```
#include <Servo.h>    //to define and control servos
#include "FlexiTimer2.h"//to set a timer to manage all servos
/* Servos ------------------------------------------------------------------
*/
//define 12 servos for 4 legs
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = { {2, 3, 4}, {5, 6, 7}, {8, 9, 10}, {11, 12, 13}
};
/* Size of the robot ------------------------------------------------------
*/
const float length_a = 55;
const float length_b = 77.5;
const float length_c = 27.5;
const float length_side = 71;
const float z_absolute = -28;
/* Constants for movement -------------------------------------------------
*/
const float z_default = -50, z_up = -30, z_boot = z_absolute;
const float x_default = 62, x_offset = 0;
const float y_start = 0, y_step = 40;
const float y_default = x_default;
/* variables for movement -------------------------------------------------
*/
volatile float site_now[4][3];    //real-time coordinates of the end of each
leg
volatile float site_expect[4][3]; //expected coordinates of the end of each
leg
float temp_speed[4][3];   //each axis' speed, needs to be recalculated before
each movement
float move_speed;     //movement speed
float speed_multiple = 1; //movement speed multiple
const float spot_turn_speed = 4;
const float leg_move_speed = 8;
```

```
const float body_move_speed = 3;
const float stand_seat_speed = 1;
volatile int rest_counter;      //+1/0.02s, for automatic rest
//functions' parameter
const float KEEP = 255;
//define PI for calculation
const float pi = 3.1415926;
/* Constants for turn ------------------------------------------------------
*/
//temp length
const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step,
2));
const float temp_b = 2 * (y_start + y_step) + length_side;
const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 *
y_start + y_step + length_side, 2));
const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) - pow(temp_c,
2)) / 2 / temp_a / temp_b);
//site for turn
const float turn_x1 = (temp_a - length_side) / 2;
const float turn_y1 = y_start + y_step / 2;
const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;
/* ------------------------------------------------------------------------
*/


/*
  - setup function
   ------------------------------------------------------------------------
*/
void setup()
{
  //start serial for debug
  Serial.begin(115200);
```

```
  Serial.println("Robot starts initialization");
  //initialize default parameter
  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
  //start servo service
  FlexiTimer2::set(20, servo_service);
  FlexiTimer2::start();
  Serial.println("Servo service started");
  //initialize servos
  servo_attach();
  Serial.println("Servos initialized");
  Serial.println("Robot initialization Complete");
}


void servo_attach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].attach(servo_pin[i][j]);
```

```
      delay(100);
    }
  }
}


void servo_detach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].detach();
      delay(100);
    }
  }
}
/*
  - loop function
   ------------------------------------------------------------------------------
*/
void loop()
{
  Serial.println("Stand");
  stand();
  delay(2000);
  Serial.println("Step forward");
  step_forward(5);
  delay(2000);
  Serial.println("Step back");
  step_back(5);
  delay(2000);
```

```
  Serial.println("Turn left");
  turn_left(5);
  delay(2000);
  Serial.println("Turn right");
  turn_right(5);
  delay(2000);
  Serial.println("Hand wave");
  hand_wave(3);
  delay(2000);
  Serial.println("Hand wave");
  hand_shake(3);
  delay(2000);
  Serial.println("Body dance");
  body_dance(10);
  delay(2000);
  Serial.println("Sit");
  sit();
  delay(5000);
}


/*
  - sit
  - blocking function
   ------------------------------------------------------------------------
*/
void sit(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_boot);
```

```
  }
  wait_all_reach();
}


/*
  - stand
  - blocking function
   ------------------------------------------------------------------------
*/
void stand(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_default);
  }
  wait_all_reach();
}



/*
  - spot turn to left
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------
*/
void turn_left(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
```

```
if (site_now[3][1] == y_start)
{
  //leg 3&1 move
  set_site(3, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x1 - x_offset, turn_y1, z_default);
  set_site(1, turn_x0 - x_offset, turn_y0, z_default);
  set_site(2, turn_x1 + x_offset, turn_y1, z_default);
  set_site(3, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(3, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x1 + x_offset, turn_y1, z_default);
  set_site(1, turn_x0 + x_offset, turn_y0, z_default);
  set_site(2, turn_x1 - x_offset, turn_y1, z_default);
  set_site(3, turn_x0 - x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(1, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_default);
  set_site(1, x_default + x_offset, y_start, z_up);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
```

```
  wait_all_reach();

  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 0&2 move
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_up);
  set_site(1, turn_x1 + x_offset, turn_y1, z_default);
  set_site(2, turn_x0 - x_offset, turn_y0, z_default);
  set_site(3, turn_x1 - x_offset, turn_y1, z_default);
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x0 - x_offset, turn_y0, z_default);
  set_site(1, turn_x1 - x_offset, turn_y1, z_default);
  set_site(2, turn_x0 + x_offset, turn_y0, z_default);
  set_site(3, turn_x1 + x_offset, turn_y1, z_default);
  wait_all_reach();

  set_site(2, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();
```

```
      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_up);
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();

      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - spot turn to right
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------
*/
void turn_right(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[2][1] == y_start)
    {
      //leg 2&0 move
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
```

```
set_site(0, turn_x0 - x_offset, turn_y0, z_default);
set_site(1, turn_x1 - x_offset, turn_y1, z_default);
set_site(2, turn_x0 + x_offset, turn_y0, z_up);
set_site(3, turn_x1 + x_offset, turn_y1, z_default);
wait_all_reach();

set_site(2, turn_x0 + x_offset, turn_y0, z_default);
wait_all_reach();

set_site(0, turn_x0 + x_offset, turn_y0, z_default);
set_site(1, turn_x1 + x_offset, turn_y1, z_default);
set_site(2, turn_x0 - x_offset, turn_y0, z_default);
set_site(3, turn_x1 - x_offset, turn_y1, z_default);
wait_all_reach();

set_site(0, turn_x0 + x_offset, turn_y0, z_up);
wait_all_reach();

set_site(0, x_default + x_offset, y_start, z_up);
set_site(1, x_default + x_offset, y_start, z_default);
set_site(2, x_default - x_offset, y_start + y_step, z_default);
set_site(3, x_default - x_offset, y_start + y_step, z_default);
wait_all_reach();

set_site(0, x_default + x_offset, y_start, z_default);
wait_all_reach();
```

```
}
else
{
  //leg 1&3 move
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x1 + x_offset, turn_y1, z_default);
  set_site(1, turn_x0 + x_offset, turn_y0, z_up);
  set_site(2, turn_x1 - x_offset, turn_y1, z_default);
  set_site(3, turn_x0 - x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(1, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x1 - x_offset, turn_y1, z_default);
  set_site(1, turn_x0 - x_offset, turn_y0, z_default);
  set_site(2, turn_x1 + x_offset, turn_y1, z_default);
  set_site(3, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(3, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default - x_offset, y_start + y_step, z_default);
  set_site(1, x_default - x_offset, y_start + y_step, z_default);
  set_site(2, x_default + x_offset, y_start, z_default);
```

```
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();


      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}


/*
  - go forward
  - blocking function
  - parameter step steps wanted to go
   ---------------------------------------------------------------------------
*/
void step_forward(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
  {
    if (site_now[2][1] == y_start)
    {
      //leg 2&1 move
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();
```

```
  move_speed = body_move_speed;

  set_site(0, x_default + x_offset, y_start, z_default);
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;

  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 0&3 move
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();

  move_speed = body_move_speed;
```

```
      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_default);
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = leg_move_speed;

      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - go back
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------
*/
void step_back(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
```

```
{
  if (site_now[3][1] == y_start)
  {
    //leg 3&0 move
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = body_move_speed;

    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
    set_site(1, x_default + x_offset, y_start, z_default);
    set_site(2, x_default - x_offset, y_start + y_step, z_default);
    set_site(3, x_default - x_offset, y_start + y_step, z_default);
    wait_all_reach();

    move_speed = leg_move_speed;

    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start, z_default);
    wait_all_reach();
  }
  else
```

```
    {
      //leg 1&2 move
      set_site(1, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = body_move_speed;

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();

      move_speed = leg_move_speed;

      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}
```

```
// add by RegisHsu


void body_left(int i)
{
  set_site(0, site_now[0][0] + i, KEEP, KEEP);
  set_site(1, site_now[1][0] + i, KEEP, KEEP);
  set_site(2, site_now[2][0] - i, KEEP, KEEP);
  set_site(3, site_now[3][0] - i, KEEP, KEEP);
  wait_all_reach();
}


void body_right(int i)
{
  set_site(0, site_now[0][0] - i, KEEP, KEEP);
  set_site(1, site_now[1][0] - i, KEEP, KEEP);
  set_site(2, site_now[2][0] + i, KEEP, KEEP);
  set_site(3, site_now[3][0] + i, KEEP, KEEP);
  wait_all_reach();
}


void hand_wave(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
```

```
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(2, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(0, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
```

```
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}


void hand_shake(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(2, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
```

```
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(0, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}


void head_up(int i)
{
  set_site(0, KEEP, KEEP, site_now[0][2] - i);
  set_site(1, KEEP, KEEP, site_now[1][2] + i);
  set_site(2, KEEP, KEEP, site_now[2][2] - i);
  set_site(3, KEEP, KEEP, site_now[3][2] + i);
  wait_all_reach();
}
```

```
void head_down(int i)
{
  set_site(0, KEEP, KEEP, site_now[0][2] + i);
  set_site(1, KEEP, KEEP, site_now[1][2] - i);
  set_site(2, KEEP, KEEP, site_now[2][2] + i);
  set_site(3, KEEP, KEEP, site_now[3][2] - i);
  wait_all_reach();
}


void body_dance(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  float body_dance_speed = 2;
  sit();
  move_speed = 1;
  set_site(0, x_default, y_default, KEEP);
  set_site(1, x_default, y_default, KEEP);
  set_site(2, x_default, y_default, KEEP);
  set_site(3, x_default, y_default, KEEP);
  wait_all_reach();
  //stand();
  set_site(0, x_default, y_default, z_default - 20);
  set_site(1, x_default, y_default, z_default - 20);
  set_site(2, x_default, y_default, z_default - 20);
  set_site(3, x_default, y_default, z_default - 20);
  wait_all_reach();
  move_speed = body_dance_speed;
```

```
  head_up(30);
  for (int j = 0; j < i; j++)
  {
    if (j > i / 4)
      move_speed = body_dance_speed * 2;
    if (j > i / 2)
      move_speed = body_dance_speed * 3;
    set_site(0, KEEP, y_default - 20, KEEP);
    set_site(1, KEEP, y_default + 20, KEEP);
    set_site(2, KEEP, y_default - 20, KEEP);
    set_site(3, KEEP, y_default + 20, KEEP);
    wait_all_reach();
    set_site(0, KEEP, y_default + 20, KEEP);
    set_site(1, KEEP, y_default - 20, KEEP);
    set_site(2, KEEP, y_default + 20, KEEP);
    set_site(3, KEEP, y_default - 20, KEEP);
    wait_all_reach();
  }
  move_speed = body_dance_speed;
  head_down(30);
}



/*
  - microservos service /timer interrupt function/50Hz
  - when set site expected,this function move the end point to it in a
straight line
  - temp_speed[4][3] should be set before set expect site,it make sure the end
point
   move in a straight line,and decide move speed.
   ----------------------------------------------------------------------------
*/
```

```
void servo_service(void)
{
  sei();
  static float alpha, beta, gamma;


  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      if (abs(site_now[i][j] - site_expect[i][j]) >= abs(temp_speed[i][j]))
        site_now[i][j] += temp_speed[i][j];
      else
        site_now[i][j] = site_expect[i][j];
    }


    cartesian_to_polar(alpha, beta, gamma, site_now[i][0], site_now[i][1],
site_now[i][2]);
    polar_to_servo(i, alpha, beta, gamma);
  }


  rest_counter++;
}


/*
  - set one of end points' expect site
  - this founction will set temp_speed[4][3] at same time
  - non - blocking function
   ---------------------------------------------------------------------------
*/
```

```
void set_site(int leg, float x, float y, float z)
{
  float length_x = 0, length_y = 0, length_z = 0;


  if (x != KEEP)
    length_x = x - site_now[leg][0];
  if (y != KEEP)
    length_y = y - site_now[leg][1];
  if (z != KEEP)
    length_z = z - site_now[leg][2];


  float length = sqrt(pow(length_x, 2) + pow(length_y, 2) + pow(length_z, 2));


  temp_speed[leg][0] = length_x / length * move_speed * speed_multiple;
  temp_speed[leg][1] = length_y / length * move_speed * speed_multiple;
  temp_speed[leg][2] = length_z / length * move_speed * speed_multiple;


  if (x != KEEP)
    site_expect[leg][0] = x;
  if (y != KEEP)
    site_expect[leg][1] = y;
  if (z != KEEP)
    site_expect[leg][2] = z;
}


/*
  - wait one of end points move to expect site
  - blocking function
```

```
   ------------------------------------------------------------------------
*/
void wait_reach(int leg)
{
  while (1)
    if (site_now[leg][0] == site_expect[leg][0])
      if (site_now[leg][1] == site_expect[leg][1])
        if (site_now[leg][2] == site_expect[leg][2])
          break;
}



/*
  - wait all of end points move to expect site
  - blocking function
   ------------------------------------------------------------------------
*/
void wait_all_reach(void)
{
  for (int i = 0; i < 4; i++)
    wait_reach(i);
}



/*
  - trans site from cartesian to polar
  - mathematical model 2/2
   ------------------------------------------------------------------------
*/
void cartesian_to_polar(volatile float &alpha, volatile float &beta, volatile
float &gamma, volatile float x, volatile float y, volatile float z)
{
  //calculate w-z degree
```

```
  float v, w;
  w = (x >= 0 ? 1 : -1) * (sqrt(pow(x, 2) + pow(y, 2)));
  v = w - length_c;
  alpha = atan2(z, v) + acos((pow(length_a, 2) - pow(length_b, 2) + pow(v, 2)
+ pow(z, 2)) / 2 / length_a / sqrt(pow(v, 2) + pow(z, 2)));
  beta = acos((pow(length_a, 2) + pow(length_b, 2) - pow(v, 2) - pow(z, 2)) /
2 / length_a / length_b);
  //calculate x-y-z degree
  gamma = (w >= 0) ? atan2(y, x) : atan2(-y, -x);
  //trans degree pi->180
  alpha = alpha / pi * 180;
  beta = beta / pi * 180;
  gamma = gamma / pi * 180;
}


/*
  - trans site from polar to microservos
  - mathematical model map to fact
  - the errors saved in eeprom will be add
   ----------------------------------------------------------------------------
*/
void polar_to_servo(int leg, float alpha, float beta, float gamma)
{
  if (leg == 0)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  else if (leg == 1)
  {
    alpha += 90;
```

```
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 2)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 3)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  servo[leg][0].write(alpha);
  servo[leg][1].write(beta);
  servo[leg][2].write(gamma);
}
```