# Pars

ENGR102 Intro to Electronics

Shoreline Community College

December 2017

## Description of the problem. Why are you building this project?

According to a report study co-authored by the specialist in Youth Development named Bradley S. Barker and the Professor in Education Psychology John Ansorge, both from the University of Nebraska-Lincoln, the use of science and technology curriculum based on robotics translates into an increase of achievement scores in students. Pars is designed to aim at students who would like to learn about mechatronics and engineering while demonstrating the power of the DYI (Do It Yourself) method to build useful robots that complete specific tasks through circuitry combined with programming skills.

## What things can you do with this Pars?

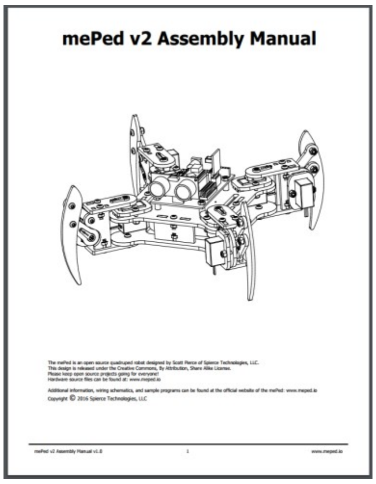
Pars is a walking and dancing robot that has been developed with open source software and hardware. Pars is based on Spierce Technology's walking robot mePed V2 (version 2). The core of Pars is the small yet powerful Arduino Nano Microcontroller and eight servos that allow Pars to move freely as commanded by an Infrared Remote Control. Pars will:

- Walk Forward, Left, Right or backwards at various speeds.
- Stand Up.
- Bow.
- Wave.
- Dance.
- Sit Down.
- Play Music.
- Trim right or left.
- Recalibrate its servos.
- Perform complex sequences as in button 5 (presentation) and button 9 (demonstration).

As commanded by a user via an Infrared Remote Control.

## Knowledge from which prior experiments is required?

How to assemble Spierce Technology's mePed V2.0



mePed v2 Assembly Manual

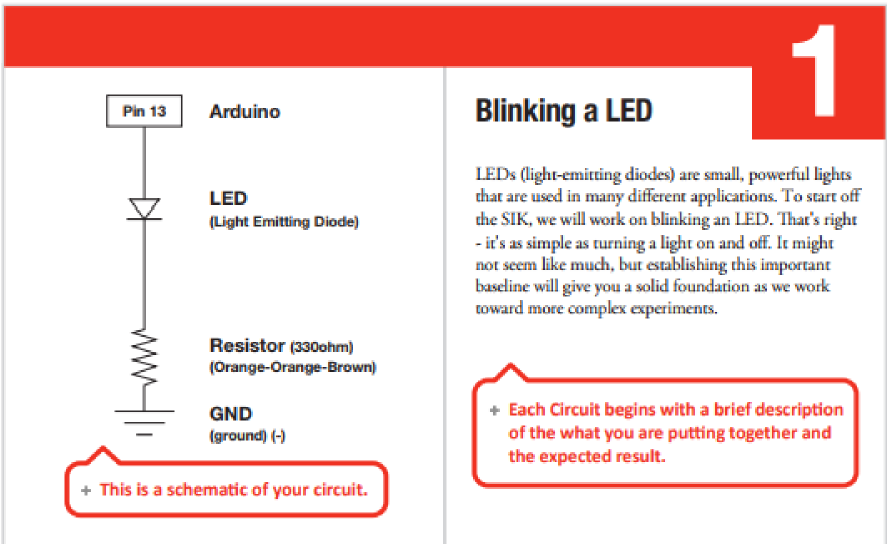
The mePed is an open source quadruped robot designed by Scott Pierce of Spierce Technologies, LLC.
This design is released under the Creative Commons, By Attribution, Share Alike License.
Please keep open source projects going for everyone!
Hardware source files can be found at: www.meped.io

Additional information, wiring schematics, and sample programs can be found at the official website of the mePed: www.meped.io
Copyright © 2016 Spierce Technologies, LLC

mePed v2 Assembly Manual v1.0 1 www.meped.io

**Figure 1.** Snippet of Spierce Technology's mePed V2 Assembly Manual (available in references).

Knowledge from Sparkfun's SIK guide experiments number 1, 8 and 11.

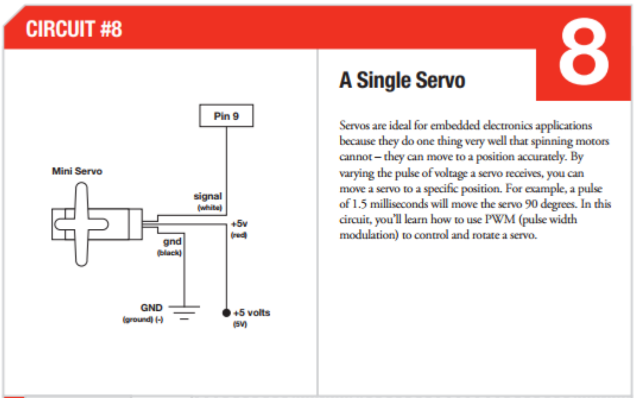


Pin 13
Arduino
LED
(Light Emitting Diode)
Resistor (330ohm)
(Orange-Orange-Brown)
GND
(ground) (-)
+ This is a schematic of your circuit.
1
+ Each Circuit begins with a brief description of the what you are putting together and the expected result.

Blinking a LED

LEDs (light-emitting diodes) are small, powerful lights that are used in many different applications. To start off the SIK, we will work on blinking an LED. That's right - it's as simple as turning a light on and off. It might not seem like much, but establishing this important baseline will give you a solid foundation as we work toward more complex experiments.

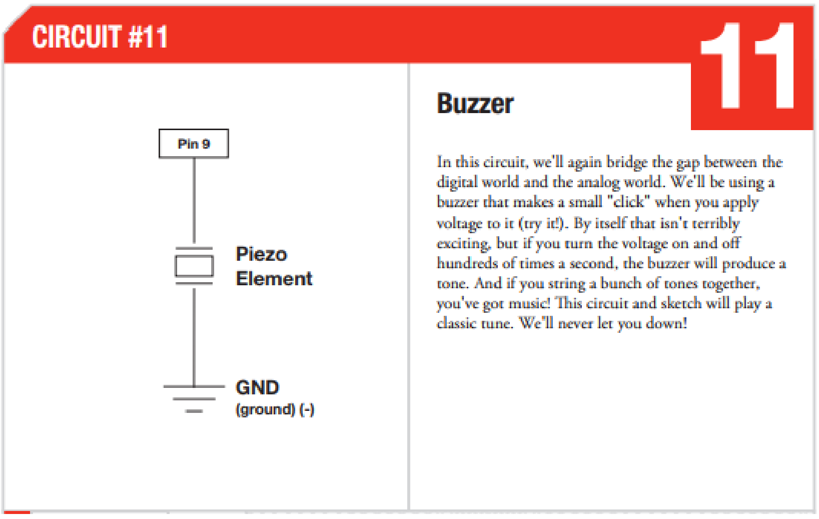
**Figure 2.** Schematic of Sparkfun's SIK Guide Experiment 1: Blinking a LED.



A Single Servo

Servos are ideal for embedded electronics applications because they do one thing very well that spinning motors cannot – they can move to a position accurately. By varying the pulse of voltage a servo receives, you can move a servo to a specific position. For example, a pulse of 1.5 milliseconds will move the servo 90 degrees. In this circuit, you'll learn how to use PWM (pulse width modulation) to control and rotate a servo.

**Figure 3.** Schematic of Sparkfun's SIK Guide Experiment 8: A Single Servo (available in references).



CIRCUIT #11

11

Pin 9
Piezo Element
GND (ground) (-)

**Buzzer**

In this circuit, we'll again bridge the gap between the digital world and the analog world. We'll be using a buzzer that makes a small "click" when you apply voltage to it (try it!). By itself that isn't terribly exciting, but if you turn the voltage on and off hundreds of times a second, the buzzer will produce a tone. And if you string a bunch of tones together, you've got music! This circuit and sketch will play a classic tune. We'll never let you down!

**Figure 4.** Schematic of Sparkfun's SIK Guide Experiment 11: Buzzer.

Elegoo's 37 Sensor Kit Tutorial explains how to use an Ultrasonic Sensor Module, an Infrared Receiver and emitter in a simple manner.

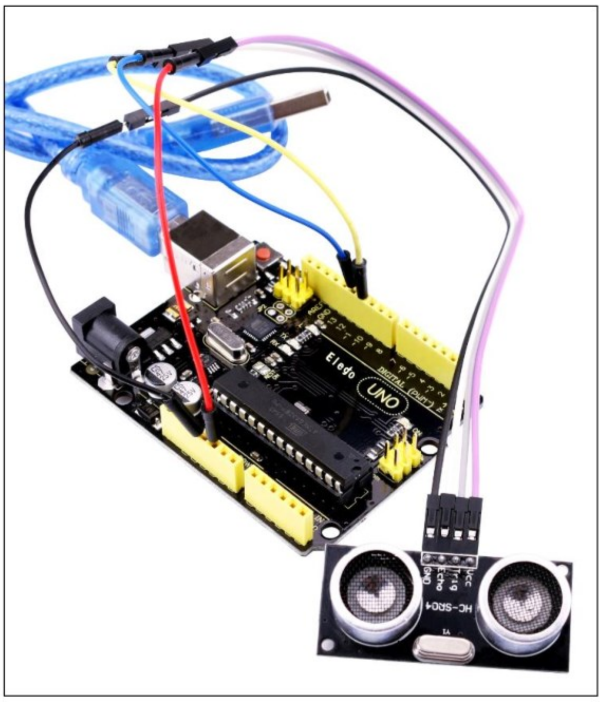


**Figure 5.** Snippet of Elegoo's 37 Sensor Kit Tutorial: Ultrasonic Sensor Module.

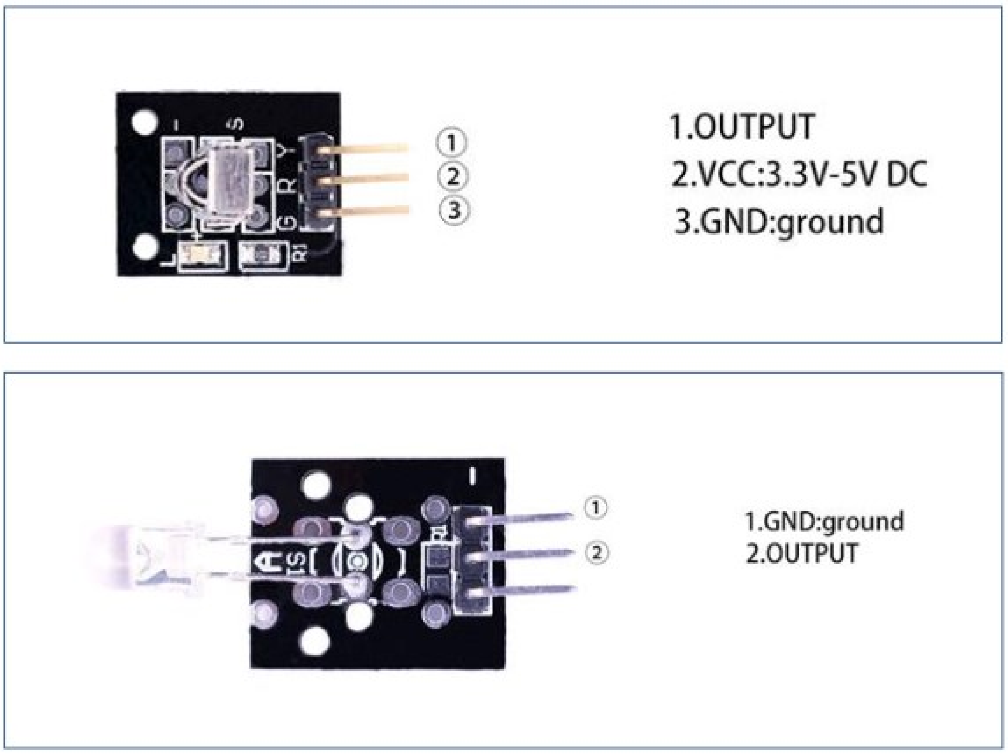


**Figure 6.** Snippet of Elegoo's 37 Sensor Kit Tutorial: IR Receiver and Emission.

**Parts Required**

- Spierce Technology's mePed V2.0 Complete Kit:

- Body Wood Kit.
- Eight (8) MG90S micro-servo.
- One (1) Ultrasonic sensor.
- One (1) Infrared Remote Control.
- One (1) Arduino Nano Microcontroller.
- Forty (40) M3 x 10mm Screws.
- Forty (40) M3 x 10mm Screws.
- Sixteen (16) M3 x 16mm Screws.
- Twenty-eight (28) M3 Hex Nuts.
- Thirty-two (32) M3 Nyloc Hex Nuts.
- Four (4) Circuit Board Spacers.
- One (1) mePed Circuit Board.

- One (1) Infrared Sensor.
- One (1) AA Battery Holder.

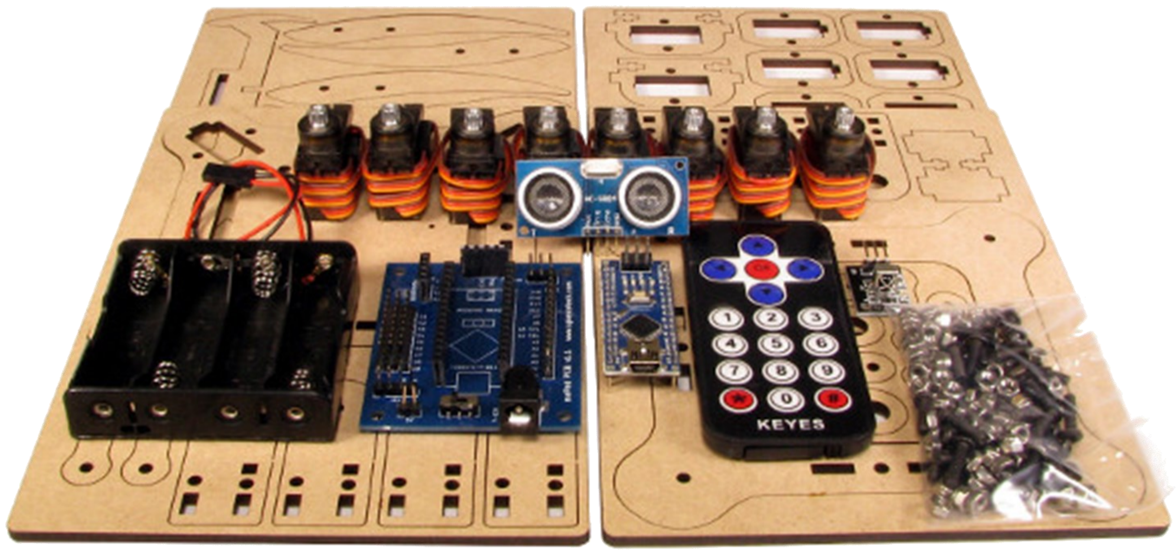


**Figure 7.** mePed's V2 Complete Kit.

- One (1) color LED.
- One (1) Passive Buzzer.
- Four (4) AA Batteries.
- Wire.
- Optional soldering supplies.

## Problem-solving approach

1) Learn how to Assemble Spierce Technology's mePed V2 (and execute).
2) Learn how to use the Arduino Nano Microcontroller and mePed's circuit board.
3) Learn how to use and modify mePed V2's starter file.
4) Learn how to turn an LED on/off.
5) Learn how to use an Ultrasonic sensor to measure distance.
6) Learn how to use a Buzzer.
7) Develop a program that can play melodies through the buzzer.
8) Integrate the Ultrasonic sensor as an input device that modifies the frequency and speed of the buzzer's melody which is a function of the distance of the nearest obstacle in front of the Ultrasonic sensor.
9) The result of (8) is then integrated into the mePed V2.0 walking robot.
10) A color LED is integrated into the result of (9) as an indicator.

11) Program mePed's behavior until it becomes our so desired Pars.

12) Test, debug and fix as required through every step.

## Problems

- Two bad servos: Along with the Spierce Technology's mePed V2 Kit I received two servos that were broken. During the early testing this issue was discovered when two of the pivot servos would not allow Pars to move properly. The servos were replaced by two SMARKN's MG90S micro servos which fixed this issue.
- No multithreading: The Arduino's lack of ability to multithreading and the way the mePed is designed makes it impossible to do multiple tasks at once. This issue has consequences through the Infrared Remote Control which is not always listening for new instructions and the buzzer which can not play music after certain speeds which would otherwise be possible to achieve. The remote issue is partially fixed by an LED indicator that indicates when you should be sending a new instruction and by adding code that checks for new instructions more frequently. The buzzer issue is fixed by picking the right frequencies and delay times between notes and actions conveniently.
- Conflicting Libraries: The tone library conflicts with the servo library due to the timers inside the Arduino. The library newTone uses another timer, but due to the quantity of servos in Pars and mePed it caused more conflicts as servo.h needs to use more timers more frequently. The issue was fixed through the TimerFreeTone Library for Arduino which sacrifices accuracy and pitch by not using any timers.
- Power Issue: The mePed V2 is designed to be powered by four AA batteries. However, it is usually not enough to power mePed or Pars properly for a considerable amount of time. In the mePed's forums there is documentation from the designers of mePed V2 themselves about power issues. The four AA batteries are not enough to power the eight servos in most surfaces as the servos will require more power and then either the Arduino will reset itself due to the increase in current or the servos will not move due to lack of enough power.

## Code

```
/*
 * Name: Rafael Antonio Laya Alfonzo
 * December 2017
 * Class: ENGR 102 intro to electronics (Fall 2017)
 * Software Version: Arduino 1.8.2
 * Source code for the walking and dancing robot: Pars
 * The reader can use this code as they please but I won't be liable of anything
 * This is based on the meped v2 (http://www.meped.io/mepedv2)
*/

#include <IRremote.h>
#include <Servo.h>

// this library replaces tone() and newTone() because those have conflicts with
servo.h
#include <TimerFreeTone.h>

// these definitions makes it easier for us to code
#define FORWARD 0
#define BACK 1
#define LEFT 2
#define RIGHT 3
#define BOW 4
#define WAVE 5
#define SPEEDUP 6
#define SPEEDDOWN 7
#define DOWN 9

// Define USRF pins and variables
#define trigPin A3
#define echoPin A2
#define INCH 0
#define CM 1

// Define IR Remote Button Codes
#define irUp  16736925
#define irDown 16754775
#define irRight 16761405
#define irLeft 16720605
#define irOK 16712445
#define ir1 16738455
#define ir2 16750695
#define ir3 16756815
#define ir4 16724175
#define ir5 16718055
#define ir6 16743045
#define ir7 16716015
#define ir8 16726215
#define ir9 16734885
#define ir0 16730805
#define irStar 16728765
#define irPound 16732845
#define irRepeat 4294967295
```

```
//=====                                                              Globals
=================================================================
==============
int beats;
bool play;
long distance;
int ledPin = A3;
bool demoing;

// calibration
int da =  -12,  // Left Front Pivot
    db =   10,  // Left Back Pivot
    dc =  -18,  // Right Back Pivot
    dd =   12;  // Right Front Pivot

// servo initial positions + calibration
int a90  = (90  + da),
    a120 = (120 + da),
    a150 = (150 + da),
    a180 = (180 + da);

int b0   = (0   + db),
    b30  = (30  + db),
    b60  = (60  + db),
    b90  = (90  + db);

int c90  = (90  + dc),
    c120 = (120 + dc),
    c150 = (150 + dc),
    c180 = (180 + dc);

int d0   = (0   + dd),
    d30  = (30  + dd),
    d60  = (60  + dd),
    d90  = (90  + dd);

// start points for servo
int s11 = 90; // Front Left Pivot Servo
int s12 = 90; // Front Left Lift Servo
int s21 = 90; // Back Left Pivot Servo
int s22 = 90; // Back Left Lift Servo
int s31 = 90; // Back Right Pivot Servo
int s32 = 90; // Back Right Lift Servo
int s41 = 90; // Front Right Pivot Servo
int s42 = 90; // Front Right Lift Servo

int f    = 0;
int b    = 0;
int l    = 0;
int r    = 0;
int spd  = 5;  // Speed of walking motion, larger the number, the slower the speed
int high = 0;   // How high the robot is standing
```

```
// set up some constants for the buzzer
const int buzzerPin = 10;
const int tempo = 113;

// Define 8 Servos
Servo myServo1; // Front Left Pivot Servo
Servo myServo2; // Front Left Lift Servo
Servo myServo3; // Back Left Pivot Servo
Servo myServo4; // Back Left Lift Servo
Servo myServo5; // Back Right Pivot Servo
Servo myServo6; // Back Right Lift Servo
Servo myServo7; // Front Right Pivot Servo
Servo myServo8; // Front Right Lift Servo

// Set up IR Sensor
int irReceiver = 12;       // Use pin D12 for IR Sensor
IRrecv irrecv(irReceiver); // create a new instance of the IR Receiver
decode_results results;
//============================================================
============================

//=====                                                  Setup
============================================================
================
void setup()
{
  pinMode(ledPin, OUTPUT);

  // Attach servos to Arduino Pins
  myServo1.attach(2);
  myServo2.attach(3);
  myServo3.attach(4);
  myServo4.attach(5);
  myServo5.attach(6);
  myServo6.attach(7);
  myServo7.attach(8);
  myServo8.attach(9);

  pinMode(buzzerPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  //start the receiver
  irrecv.enableIRIn();
}
//============================================================
============================

//==                                                      Loop
============================================================
===================
void loop()
{
  // keep track of the received values from the IR Remote
  unsigned long value;
  unsigned long lastValue;

  center_servos();  // Center all servos
  high = 15;        // Set high servo position to 15
  spd = 5;          // Set speed to 3

  while (true)    // Loop forever
  {
    // if we're looping we're not demoing yet
    demoing = false;
    if (irrecv.decode(&results)) // If we have received an IR signal
    {
      value = results.value;

      // check if we've received something new
      if (value == irRepeat)
        value = lastValue;

      // do as instructed by the user
      switch (value)
      {
        case irUp:
          lastValue = irUp;
          forward();
          break;

        case irDown:
          lastValue = irDown;
          back();
          break;

        case irRight:
          lastValue = irRight;
          turn_right();
          break;

        case irLeft:
          lastValue = irLeft;
          turn_left();
          break;

        case irOK:
          lastValue = irOK;
          center_servos();
          break;

        case ir1:
          lastValue = ir1;
          bow();
          break;

        case ir2:
          lastValue = ir2;
          wave();
          break;

        case ir3:
          lastValue = ir3;
          increase_speed();
          break;

        case ir4:
          lastValue = ir4;
          dance();
          break;

        case ir5:
          lastValue = ir5;
          dancePresentation();
          break;

        case ir6:
          lastValue = ir6;
          decrease_speed();
          break;

        case ir7:
          lastValue = ir7;
          goDown();
          break;

        case ir8:
          lastValue = ir8;
          discoteca();
          break;

        case ir9:
          lastValue = ir9;
          demo();
          break;

        case ir0:
          lastValue = ir0;
          calibrate();
          break;

        case irStar:
          lastValue = irStar;
          trim_left();
          break;
        case irPound:
          lastValue = irPound;
          trim_right();
          break;

        default:
          break;
      }

      // this led makes it easier for the user to send signals to the robot
      digitalWrite(ledPin, HIGH);
      irrecv.resume(); //next value
      delay(50);  // Pause for 50ms before executing next movement
      digitalWrite(ledPin, LOW);
    }
  }
}

// this function is used for demonstration purposes
void demo()
{
  demoing = true; // we're demoing
  demoMovement(); // forward, back, left and right
  repeatnTimes(DOWN, 5);
  repeatnTimes(BOW, 5);
  repeatnTimes(WAVE, 2);
  dance();
  center_servos();
  dancePresentation();
  delay(1000);
  discoteca();
  center_servos();
}

// this functions is for movement testing: forward, back, left, right
void demoMovement()
{
  center_servos();
  repeatnTimes(FORWARD, 6);
  repeatnTimes(BACK, 6);
  repeatnTimes(LEFT, 2);
  repeatnTimes(RIGHT, 2);
  repeatnTimes(LEFT, 1);
}

// 0: forward
// 1: back
// 2: left
// 3: right
// 4: bow
// 5: wave
// 6: speed down
// 7: speed up
// 8: center_servos()
// 9: goDown() and center_servos();
// repeats action 'action' 'n' times
void repeatnTimes(int action, int times)
{
  for(int i = 0; i < times; i++)
  {
    switch(action)
    {
      case 0:
        forward();
        delay(50);
        break;

      case 1:
        back();
        delay(50);
        break;

      case 2:
        turn_left();
        delay(50);
        break;

      case 3:
        turn_right();
        delay(50);
        break;

      case 4:
        bow();
        break;

      case 5:
        wave();
        break;

      case 6:
        increase_speed();
        break;

      case 7:
        decrease_speed();
        break;

      case 8:
        center_servos();
        break;

      case 9:
        goDown();
        delay(300);
        center_servos();
        delay(300);
        break;

      default:
        return;
    }
  }
  center_servos();
}

// this function calibrates the servos
// which would usually decalibrate by trim_right() and trim_left()
void calibrate()
{
  // calibration
    da =  -12,  // Left Front Pivot
    db =   10,  // Left Back Pivot
    dc =  -18,  // Right Back Pivot
    dd =   12;  // Right Front Pivot
}

// The robot walks, waves and dances, then bows and comes back to its original
position
void dancePresentation()
{
  repeatnTimes(FORWARD, 6);
  wave();
  dance();
  center_servos();
  wave();
  bow();
  turnAround(RIGHT);
  repeatnTimes(FORWARD, 6);
  turnAround(LEFT);
  bow();
  center_servos();
}

// walk n steps forward
void forwardnSteps(int n)
{
  for(int i = 0; i < n; i++)
  {
    forward();
    delay(50);
  }
}

// 2: LEFT
// 3: RIGHT
// turns around in the left or right position
void turnAround(int where)
{
  repeatnTimes(where, 5);
}

// this plays some melodies (without dancing)
void discoteca()
{
  // if we're demoing we want to dance only once!
  bool hasPlayedOnce = false;

  // some variables used for the melodies
  int note;
  unsigned long discoValue;
  char notesList[] = {'a', 'b', 'c', 'C', 'd', 'e', 'f', 'g', ' '};
  const int numNotes = 9;

  // get ready for the music
  center_servos();
  updateBeatsAndDistance();

  while (true)
  {
    // beats are a function of the distance
    updateBeatsAndDistance();
    note = chooseNote();

    // keep listening for * (returns upon receival of *)
    if (irrecv.decode(&results)) // If we have received an IR signal
    {
      discoValue = results.value;

      if (discoValue == irStar)
      {
        return;
      }
    }

    // if there's and obstacle close enough
    if(play)
    {
      hasPlayedOnce = true;
      TimerFreeTone(buzzerPin, frequency(note, numNotes), beats * tempo);
    }
    else
    {
      if(hasPlayedOnce && demoing)
      {
        return;
      }
    }
    irrecv.resume();
  }
}

// this function is discoteca + dancing (mostly self-explanatory)
void dance()
{
  bool hasDancedOnce = false;
  int note;
  unsigned long danceValue;
  char notesList[] = {'a', 'b', 'c', 'C', 'd', 'e', 'f', 'g', ' '};
  const int numNotes = 9;

  while(true)
  {
    if (irrecv.decode(&results)) // If we have received an IR signal
    {
      danceValue = results.value;

      if (danceValue == irStar)
      {
        return;
      }
    }

    updateBeatsAndDistance();
    note = chooseNote();
    center_servos();

    updateBeatsAndDistance();

    // the delaytime between notes depends on the distance between
    // the ultrasonic sensors and the closest obstacle
    int delayTime = constrain(map(distance, 0, 100, 1, 300), 1, 300);

    note = chooseNote();

      if (demoing && play)
      {
        hasDancedOnce = true;
      }

      while(play)
      {
        updateBeatsAndDistance();
        note = chooseNote();
        TimerFreeTone(buzzerPin, frequency(note, numNotes), beats * tempo);
        irrecv.resume();
        delay(delayTime);
        lean_left();
        updateBeatsAndDistance();
```

```
          note = chooseNote();
          TimerFreeTone(buzzerPin, frequency(note, numNotes), beats * tempo);
          irrecv.resume();
          delay(delayTime);
          lean_right();
        }

        // this led works as an indiciator to help the user send signals through the IR
Remote
        digitalWrite(ledPin, HIGH);
        irrecv.resume(); //next value
        delay(100);
        digitalWrite(ledPin, LOW);

        if (demoing && hasDancedOnce)
        {
          return;
        }
      }
}

// beats depend on the distance from the next obstacle
void updateBeatsAndDistance()
{
    distance = get_distance(CM);
    get_beats();
}

void get_beats()
{
  // beats is a function of the distance
    switch (distance)
    {
      case 0 ... 10:
        beats = 2;
        play = true;
        break;
      case 11 ... 15:
        beats = 3;
        play = true;
        break;
      case 16 ... 25:
        beats = 4;
        play = true;
        break;
      case 26 ... 35:
        beats = 5;
        play = true;
        break;
      case 36 ... 45:
        beats = 6;
        play = true;
      break;
    case 46 ... 55:
      beats = 7;
      play = true;
      break;

    default:
      play = false;
      break;
  }
}

// the note is random
char chooseNote()
{
 char notesList[] = {'a', 'b', 'c', 'C', 'd', 'e', 'f', 'g', ' '};
 const int numNotes = 9;
 return pickRandomNote(notesList, numNotes);
}

char pickRandomNote(char notes[], int numNotes)
{
 return notes[random(0, numNotes)];
}

//==                                                                 Wave
===============================================================
====================
void wave()
{
 /*
 myServo1 - Front Left Pivot Servo
 myServo2 - Front Left Lift Servo
 myServo3 - Back Left Pivot Servo
 myServo4 - Back Left Lift Servo
 myServo5 - Back Right Pivot Servo
 myServo6 - Back Right Lift Servo
 myServo7 - Front Right Pivot Servo
 myServo8 - Front Right Lift Servo
 */

 center_servos();

 // uncomment this block and comment the next one for original waving
 /* myServo4.write(45);
 myServo6.write(45);
 delay(200);
 myServo8.write(0);
 delay(200);
 myServo7.write(180);
 delay(200);
 myServo7.write(30);
```

```
  delay(300);
  myServo7.write(180);
  delay(300);
  myServo7.write(30);
  delay(300);
  myServo7.write(s41);
  delay(300);
  myServo8.write(s42);*/

  myServo4.write(15);
  delay(200);
  myServo6.write(150);
  delay(200);
  myServo2.write(150);
  delay(200);
  myServo8.write(90);
  delay(300);
  myServo7.write(180);
  delay(200);
  myServo7.write(30);
  delay(300);
  myServo7.write(180);
  delay(300);
  myServo7.write(30);
  delay(300);
  myServo7.write(s41);
  delay(300);
  myServo4.write(90);
  delay(200);
  center_servos();
  delay(50);
}

//==                                                              Bow
=================================================================
=====================
void bow()
{
  center_servos();
  delay(200);
  myServo2.write(15);
  myServo8.write(15);
  delay(700);
  myServo2.write(90);
  myServo8.write(90);
  delay(700);
}

void goDown()
{
  center_servos();
  myServo2.write(15);
  myServo4.write(15);
  myServo6.write(15);
  myServo8.write(15);
}

//==                                                        Lean_Left
===============================================================
===============
void lean_left()
{
  myServo2.write(15);
  myServo4.write(15);
  myServo6.write(150);
  myServo8.write(150);
}

//==                                                        Lean_Right
===============================================================
==============
void lean_right()
{
  myServo2.write(150);
  myServo4.write(150);
  myServo6.write(15);
  myServo8.write(15);
}

//==                                                        Lean_Left
===============================================================
===============
void trim_left()
{
  da--; // Left Front Pivot
  db--; // Left Back Pivot
  dc--; // Right Back Pivot
  dd--; // Right Front Pivot
}

//==                                                        Lean_Right
===============================================================
==============
void trim_right()
{
  da++; // Left Front Pivot
  db++; // Left Back Pivot
  dc++; // Right Back Pivot
  dd++; // Right Front Pivot
}

//==                                                        Forward
===============================================================
=================
void forward()
{
  // calculation of points

  // Left Front Pivot
  a90 = (90 + da),
  a120 = (120 + da),
  a150 = (150 + da),
  a180 = (180 + da);

  // Left Back Pivot
  b0 = (0 + db),
  b30 = (30 + db),
  b60 = (60 + db),
  b90 = (90 + db);

  // Right Back Pivot
  c90 = (90 + dc),
  c120 = (120 + dc),
  c150 = (150 + dc),
  c180 = (180 + dc);

  // Right Front Pivot
  d0 = (0 + dd),
  d30 = (30 + dd),
  d60 = (60 + dd),
  d90 = (90 + dd);

  // set servo positions and speeds needed to walk forward one step
  // (LFP,  LBP, RBP,  RFP, LFL, LBL, RBL, RFL, S1, S2, S3, S4)
  srv(a180, b0 , c120, d60, 42,  33,  33,  42,  1,  3,  1,  1);
  srv( a90, b30, c90,  d30, 6,   33,  33,  42,  3,  1,  1,  1);
  srv( a90, b30, c90,  d30, 42,  33,  33,  42,  3,  1,  1,  1);
  srv(a120, b60, c180, d0,  42,  33,  6,   42,  1,  1,  3,  1);
  srv(a120, b60, c180, d0,  42,  33,  33,  42,  1,  1,  3,  1);
  srv(a150, b90, c150, d90, 42,  33,  33,  6,   1,  1,  1,  3);
  srv(a150, b90, c150, d90, 42,  33,  33,  42,  1,  1,  1,  3);
  srv(a180, b0,  c120, d60, 42,  6,   33,  42,  1,  3,  1,  1);
  //srv(a180, b0,  c120, d60, 42,  15,  33,  42,  1,  3,  1,  1);
}

//==                                                                    Back
=================================================================
====================
void back ()
{
  // set servo positions and speeds needed to walk backward one step
  // (LFP,  LBP, RBP,  RFP, LFL, LBL, RBL, RFL, S1, S2, S3, S4)
  srv(180, 0,  120, 60, 42, 33, 33, 42, 3,  1, 1, 1);
  srv(150, 90, 150, 90, 42, 18, 33, 42, 1,  3, 1, 1);
  srv(150, 90, 150, 90, 42, 33, 33, 42, 1,  3, 1, 1);
  srv(120, 60, 180, 0,  42, 33, 33, 6,  1,  1, 1, 3);
  srv(120, 60, 180, 0,  42, 33, 33, 42, 1,  1, 1, 3);
  srv(90,  30, 90,  30, 42, 33, 18, 42, 1,  1, 3, 1);
  srv(90,  30, 90,  30, 42, 33, 33, 42, 1,  1, 3, 1);
  srv(180, 0,  120, 60, 6,  33, 33, 42, 3,  1, 1, 1);
}

//==                                                                    Left
==============================================================
===================
void turn_left ()
{
  // set servo positions and speeds needed to turn left one step
  // (LFP,  LBP, RBP,  RFP, LFL, LBL, RBL, RFL, S1, S2, S3, S4)
  srv(150,  90, 90,  30, 42, 6,  33, 42, 1, 3, 1, 1);
  srv(150,  90, 90,  30, 42, 33, 33, 42, 1, 3, 1, 1);
  srv(120,  60, 180, 0,  42, 33, 6,  42, 1, 1, 3, 1);
  srv(120,  60, 180, 0,  42, 33, 33, 24, 1, 1, 3, 1);
  srv(90,   30, 150, 90, 42, 33, 33, 6,  1, 1, 1, 3);
  srv(90,   30, 150, 90, 42, 33, 33, 42, 1, 1, 1, 3);
  srv(180,  0,  120, 60, 6,  33, 33, 42, 3, 1, 1, 1);
  srv(180,  0,  120, 60, 42, 33, 33, 33, 3, 1, 1, 1);
}

//==                                                                   Right
==============================================================
==================
void turn_right ()
{
  // set servo positions and speeds needed to turn right one step
  // (LFP,  LBP, RBP,  RFP, LFL, LBL, RBL, RFL, S1, S2, S3, S4)
  srv( 90, 30, 150, 90, 6,  33, 33, 42, 3, 1, 1, 1);
  srv( 90, 30, 150, 90, 42, 33, 33, 42, 3, 1, 1, 1);
  srv(120, 60, 180, 0,  42, 33, 33, 6,  1, 1, 1, 3);
  srv(120, 60, 180, 0,  42, 33, 33, 42, 1, 1, 1, 3);
  srv(150, 90, 90,  30, 42, 33, 6,  42, 1, 1, 3, 1);
  srv(150, 90, 90,  30, 42, 33, 33, 42, 1, 1, 3, 1);
  srv(180, 0,  120, 60, 42, 6,  33, 42, 1, 3, 1, 1);
  srv(180, 0,  120, 60, 42, 33, 33, 42, 1, 3, 1, 1);
}

//==                          Center                                  Servos
==============================================================
==========
void center_servos()
{
  myServo1.write(90);
  myServo2.write(90);
  myServo3.write(90);
  myServo4.write(90);
  myServo5.write(90);
  myServo6.write(90);
  myServo7.write(90);
  myServo8.write(90);

  int s11 = 90; // Front Left Pivot Servo
  int s12 = 90; // Front Left Lift Servo
  int s21 = 90; // Back Left Pivot Servo
  int s22 = 90; // Back Left Lift Servo
  int s31 = 90; // Back Right Pivot Servo
  int s32 = 90; // Back Right Lift Servo
  int s41 = 90; // Front Right Pivot Servo
  int s42 = 90; // Front Right Lift Servo
}

//==                              Increase                              Speed
================================================================
==========
void increase_speed()
{
  if (spd > 5)
    spd--;
}

//==                              Decrease                              Speed
================================================================
==========
void decrease_speed()
{
  if (spd < 50)
    spd++;
}

//==                                                                      Srv
================================================================
=====================
void srv( int  p11, int p21, int p31, int p41, int p12, int p22, int p32, int p42, int sp1,
int sp2, int sp3, int sp4)
{
  // p11: Front Left Pivot Servo
  // p21: Back Left Pivot Servo
  // p31: Back Right Pivot Servo
  // p41: Front Right Pivot Servo
  // p12: Front Left Lift Servo
  // p22: Back Left Lift Servo
  // p32: Back Right Lift Servo
  // p42: Front Right Lift Servo
  // sp1: Speed 1
  // sp2: Speed 2
  // sp3: Speed 3
  // sp4: Speed 4
  // Multiply lift servo positions by manual height adjustment
  p12 = p12 + high * 3;
  p22 = p22 + high * 3;
  p32 = p32 + high * 3;
  p42 = p42 + high * 3;

  while ((s11 != p11) || (s21 != p21) || (s31 != p31) || (s41 != p41) || (s12 != p12) ||
(s22 != p22) || (s32 != p32) || (s42 != p42))
  {
    // Front Left Pivot Servo
    if (s11 < p11)           // if servo position is less than programmed position
    {
      if ((s11 + sp1) <= p11)
        s11 = s11 + sp1;      // set servo position equal to servo position plus speed
constant
      else
        s11 = p11;
    }

    if (s11 > p11)           // if servo position is greater than programmed position
    {
      if ((s11 - sp1) >= p11)
        s11 = s11 - sp1;      // set servo position equal to servo position minus speed
constant
      else
        s11 = p11;
    }

    // Back Left Pivot Servo
    if (s21 < p21)
    {
      if ((s21 + sp2) <= p21)
        s21 = s21 + sp2;
      else
        s21 = p21;
    }

    if (s21 > p21)
    {
      if ((s21 - sp2) >= p21)
        s21 = s21 - sp2;
      else
        s21 = p21;
    }

    // Back Right Pivot Servo
    if (s31 < p31)
    {
      if ((s31 + sp3) <= p31)
        s31 = s31 + sp3;
      else
        s31 = p31;
    }

    if (s31 > p31)
    {
```

```
    if ((s31 - sp3) >= p31)
      s31 = s31 - sp3;
    else
      s31 = p31;
  }

  // Front Right Pivot Servo
  if (s41 < p41)
  {
    if ((s41 + sp4) <= p41)
      s41 = s41 + sp4;
    else
      s41 = p41;
  }

  if (s41 > p41)
  {
    if ((s41 - sp4) >= p41)
      s41 = s41 - sp4;
    else
      s41 = p41;
  }

  // Front Left Lift Servo
  if (s12 < p12)
  {
    if ((s12 + sp1) <= p12)
      s12 = s12 + sp1;
    else
      s12 = p12;
  }

  if (s12 > p12)
  {
    if ((s12 - sp1) >= p12)
      s12 = s12 - sp1;
    else
      s12 = p12;
  }

  // Back Left Lift Servo
  if (s22 < p22)
  {
    if ((s22 + sp2) <= p22)
      s22 = s22 + sp2;
    else
      s22 = p22;
  }

  if (s22 > p22)
  {
    if ((s22 - sp2) >= p22)
```

```
      s22 = s22 - sp2;
    else
      s22 = p22;
  }

  // Back Right Lift Servo
  if (s32 < p32)
  {
    if ((s32 + sp3) <= p32)
      s32 = s32 + sp3;
    else
      s32 = p32;
  }

  if (s32 > p32)
  {
    if ((s32 - sp3) >= p32)
      s32 = s32 - sp3;
    else
      s32 = p32;
  }

  // Front Right Lift Servo
  if (s42 < p42)
  {
    if ((s42 + sp4) <= p42)
      s42 = s42 + sp4;
    else
      s42 = p42;
  }

  if (s42 > p42)
  {
    if ((s42 - sp4) >= p42)
      s42 = s42 - sp4;
    else
      s42 = p42;
  }

  // Write Pivot Servo Values
  myServo1.write(s11 + da);
  myServo3.write(s21 + db);
  myServo5.write(s31 + dc);
  myServo7.write(s41 + dd);

  // Write Lift Servos Values
  myServo2.write(s12);
  myServo4.write(s22);
  myServo6.write(s32);
  myServo8.write(s42);
  delay(spd); // Delay before next movement
}
```

```
}

//==                                    USRF                                    Function
=================================================================
==========
long get_distance(bool unit)
{
  // if unit == 0 return inches, else return cm
  long duration = 0,
       cm = 0,
       inches = 0;

  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  cm = (duration / 2) / 29.1;
  inches = (duration / 2) / 74;

  if (unit == INCH)
    return inches;
  else
    return cm;
}

// this function takes a note and assigns to it a frequency
// thanks to the SIK guide for this piece of code
int frequency(char note, int numNotes)
{
  int i;

  // these are the notes
  char notesList[numNotes] = {
    'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};

  // frequency[i] matches names[i]
  int frequencies[numNotes] = {
    262, 294, 330, 349, 392, 440, 494, 523};

  // Now we'll search through the letters in the array, and if
  // we find it, we'll return the frequency for that note.
  for (i = 0; i < numNotes; i++)  // Step through the notes
  {
    if (notesList[i] == note)         // Is this the one?
    {
      return (frequencies[i]);    // Yes! Return the frequency and exit function.
    }
  }
  return 0;  // We looked through everything and didn't find it,
  // but we still need to return a value, so return 0.
}

// given the note return its value (see the discoteca() function)
int noteValue(char note, int numNotes)
{
  if (note == ' ')
  {
    return numNotes + 100;
  }

  char notesList[numNotes] = {
    'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};

  for(int i = 0; i < 8; i++)
  {
    if (notesList[i] == note)
    {
      return i;
    }
    else
    {
      continue;
    }
  }
  return 0;
}
```

The code acts as follows:

1) Pars is listening to the IR Remote Control.
2) As soon as the user sends a signal through the IR Remote Control.
3) Pars will execute said action and then keep listening for more instructions.
4) If a signal from the IR Remote Control is misunderstood, then Pars will execute the most previous understood instruction.
5) During the sequences that use dance() Pars will stop execution when the nearest obstacle does not trigger the dancing behavior and star is pressed (except for the demo() function).

6) The LED indicator helps the user gain an intuitive understanding of when is Pars most listening to the IR Remote Control.

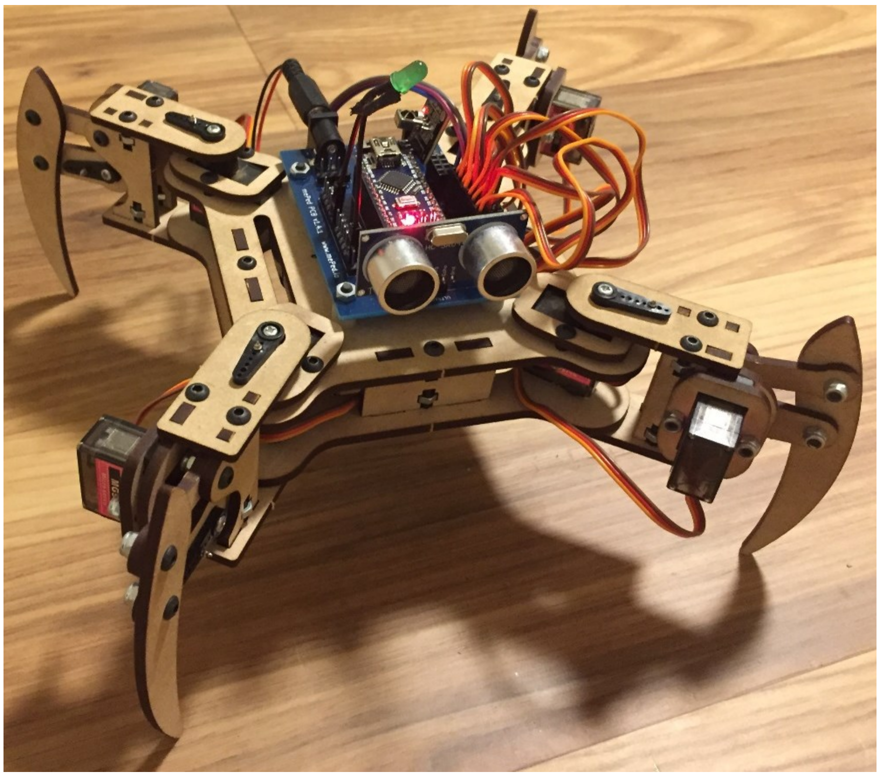


**Figure 8.** Photo of Pars

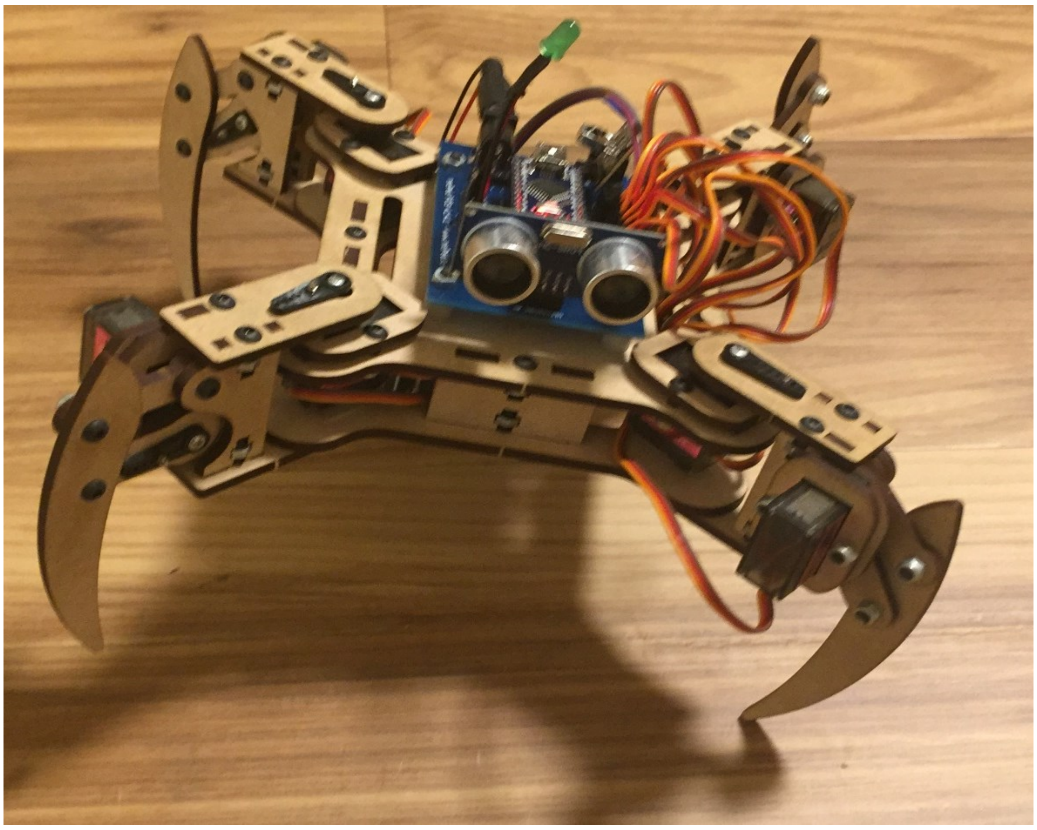
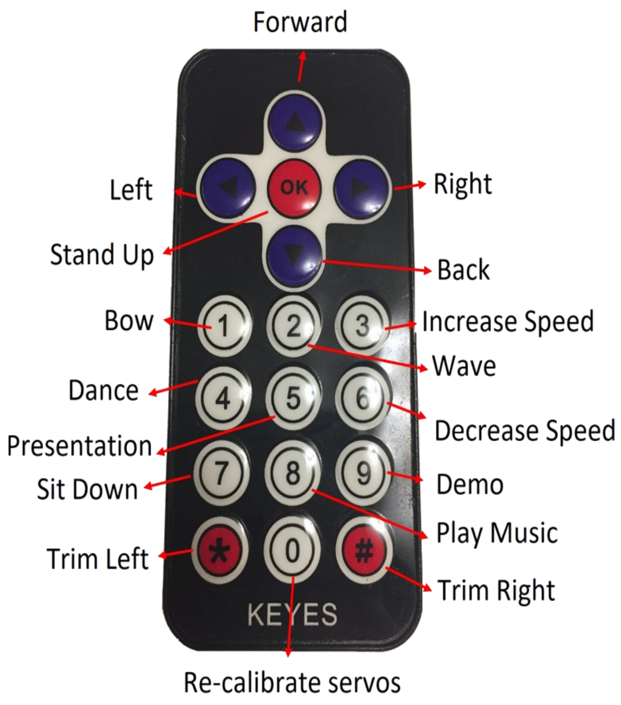


**Figure 9.** Pars about to wave.

## Usage

Press the corresponding instruction on the IR Remote Control while pointing at the IR receiver on Pars.



## Resources

http://www.meped.io/sites/default/files/2016-09/mePed_v2_Assembly_Manual_0.pdf

https://www.sparkfun,com/sikcode

https://www.sparkfun,com/sikguide

https://www.elegoo.com/product/elegoo-37-in-1-sensor-module-kit/

https://bitbucket.org/teckel12/arduino-toneac/wiki/Home

https://bitbucket.org/teckel12/arduino-timer-free-tone/wiki/Home