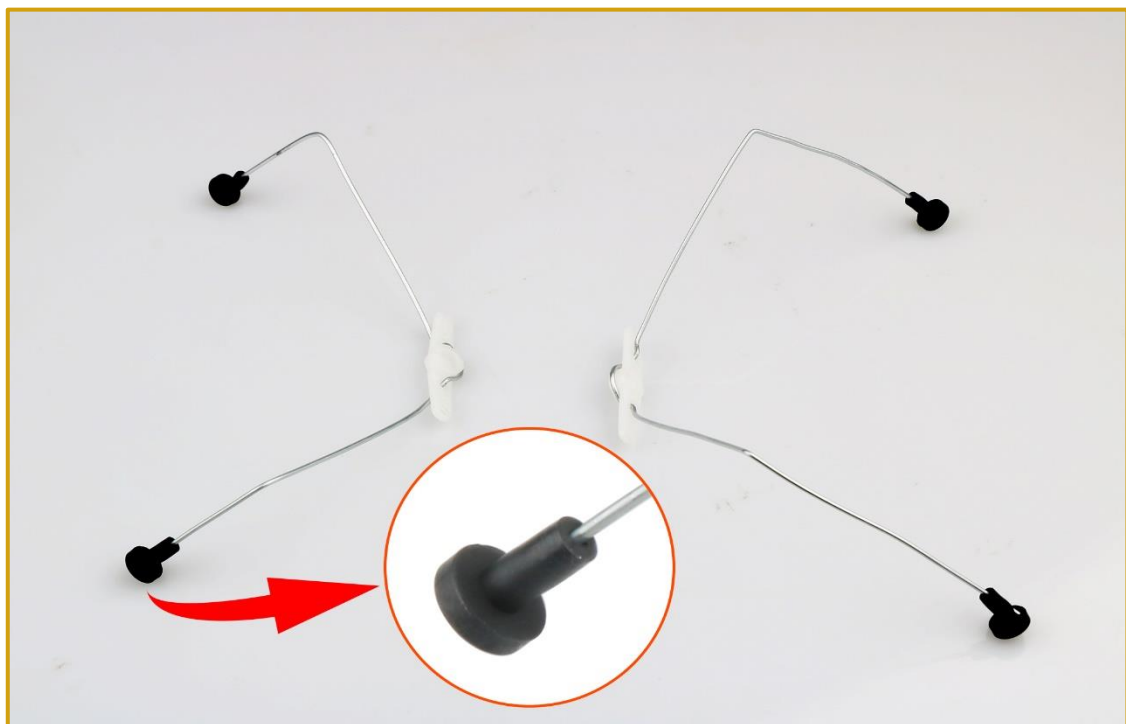


Figure-13

If you complete the above steps properly, a robot should stand up as shown in Figure-14 and Figure-15.

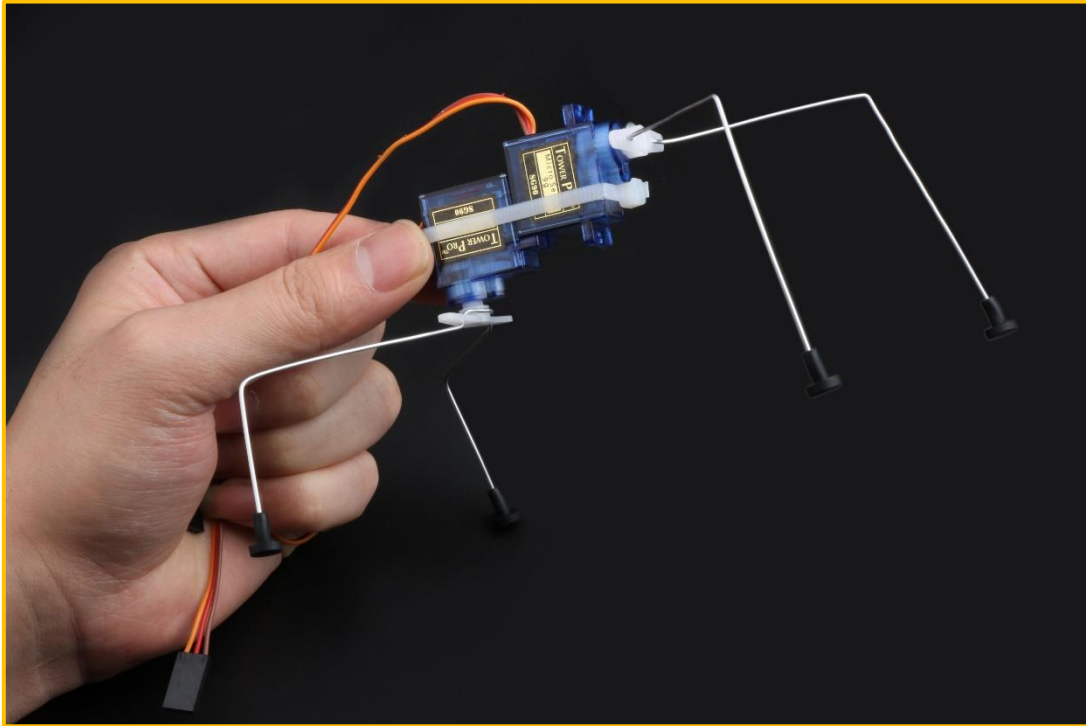


Figure-14

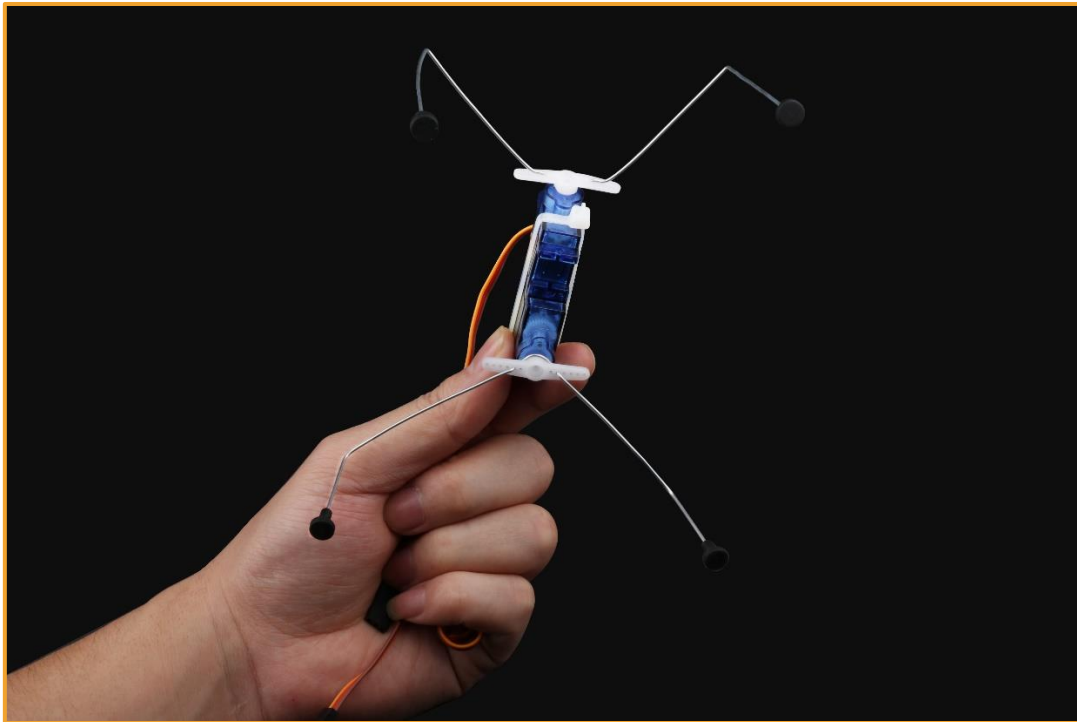


Figure-15

The supporting paper cut into the shape shown in Figure 16 (size can refer to Figure 16), when completed, as shown in Figure 17.

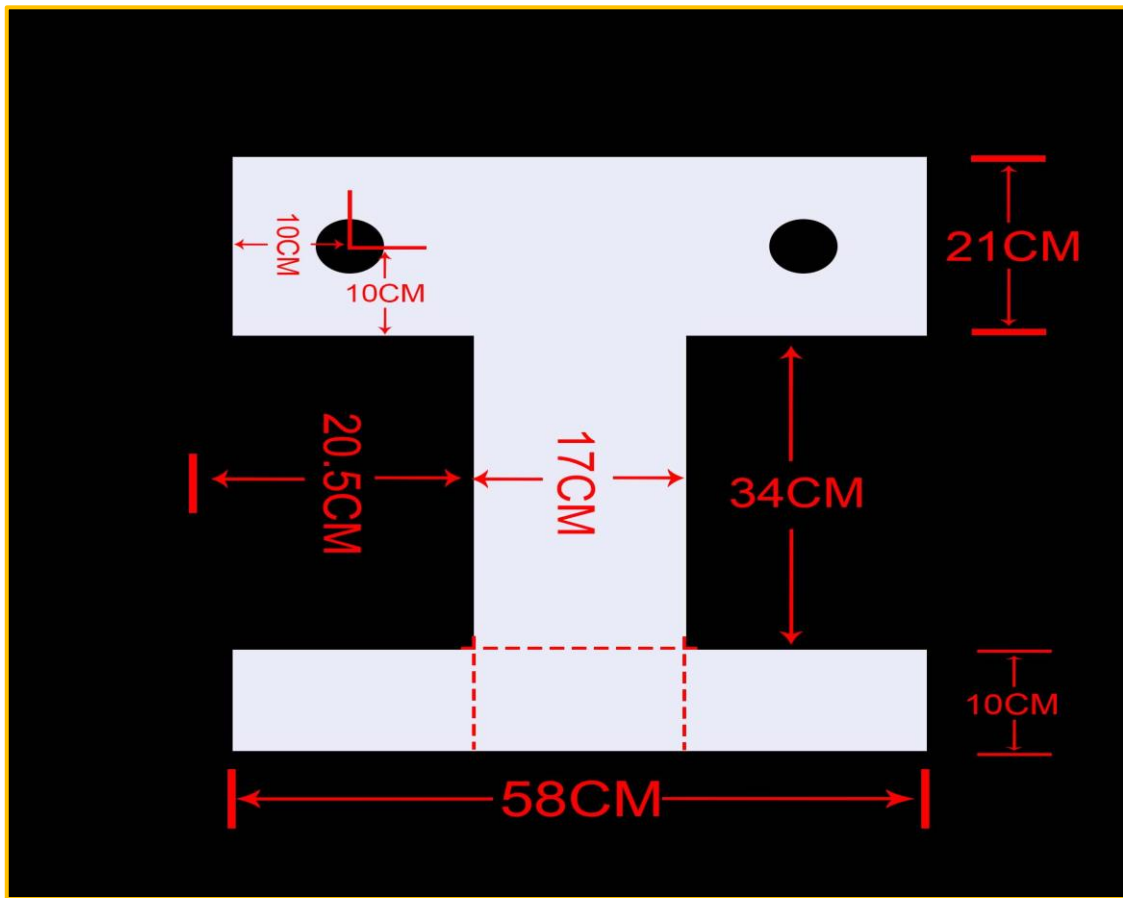


Figure-16

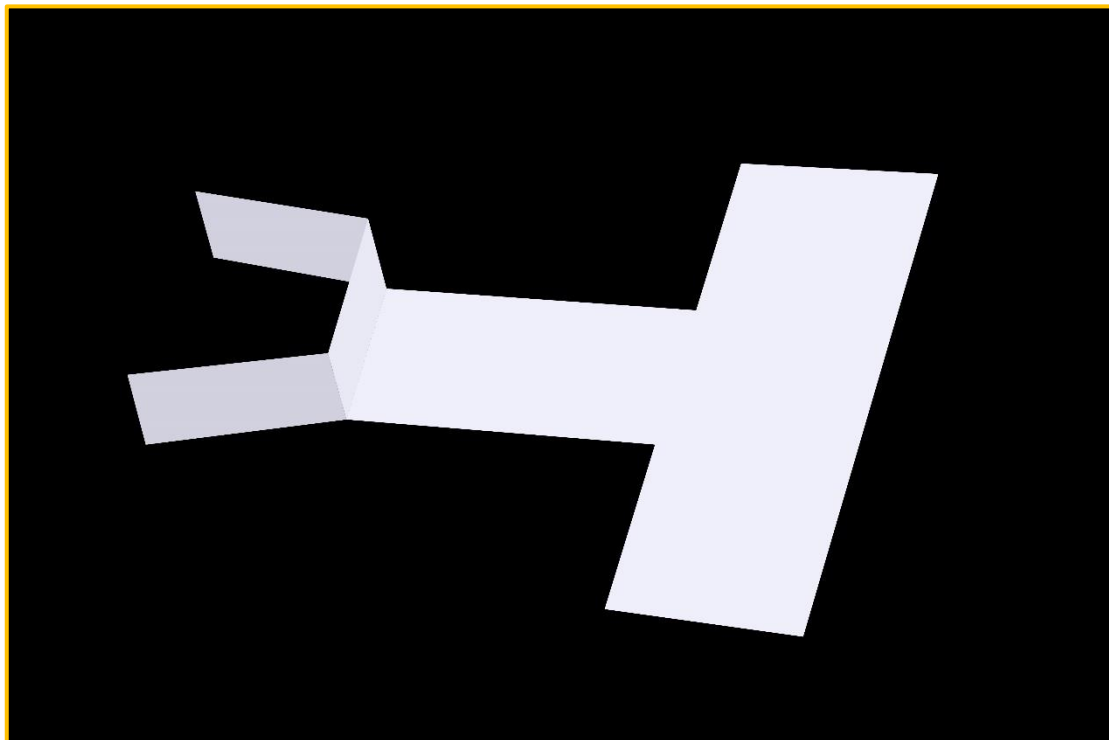


Figure-17

Screwing the support to the upper part of the servo motor, as shown in Figure 18.

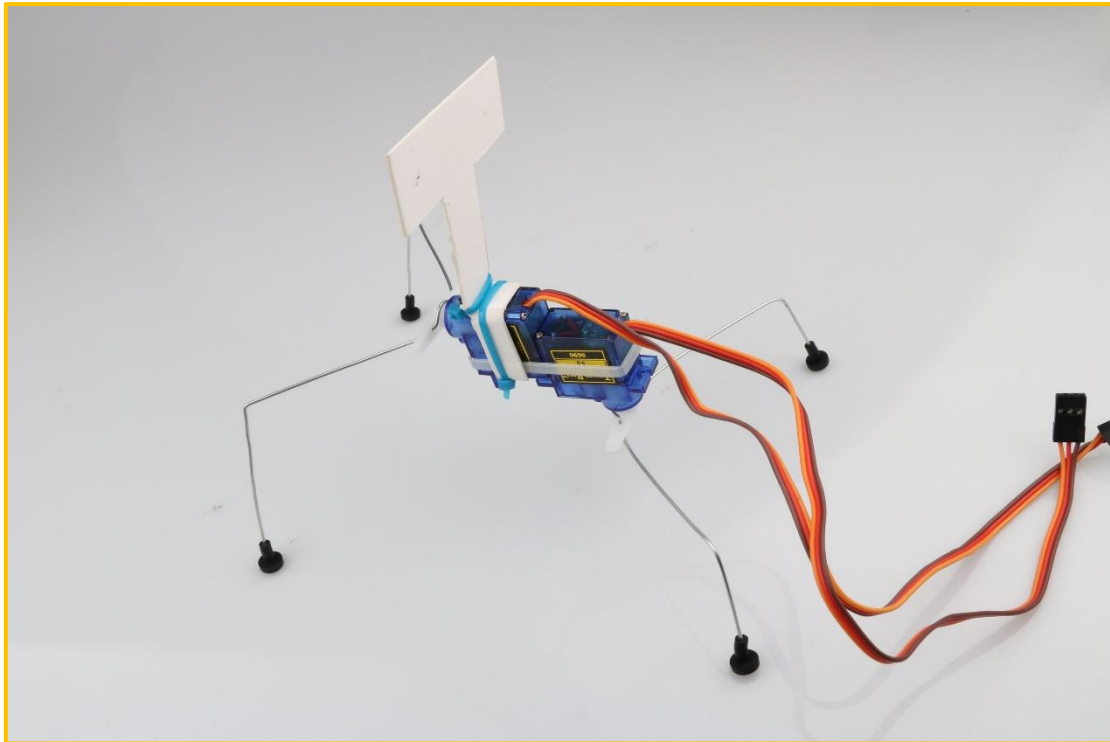


Figure-18

Figure-19. There is only one +5V pin and two GND pins on the Nano board, so it is necessary to make two wires below.

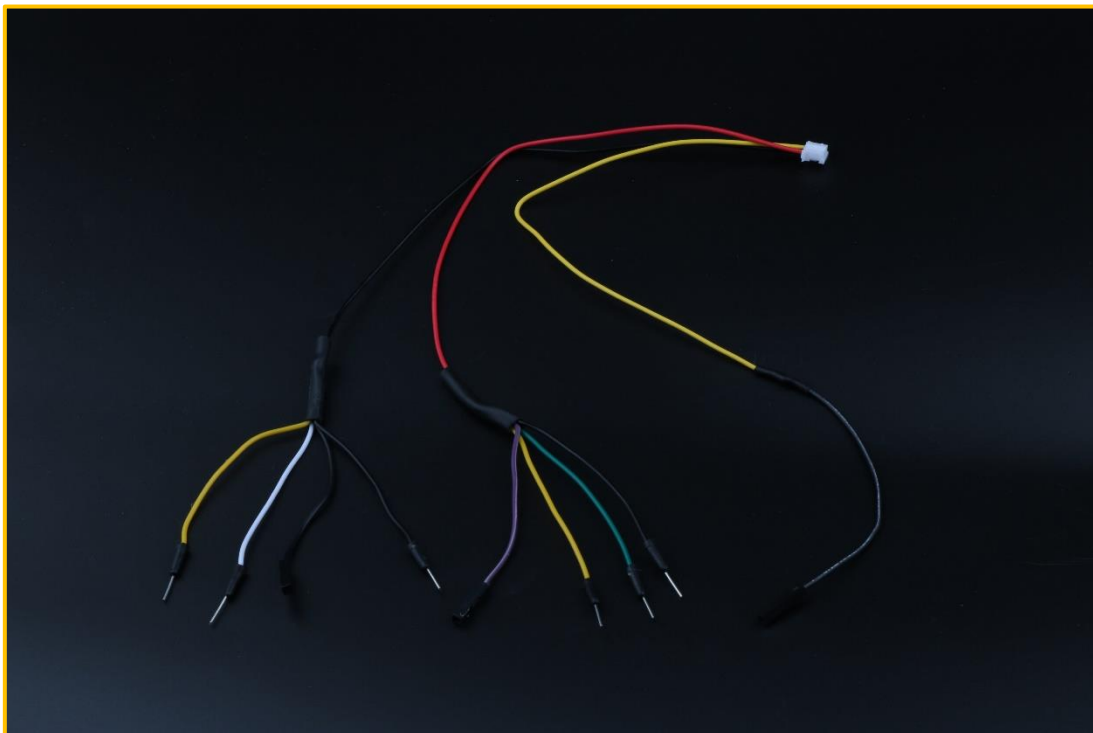


Figure-19

Figure 20. When connecting the wires, make sure you wrap every naked interface with adhesive tape like the top first one.

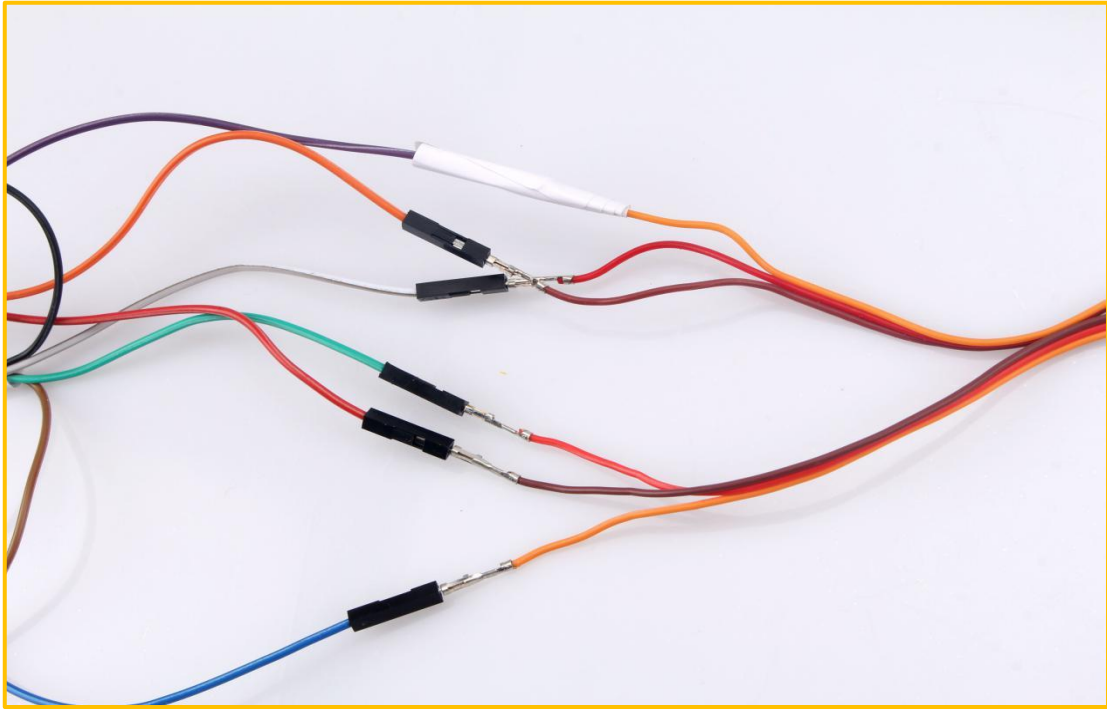


Figure-20

Putting the white interface of the black sensor up and place it on the holder. Locate the corresponding hole at both ends and screw two 2mm diameter holes. Then, as shown in Figure-21, the sensor is secured to the holder with two cable ties.

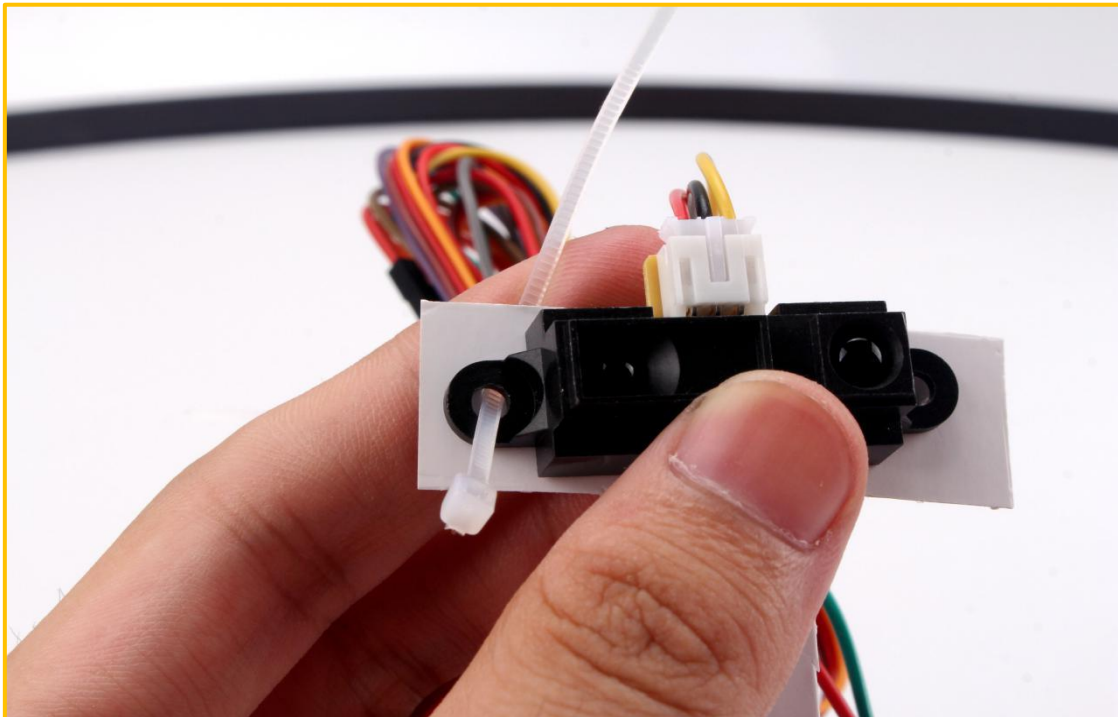


Figure-21

If you assemble the motors and sensors properly, the effect should be as shown in Figure-22.

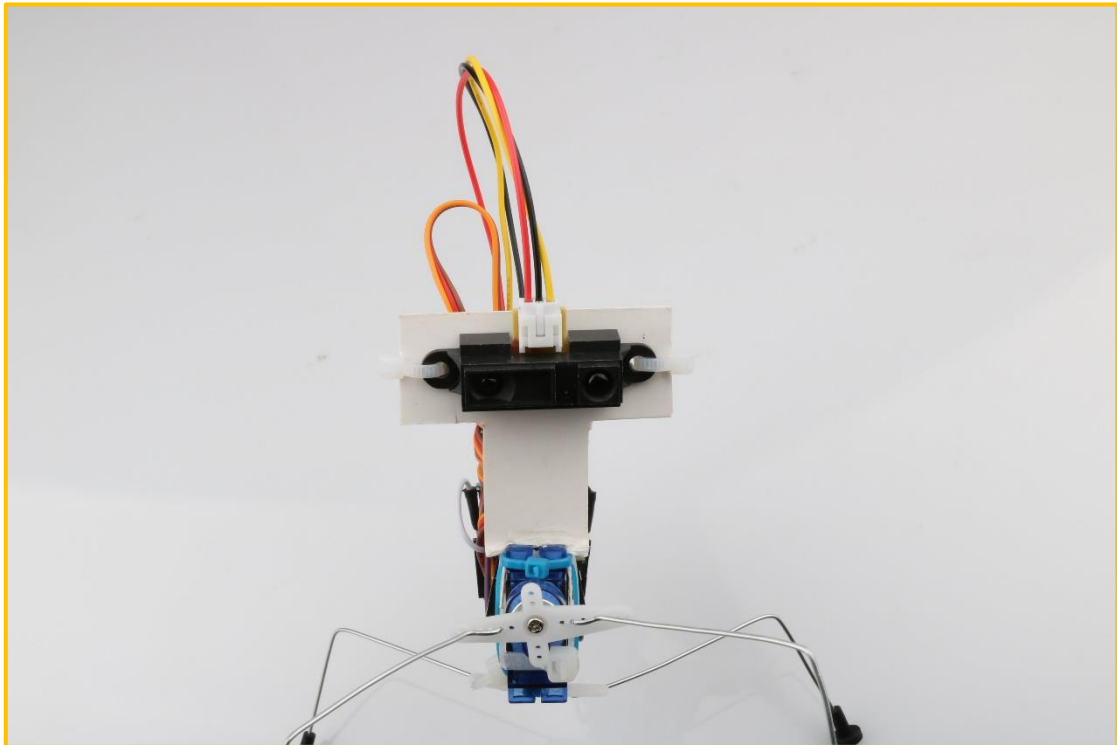


Figure-22

Figure-23. Now strapping FunDuino Nano onto the back of the servo motors.

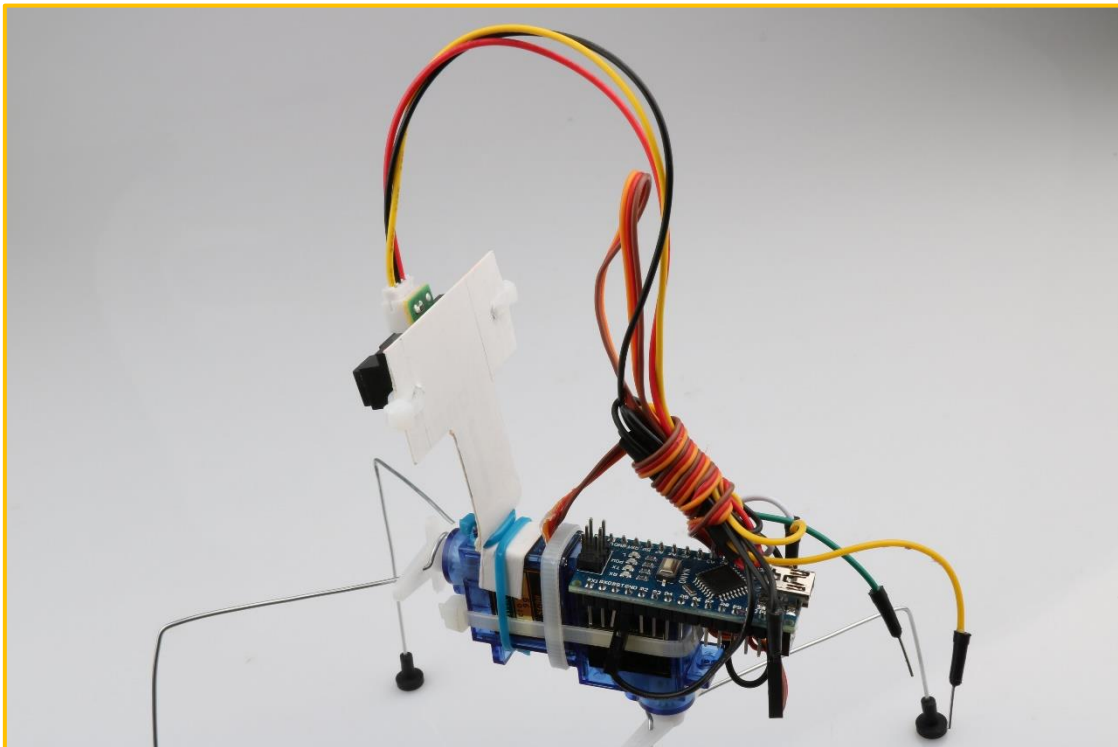


Figure-23

Figure-24. Reforming the battery like this.

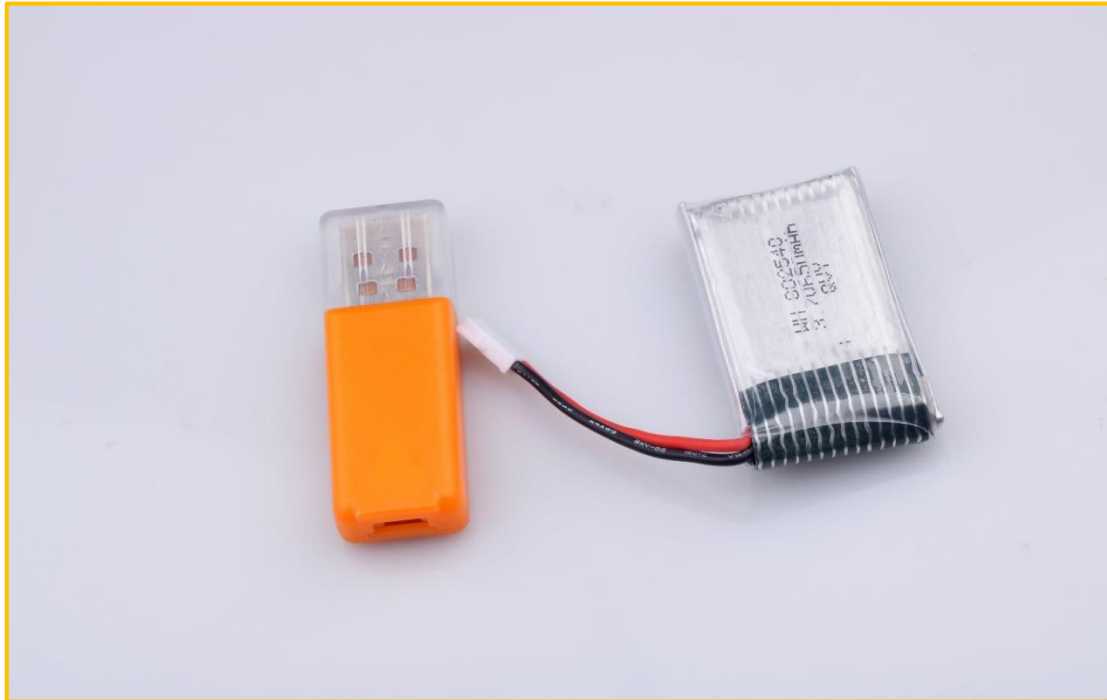


Figure-24

Figure-25. Now that the servos are centered. It's also a good idea to attach the battery now, because the additional weight will affect the way the robot walks. However, it is not necessary to connect the battery wires to the Funduino Nano; you can take power straight from the USB cable after you are programming and testing the device.

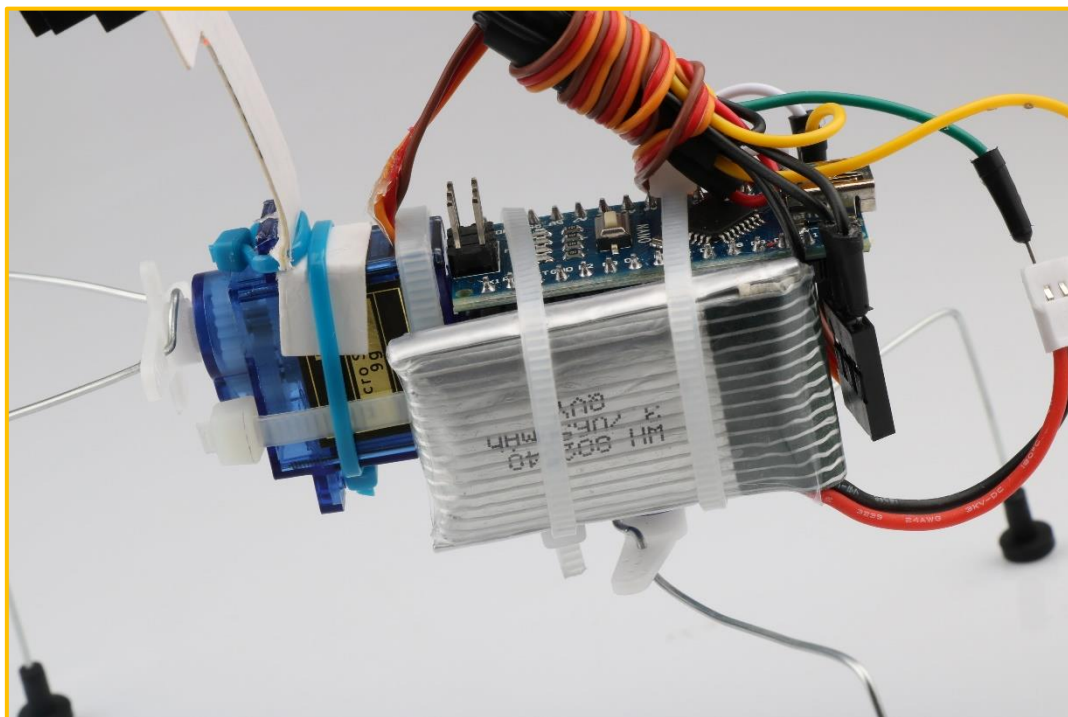


Figure-25

After the assembly is completed, you can first download the program to the Nano development board, adjust the steering servo the right direction to ensure that the robot can walk normally, Securing the front and rear legs to the servo motors by using a small screw. as shown in Figure -26.

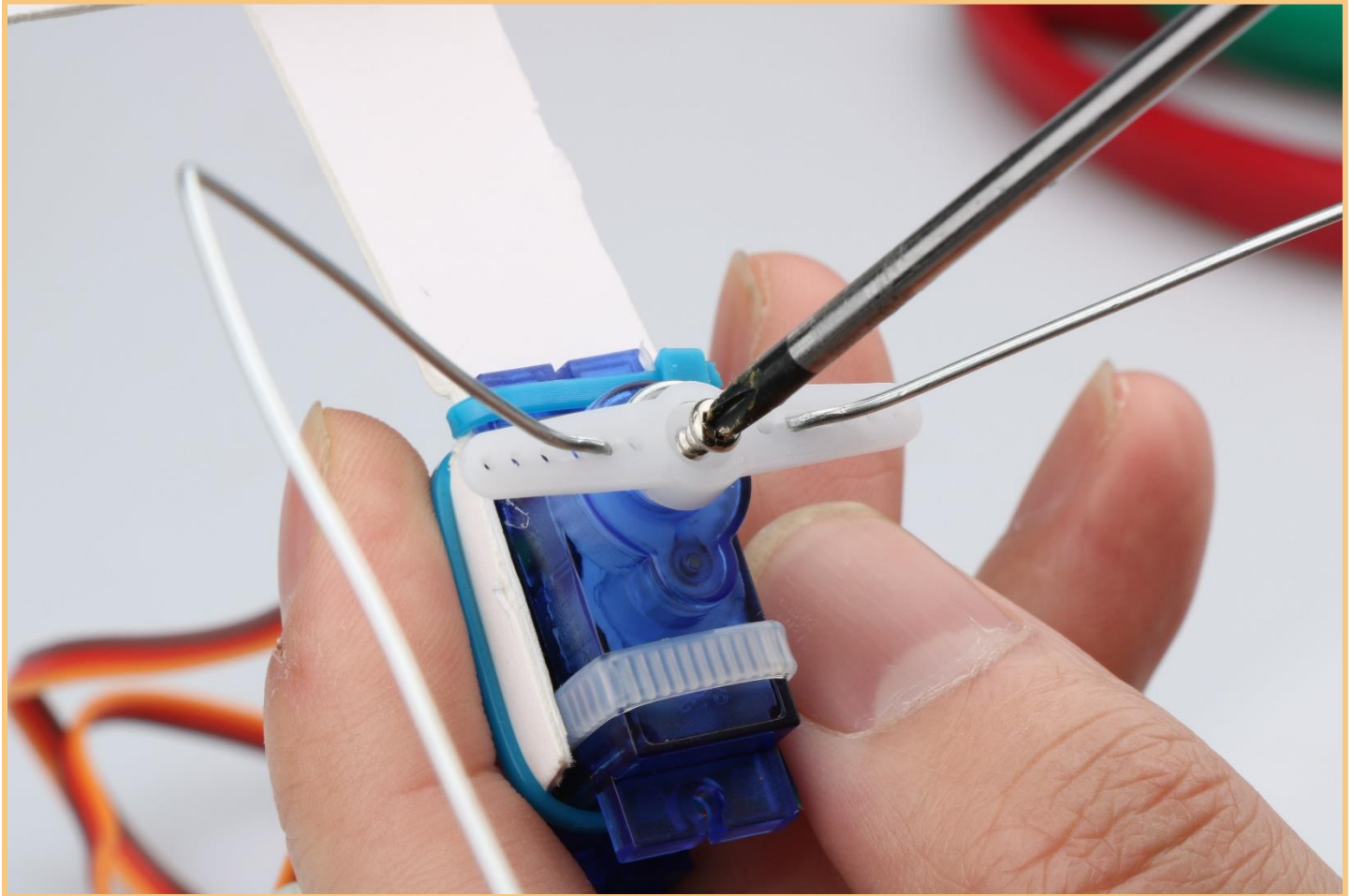


Figure-26

Figure-27. Finally, the complete insect robot is shown as below.

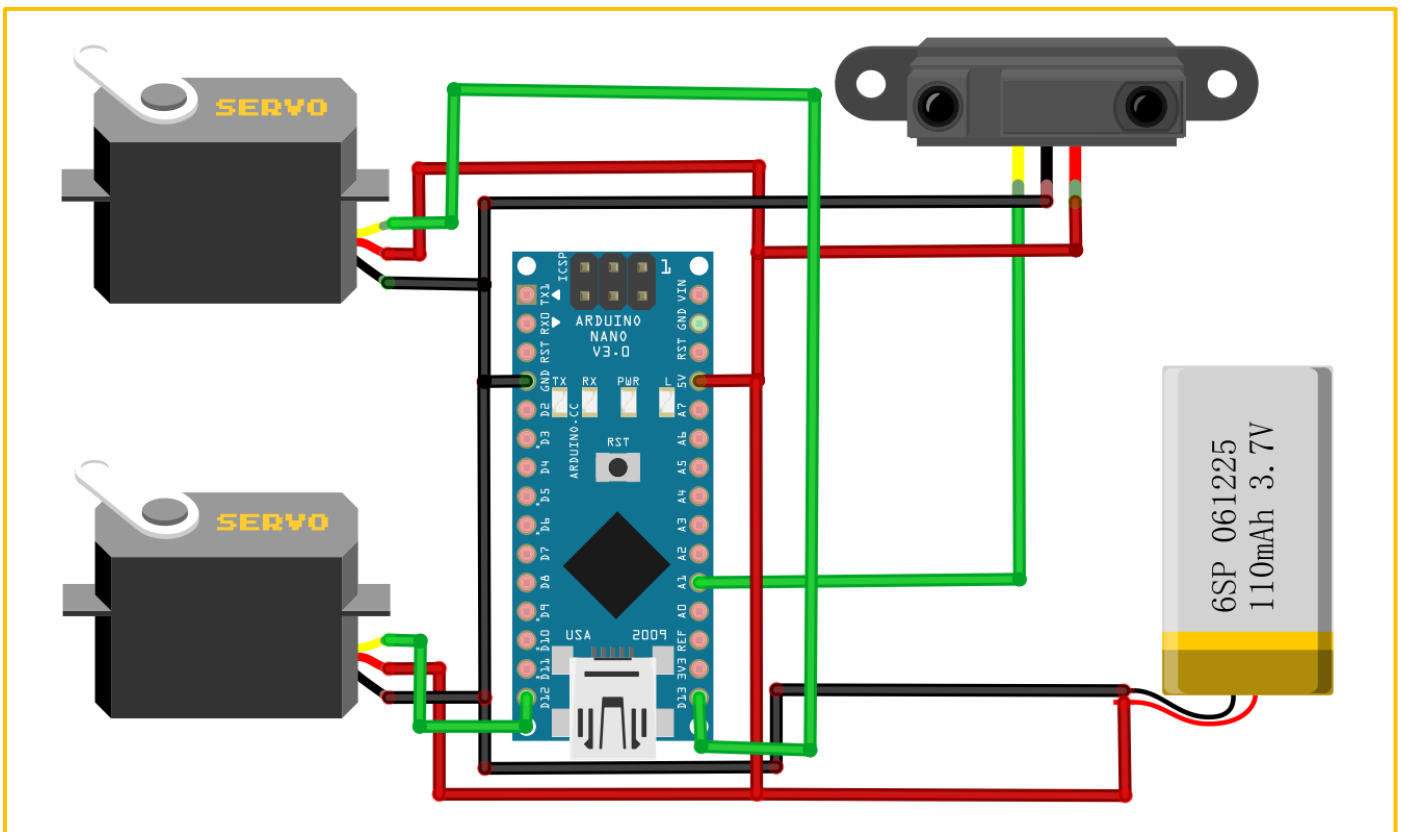


Figure-27

Note: If the robot walks slowly or the posture is very unstable, the wire bending angle and height, direction and so on need to be properly adjusted.

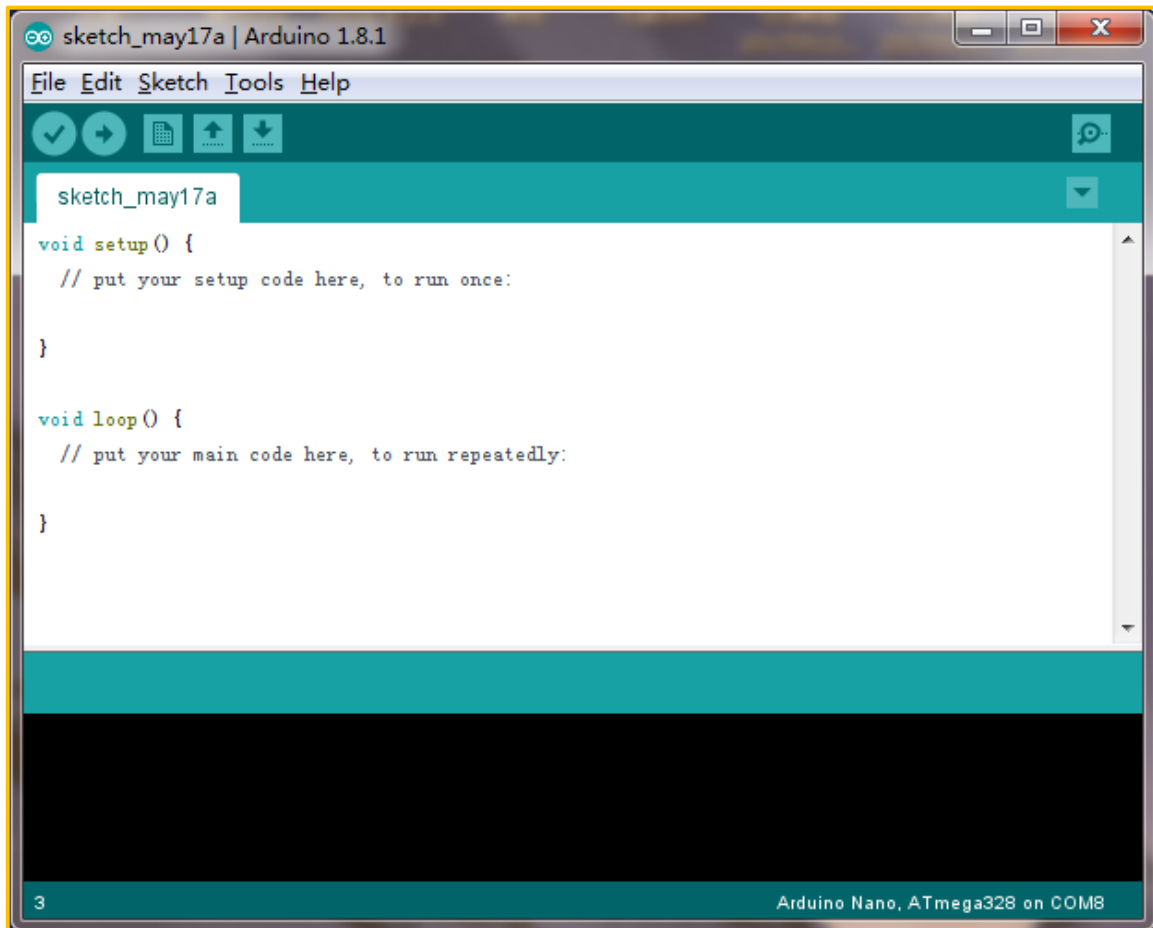
Wiring of Circuit

Funduino Nano	Front Servo
GND	-
+5v	+
13	data
Funduino Nano	Rear Servo
GND	-
+5v	+
12	data
Funduino Nano	Infrared Sensor
GND	-
+5v	+
A1	data
Funduino Nano	Battery
GND	-
+5v	+

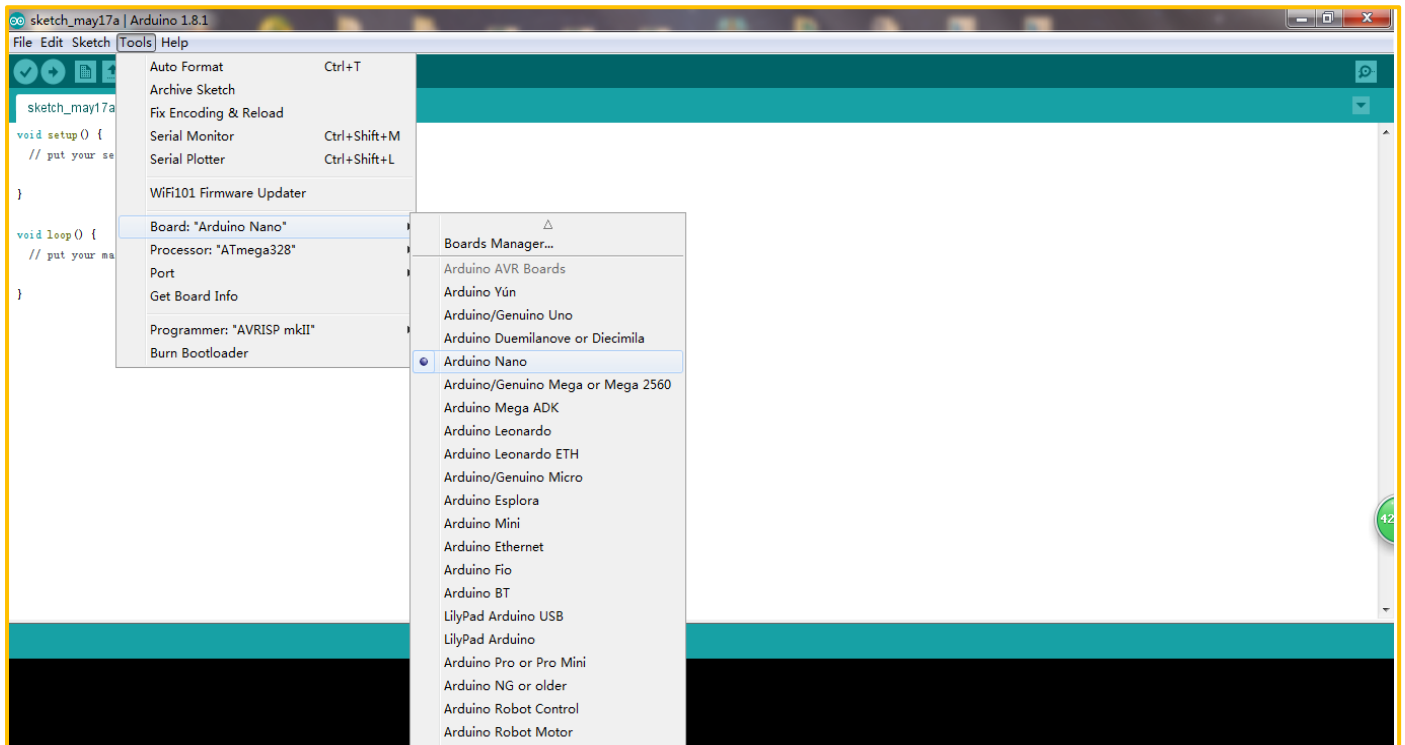


Programming

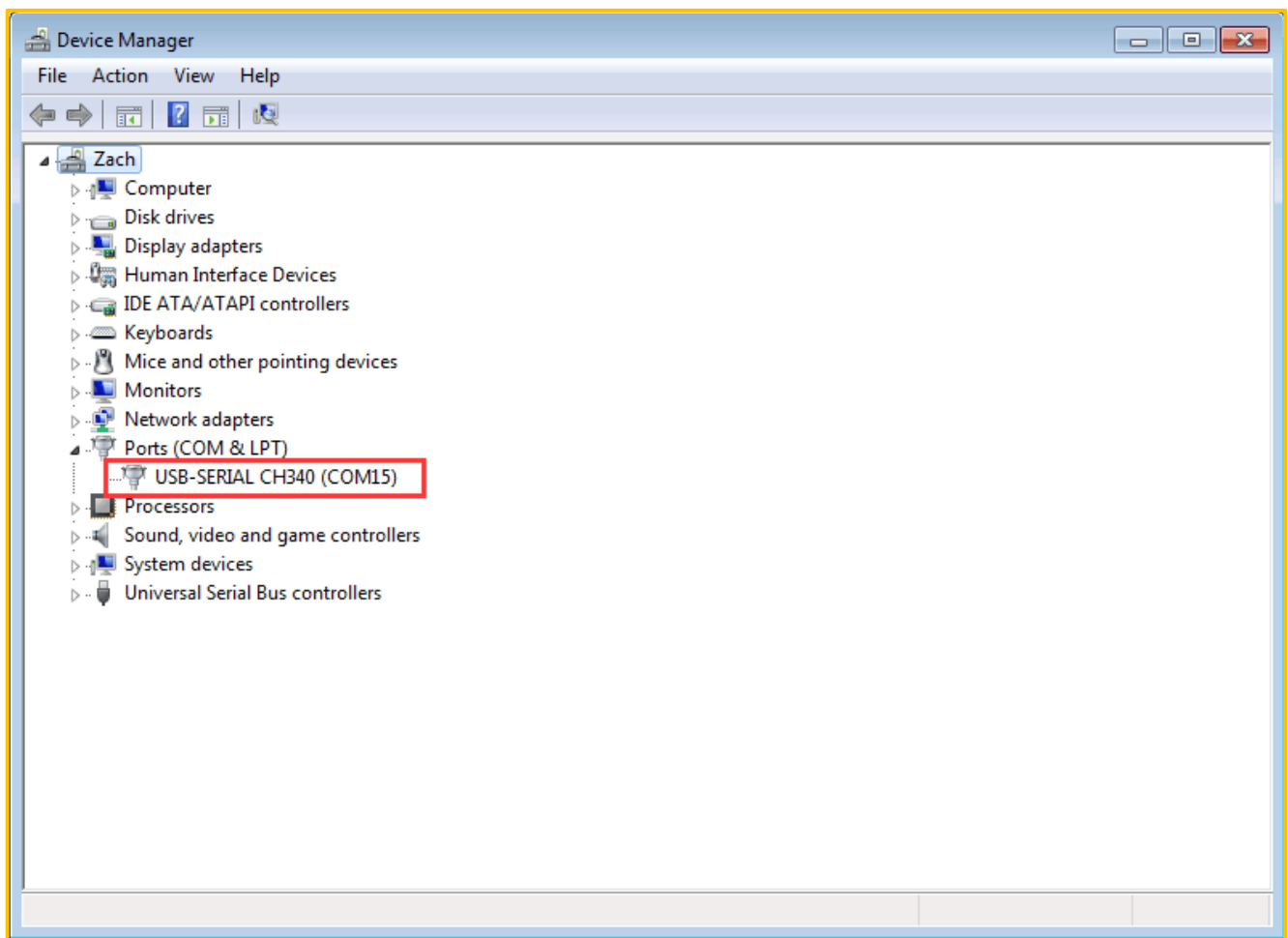
Step1: Arduino IDE, a programming software, is required to be installed into your computer. If already installed, please skip this step. The software can be obtained from <https://www.arduino.cc/en/Main/Software>



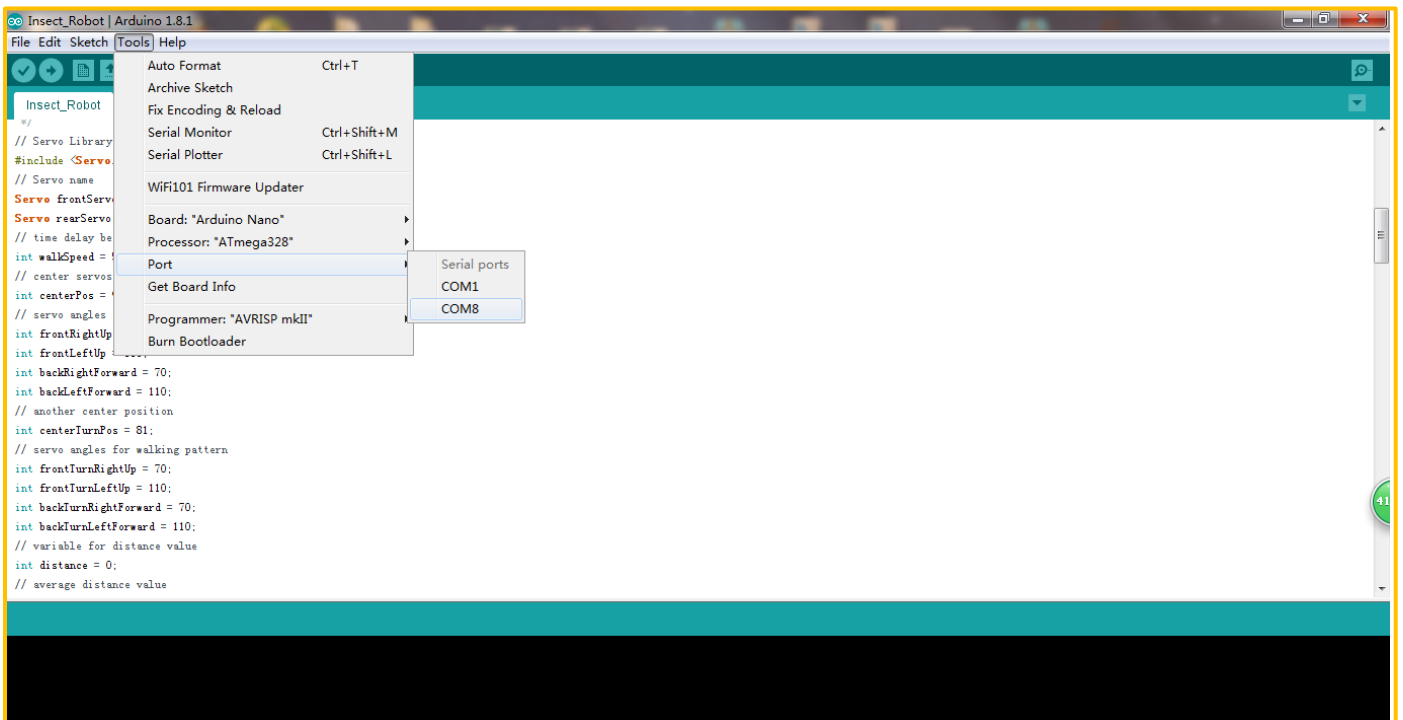
Step2: Choose “Arduino Nano” in Tools/Board





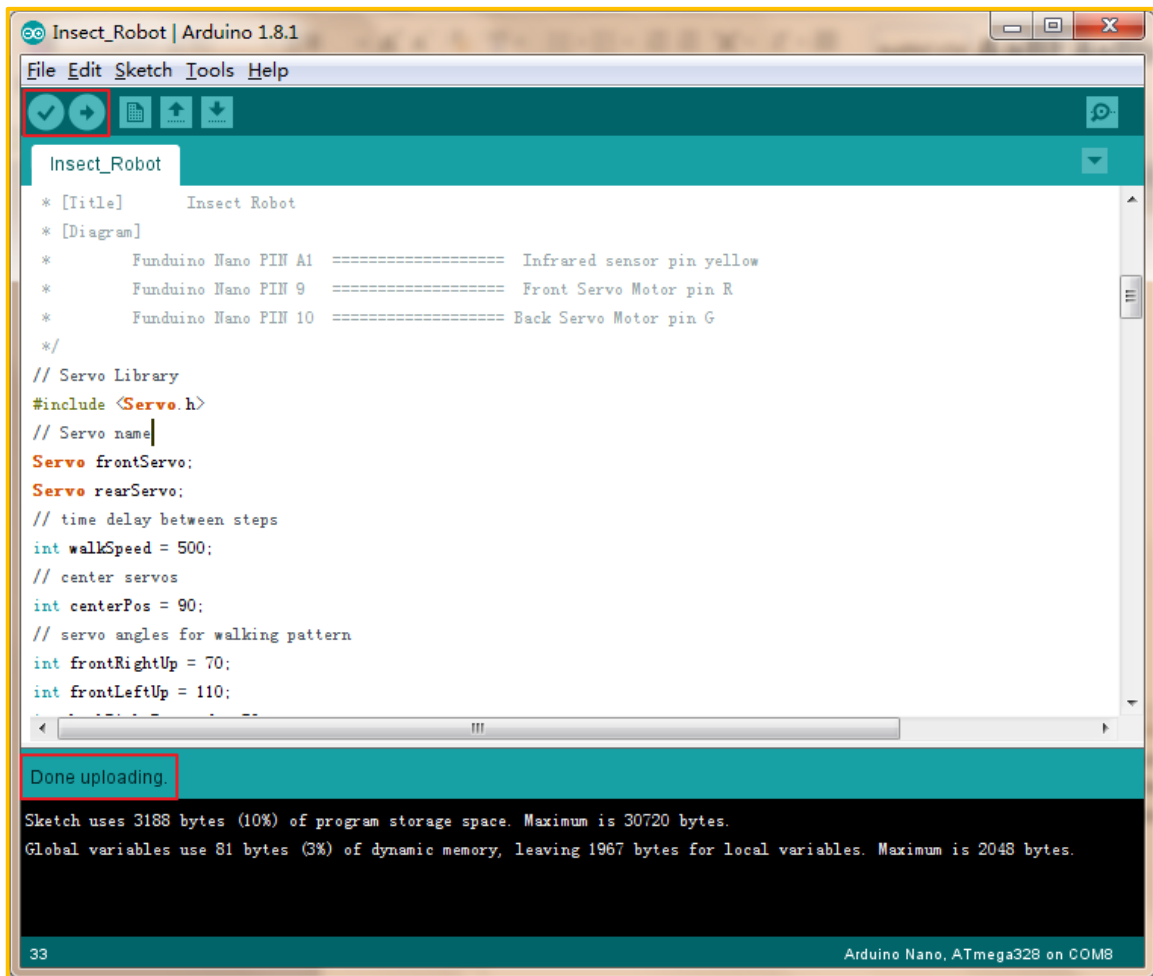
Step3: In the device manager of the Windows control panel, locate the port corresponding to Arduino Nano, such as COM15.



In the following Arduino programming software Tools/Port, selecting the port number you have found, such as COM8.



Step4: Open the *Insect_Robot.ino* program file, and then click  to check the program. Finally, click on  to upload the program into the controller. The white LED on the controller blinks for about 2 seconds, and the lower end of the window shows Done Uploading, indicating that the program has been uploaded successfully.



Step5: Now the robot has been already completed, and the program has been burned into controller. As long as we connect the +3.7V pin of the battery to the extended +5V pin on Funduino Nano, the insect robot is ready to take his first step.

Code

```
// Servo Library
#include <Servo.h>
// Servo name
Servo frontServo;
Servo rearServo;
// time delay between steps
int walkSpeed = 500;
// center servos
int centerPos = 90;
// servo angles for walking pattern
int frontRightUp = 70;
int frontLeftUp = 110;
int backRightForward = 70;
int backLeftForward = 110;
// another center position
int centerTurnPos = 81;
// servo angles for walking pattern
int frontTurnRightUp = 70;
int frontTurnLeftUp = 110;
int backTurnRightForward = 70;
int backTurnLeftForward = 110;
// variable for distance value
int distance = 0;
// average distance value
int distanceCheck = 0;
// Array for distance values
int collectDistance[5];
// Variables for counters
int i;
int f;
int r;
// assign analog pin A1
```

```
int sensorPin = A1;
// distance value for danger close. Bigger values are greater distance
// and smaller values are lower distance
int dangerDistance = 350;
/* Setup function */
void setup()
{
  // attach servos
  frontServo.attach(12);
  rearServo.attach(13);
  // assign sensor
  pinMode(sensorPin, INPUT);
  // center servos
  frontServo.write(centerPos);
  rearServo.write(centerPos);
  // wait 3 seconds for start walking
  delay(3000);
  //Serial.begin(9600); // serial data setup
}
/* distance check function */
void scan()
{
  // read 5 distance values
  for (i = 0; i < 5; i = i + 1) {
    distanceCheck = analogRead(sensorPin);
    collectDistance[i] = distanceCheck;
    // serial output for testing
    //Serial.print (i);
    //Serial.print(" = ");
    //Serial.println(collectDistance[i]);
  }
  // checksum of the 5 distance values for getting an average
  // value. This will prevent the robot to change behavior by reading one
  // wrong value
}
```

```
distance =
(collectDistance[0]+collectDistance[1]+collectDistance[2]+collectDistance[3]+collectDistance[4])/5;
    delay(20);
}
// walk forward
void moveForward()
{
    // loop for the servo angels to smoothen the steps
    for (f = 0; f < 39; f++){
        frontRightUp++;
        backLeftForward--;
        frontServo.write(frontRightUp);
        rearServo.write(backLeftForward);
        delay(10);
    }
    // loop for the servo angels to smoothen the steps
    for (r = 0; r < 39; r++){
        frontRightUp--;
        backLeftForward++;
        frontServo.write(frontRightUp);
        rearServo.write(backLeftForward);
        delay(10);
    }
}
// walk backwards to the left
void moveBackRight()
{
    frontServo.write(frontRightUp);
    rearServo.write(backRightForward-6);
    delay(110);
    frontServo.write(centerPos);
    rearServo.write(centerPos-6);
}
```



```
delay(80);
frontServo.write(frontLeftUp+9);
rearServo.write(backLeftForward-6);
delay(110);
frontServo.write(centerPos);
rearServo.write(centerPos);
delay(80);
}
// walk forward to the left
void moveTurnLeft()
{
frontServo.write(frontTurnRightUp);
rearServo.write(backTurnLeftForward);
delay(110);
frontServo.write(centerTurnPos);
rearServo.write(centerTurnPos);
delay(80);
frontServo.write(frontTurnLeftUp);
rearServo.write(backTurnRightForward);
delay(110);
frontServo.write(centerTurnPos);
rearServo.write(centerTurnPos);
delay(80);
}
// blink LED. This function can be called in any situation you want.
Just add led(); in the code where you want to blink the LED.
void led(){
    // loop for the LED to blink it 5 times for 0.05 sec on and 0.1 sec
off
    for(int l=0; l<=5; l++) {
        digitalWrite(13, HIGH);
        delay(50);
        digitalWrite(13, LOW);
        delay(100);
    }
}
```

```
    }  
}  
// that's the loop. This is repeatedly called as long as the robot is  
powered on  
void loop()  
{  
    // call function for checking the distance  
    scan();  
    //Serial.println(distance);  
    if (distance > 1){ // filters out the zero readings  
        // an obstacle is being detected  
        if (distance > dangerDistance) {  
            // LED at Pin 13 (standard) blinks 5x  
            led();  
            // 4 steps backward left  
            for(int i=0; i<=3; i++) {  
                moveBackRight();  
                delay(walkSpeed);  
            }  
            // 4 steps forward left  
            for(int i=0; i<=3; i++) {  
                moveTurnLeft();  
                delay(walkSpeed);  
            }  
        } else {  
            // all clear, no obstacle detected. Just walk forward  
            moveForward();  
            delay(walkSpeed/100);  
        }  
    }  
}
```