Final Project
Remote Controlled BRAT Biped

Amartya Barua

Jose Antonio De Garcia Gomez

NYU POLYTECHNIC SCHOOL OF ENGINEERING

Features

- A biped controlled over a network using a smartphone or computer
- Live feed directly from a camera mounted on the chassis
- Take advantage of multiple processors running simultaneously
- Control over wifi

Components

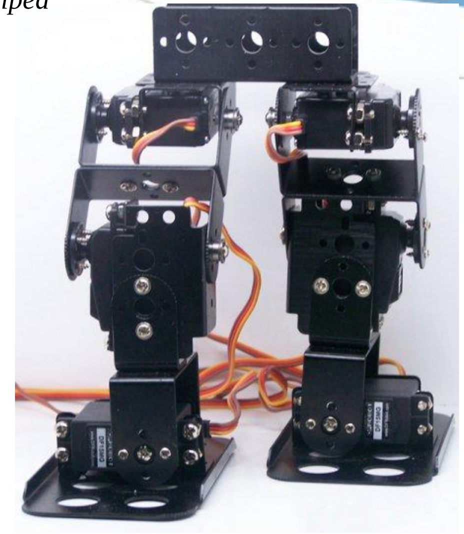
- Raspberry Pi
- Pi Camera Board
- Parallax Propeller
- 6 Servos
- Arduino
- Wifly Shield
- USB wifi adapter
- Battery packs

Design Overview

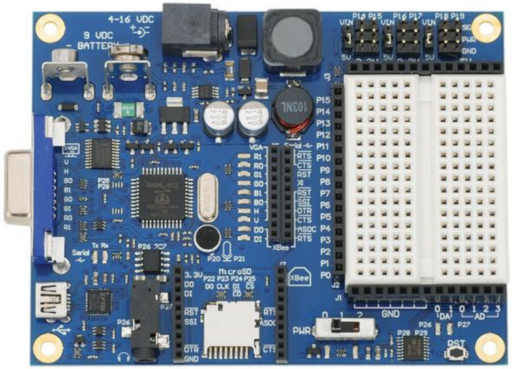
- The system consists of the following subsystems:
  - Live feed – uses a raspberry pi to stream footage from raspberry pi camera board
  - Biped – uses parallax propeller to move the actuators based on the user input
  - TCP communication – uses arduino + wifly shield to communicate with a smartphone or computer

How Does It Work

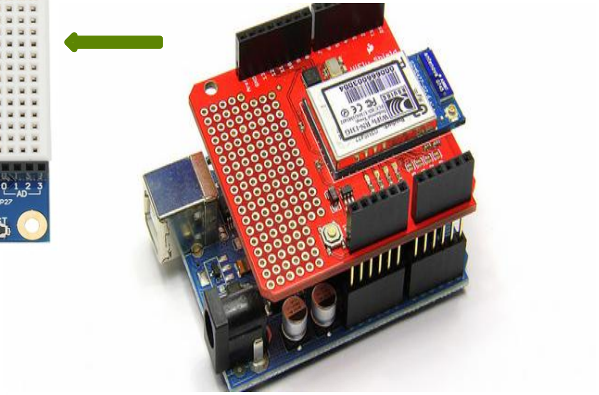
BRAT Biped Chasis



Code (SIMPLE IDE)

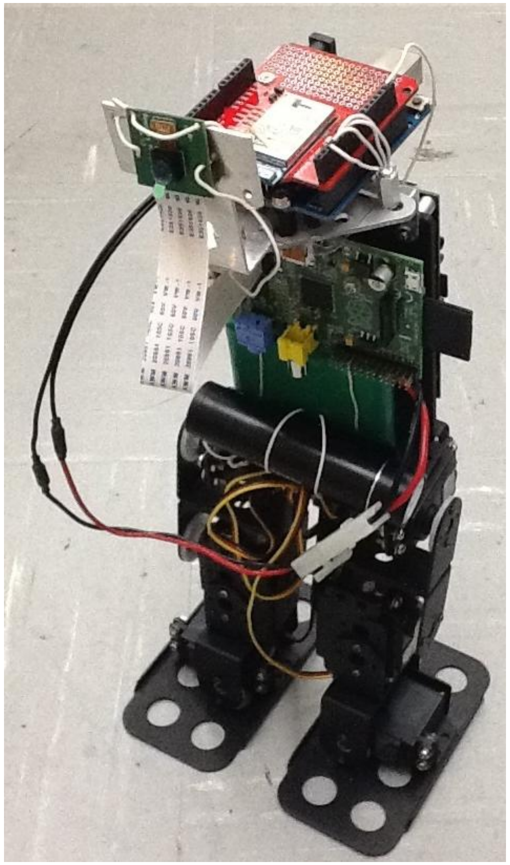


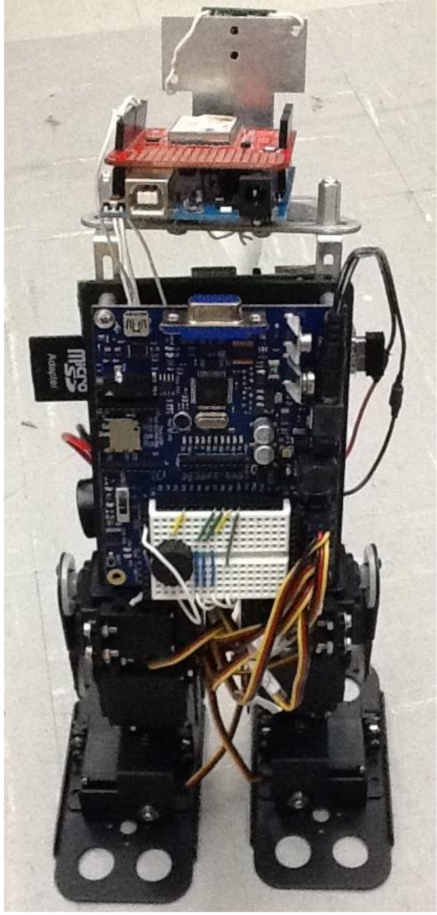
Arduino + wifly



6x HSS - 422 Servo

Mechanical Design

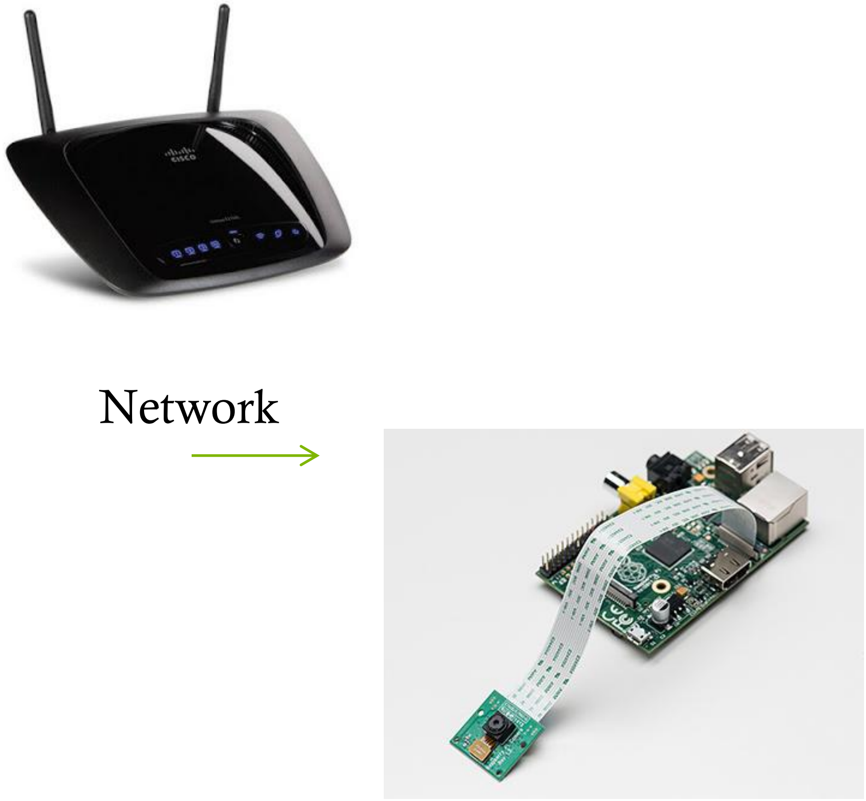




NYU POLYTECHNIC SCHOOL OF ENGINEERING

Raspberry Pi

- Uses pi camera board to capture video
- Stream the video using gstreamer1.0 over port 5000
- The commands are collated into a single script file and run continuously after the initial bootup



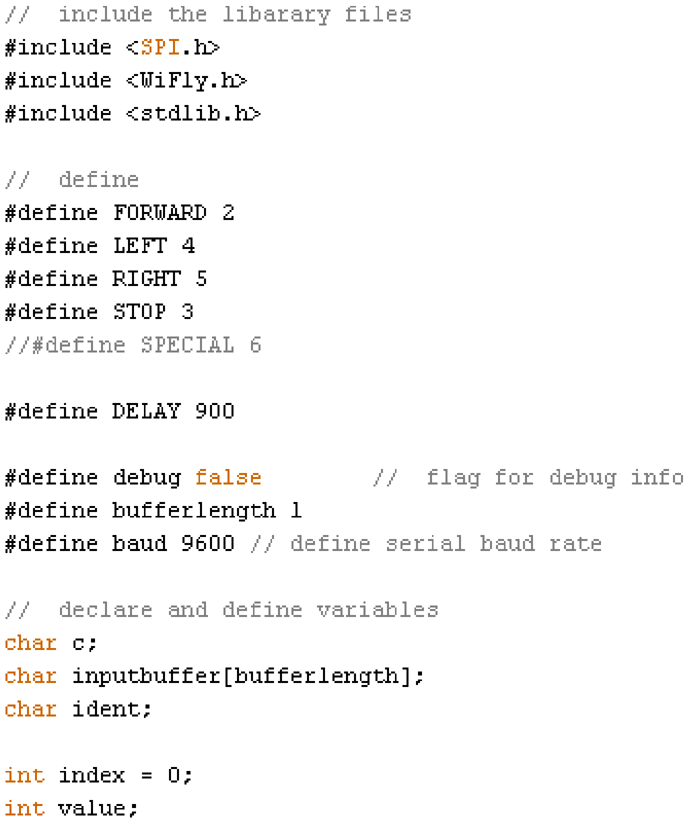
Shell Script

- set +H – turns off the hash function; limits the risk of using old programs
- raspivid -t 999999 -h 720 -w 1080 -fps 25 -hf -b 2000000 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=1 pt=96 ! gdppay ! tcpserversink host=IP-ADDRESS port=5000
- Save it as a script file, make it executable in /etc/init.d/ and run at bootup

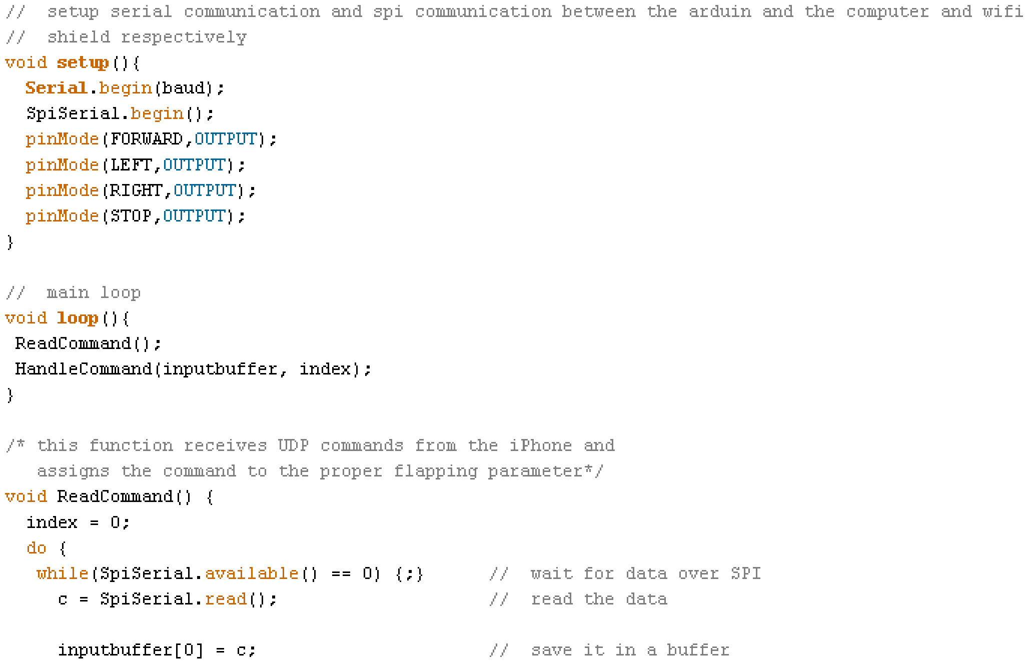
Arduino + Wifly

- The wifly shield communicate to the arduino over the SPI
- Only the pins dedicated to SPI are used to all other I/O pins are available to be used
- Configure the shield to join an existing network or create an ad hoc network
  - Here we connect to an existing network where the computer/smartphone is already connected
  - Provide static ip address to the shield, the network name, password, authentication protocol etc.

Code Highlight

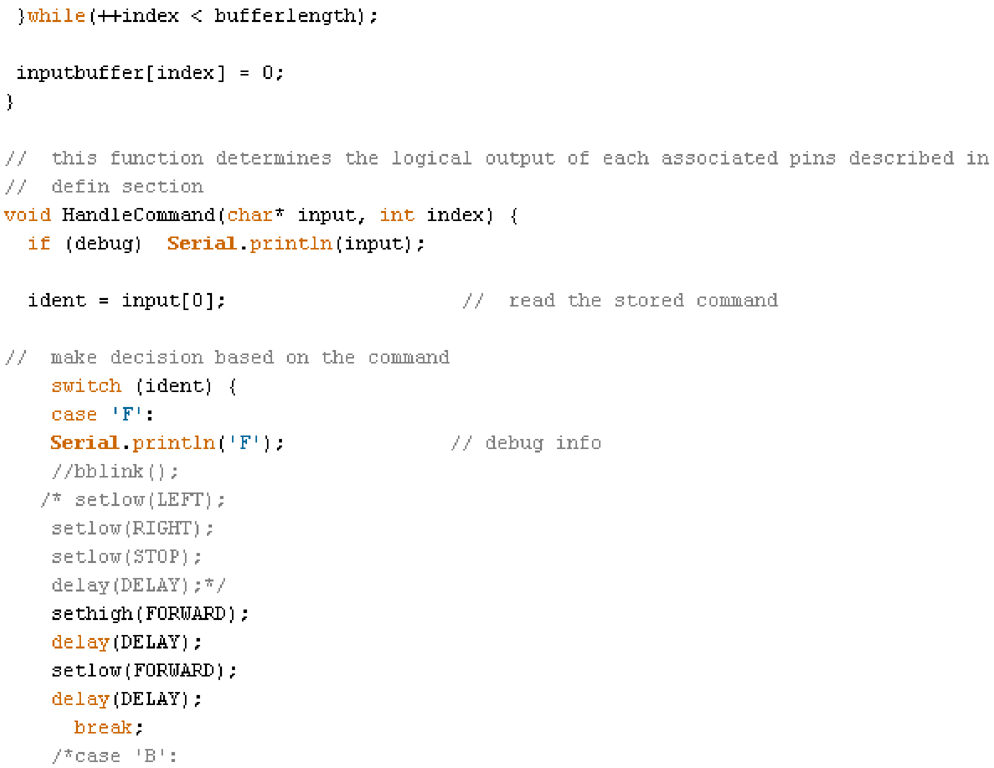


```
//  include the libarary files
#include <SPI.h>
#include <WiFly.h>
#include <stdlib.h>

//  define
#define FORWARD 2
#define LEFT 4
#define RIGHT 5
#define STOP 3
//#define SPECIAL 6

#define DELAY 900

#define debug false         //  flag for debug info
#define bufferlength 1
#define baud 9600 // define serial baud rate

//  declare and define variables
char c;
char inputbuffer[bufferlength];
char ident;

int index = 0;
int value;
```

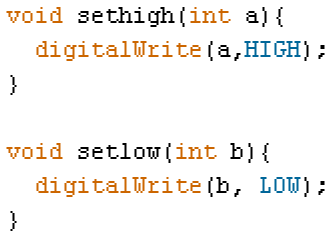


```
//  setup serial communication and spi communication between the arduin and the computer and wifi
//  shield respectively
void setup(){
  Serial.begin(baud);
  SpiSerial.begin();
  pinMode(FORWARD,OUTPUT);
  pinMode(LEFT,OUTPUT);
  pinMode(RIGHT,OUTPUT);
  pinMode(STOP,OUTPUT);
}

//  main loop
void loop(){
 ReadCommand();
 HandleCommand(inputbuffer, index);
}

/* this function receives UDP commands from the iPhone and
   assigns the command to the proper flapping parameter*/
void ReadCommand() {
  index = 0;
  do {
   while(SpiSerial.available() == 0) {;}      //  wait for data over SPI
     c = SpiSerial.read();                    //  read the data

     inputbuffer[0] = c;                      //  save it in a buffer
```

Code Highlight



```
 }while(++index < bufferlength);

 inputbuffer[index] = 0;
}

//  this function determines the logical output of each associated pins described in
//  defin section
void HandleCommand(char* input, int index) {
  if (debug)  Serial.println(input);

  ident = input[0];                    //  read the stored command

//  make decision based on the command
    switch (ident) {
    case 'F':
    Serial.println('F');               // debug info
    //bblink();
   /* setlow(LEFT);
    setlow(RIGHT);
    setlow(STOP);
    delay(DELAY);*/
    sethigh(FORWARD);
    delay(DELAY);
    setlow(FORWARD);
    delay(DELAY);
      break;
    /*case 'B':
```

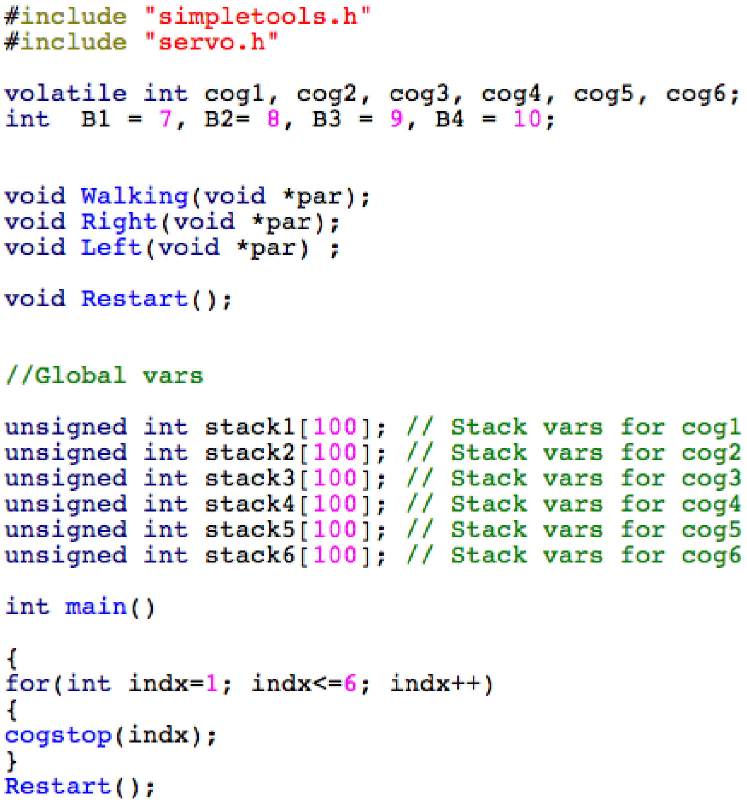


```
void sethigh(int a){
  digitalWrite(a,HIGH);
}

void setlow(int b){
  digitalWrite(b, LOW);
}
```

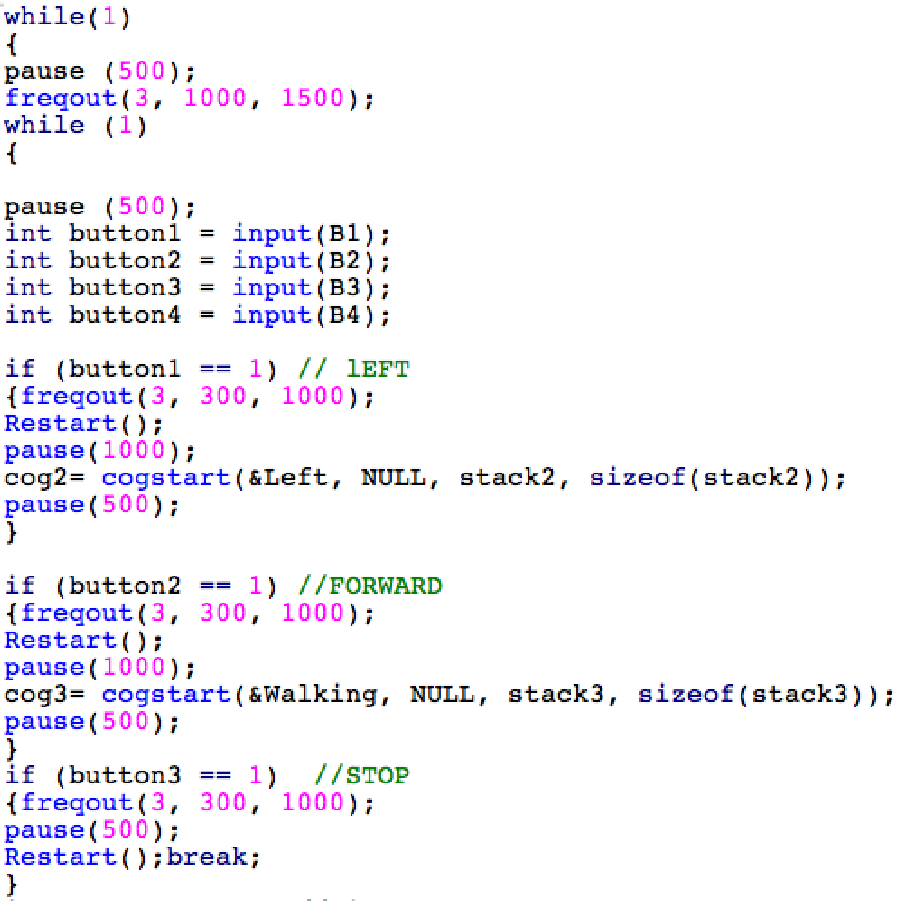
Propeller

- Using the different cogs that Propeller offers us we can run more that one action at the same time i.e. read user commands and move actuators in parallel
- It continuously receives signal form the arduino + wifly
- Propeller interprets and process the signal. Depending on this, the robot will walk in different directions

Code Main Cog



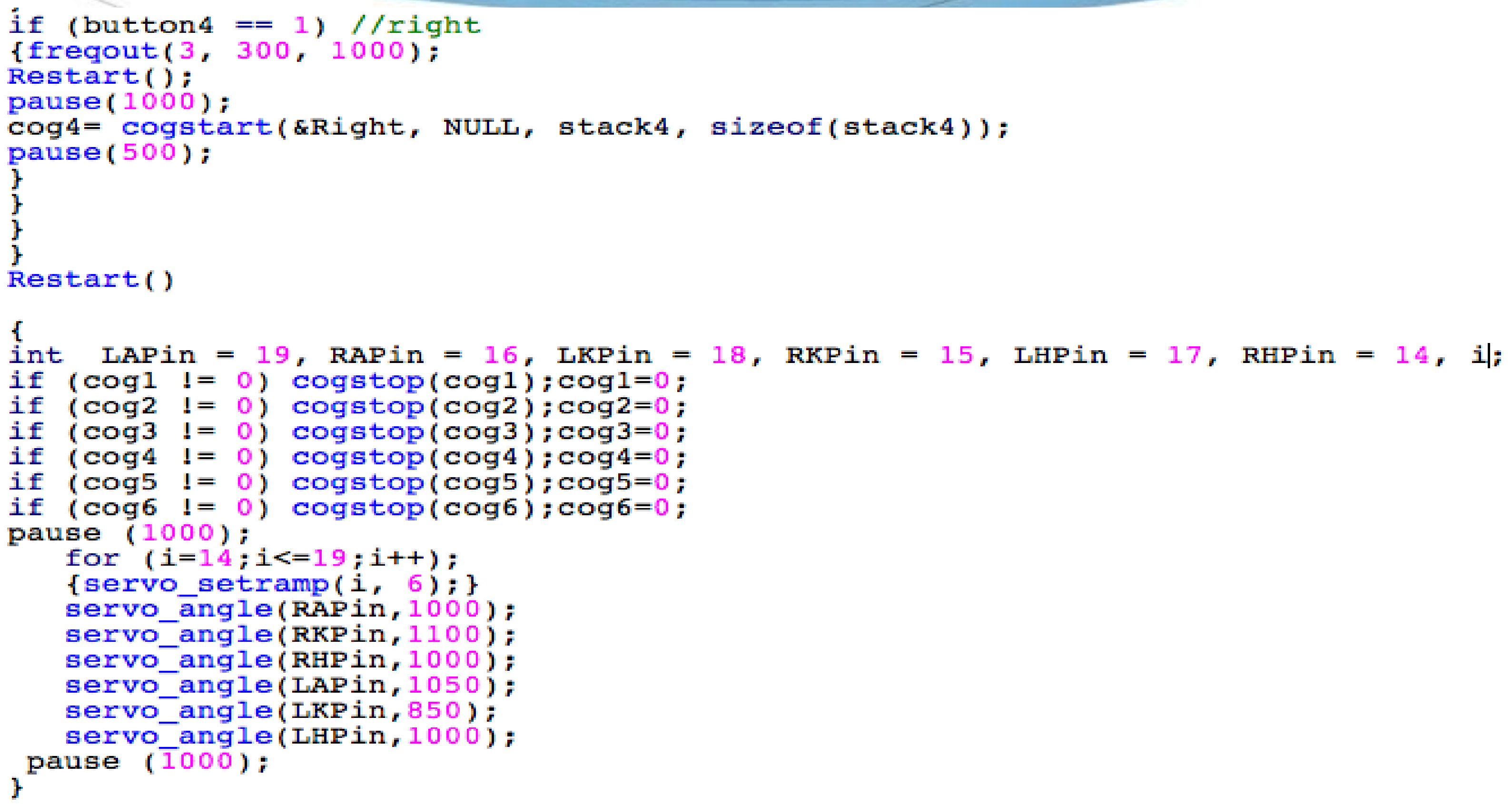
```
#include "simpletools.h"
#include "servo.h"

volatile int cog1, cog2, cog3, cog4, cog5, cog6;
int  B1 = 7, B2= 8, B3 = 9, B4 = 10;


void Walking(void *par);
void Right(void *par);
void Left(void *par) ;

void Restart();


//Global vars

unsigned int stack1[100]; // Stack vars for cog1
unsigned int stack2[100]; // Stack vars for cog2
unsigned int stack3[100]; // Stack vars for cog3
unsigned int stack4[100]; // Stack vars for cog4
unsigned int stack5[100]; // Stack vars for cog5
unsigned int stack6[100]; // Stack vars for cog6

int main()

{
for(int indx=1; indx<=6; indx++)
{
cogstop(indx);
}
Restart();
```



```
while(1)
{
pause (500);
freqout(3, 1000, 1500);
while (1)
{

pause (500);
int button1 = input(B1);
int button2 = input(B2);
int button3 = input(B3);
int button4 = input(B4);

if (button1 == 1) // lEFT
{freqout(3, 300, 1000);
Restart();
pause(1000);
cog2= cogstart(&Left, NULL, stack2, sizeof(stack2));
pause(500);
}

if (button2 == 1) //FORWARD
{freqout(3, 300, 1000);
Restart();
pause(1000);
cog3= cogstart(&Walking, NULL, stack3, sizeof(stack3));
pause(500);
}
if (button3 == 1)  //STOP
{freqout(3, 300, 1000);
pause(500);
Restart();break;
}
```

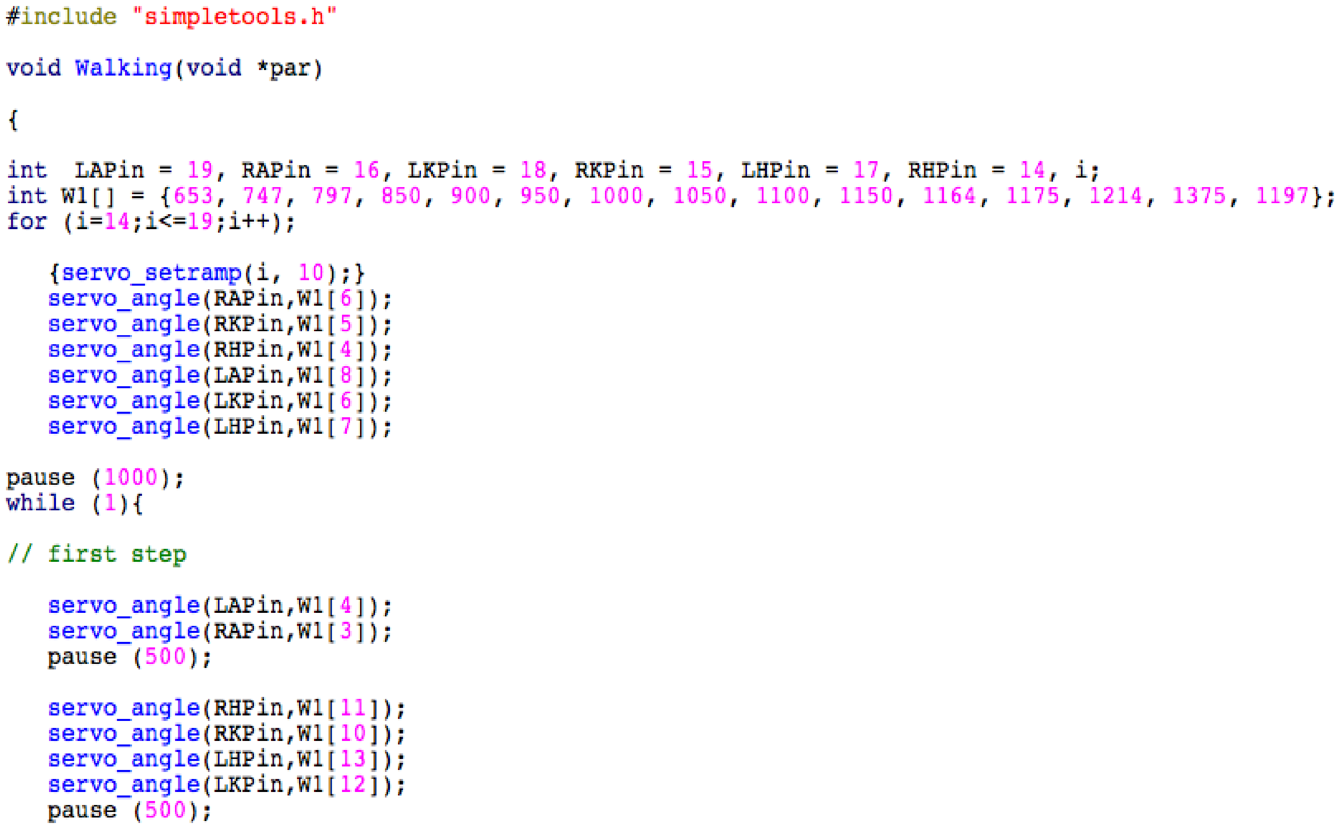
NYU POLYTECHNIC SCHOOL OF ENGINEERING

Code Main Cog



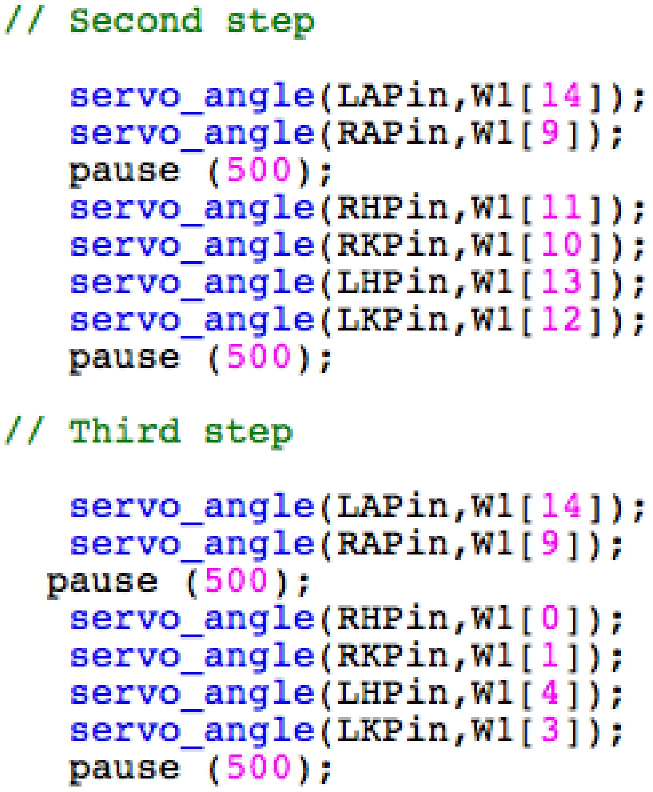
```
if (button4 == 1) //right
{freqout(3, 300, 1000);
Restart();
pause(1000);
cog4= cogstart(&Right, NULL, stack4, sizeof(stack4));
pause(500);
}
}
}
}
Restart()

{
int  LAPin = 19, RAPin = 16, LKPin = 18, RKPin = 15, LHPin = 17, RHPin = 14, i;
if (cog1 != 0) cogstop(cog1);cog1=0;
if (cog2 != 0) cogstop(cog2);cog2=0;
if (cog3 != 0) cogstop(cog3);cog3=0;
if (cog4 != 0) cogstop(cog4);cog4=0;
if (cog5 != 0) cogstop(cog5);cog5=0;
if (cog6 != 0) cogstop(cog6);cog6=0;
pause (1000);
   for (i=14;i<=19;i++);
   {servo_setramp(i, 6);}
   servo_angle(RAPin,1000);
   servo_angle(RKPin,1100);
   servo_angle(RHPin,1000);
   servo_angle(LAPin,1050);
   servo_angle(LKPin,850);
   servo_angle(LHPin,1000);
 pause (1000);
}
```

Code Walking Tab

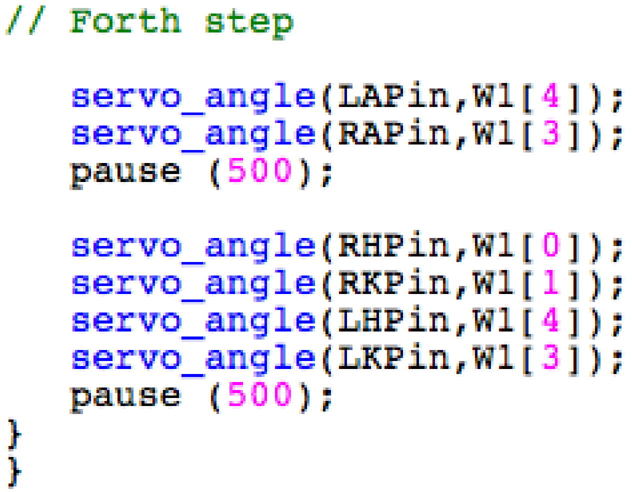


```
#include "simpletools.h"

void Walking(void *par)

{

int  LAPin = 19, RAPin = 16, LKPin = 18, RKPin = 15, LHPin = 17, RHPin = 14, i;
int Wl[] = {653, 747, 797, 850, 900, 950, 1000, 1050, 1100, 1150, 1164, 1175, 1214, 1375, 1197};
for (i=14;i<=19;i++);

   {servo_setramp(i, 10);}
   servo_angle(RAPin,Wl[6]);
   servo_angle(RKPin,Wl[5]);
   servo_angle(RHPin,Wl[4]);
   servo_angle(LAPin,Wl[8]);
   servo_angle(LKPin,Wl[6]);
   servo_angle(LHPin,Wl[7]);

pause (1000);
while (1){

// first step

   servo_angle(LAPin,Wl[4]);
   servo_angle(RAPin,Wl[3]);
   pause (500);

   servo_angle(RHPin,Wl[11]);
   servo_angle(RKPin,Wl[10]);
   servo_angle(LHPin,Wl[13]);
   servo_angle(LKPin,Wl[12]);
   pause (500);
```

Code Walking Tab

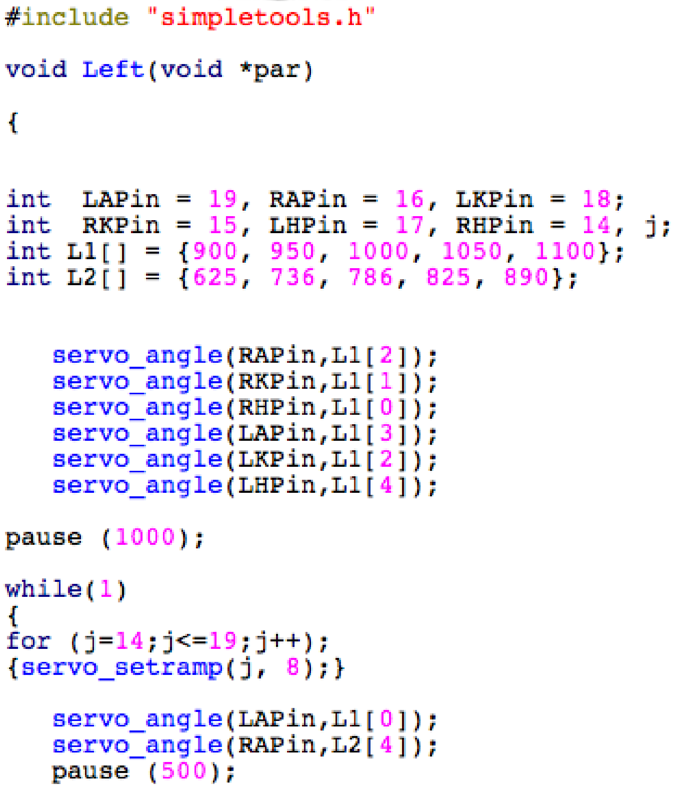


```
// Second step

   servo_angle(LAPin,W1[14]);
   servo_angle(RAPin,W1[9]);
   pause (500);
   servo_angle(RHPin,W1[11]);
   servo_angle(RKPin,W1[10]);
   servo_angle(LHPin,W1[13]);
   servo_angle(LKPin,W1[12]);
   pause (500);

// Third step

   servo_angle(LAPin,W1[14]);
   servo_angle(RAPin,W1[9]);
  pause (500);
   servo_angle(RHPin,W1[0]);
   servo_angle(RKPin,W1[1]);
   servo_angle(LHPin,W1[4]);
   servo_angle(LKPin,W1[3]);
   pause (500);
```

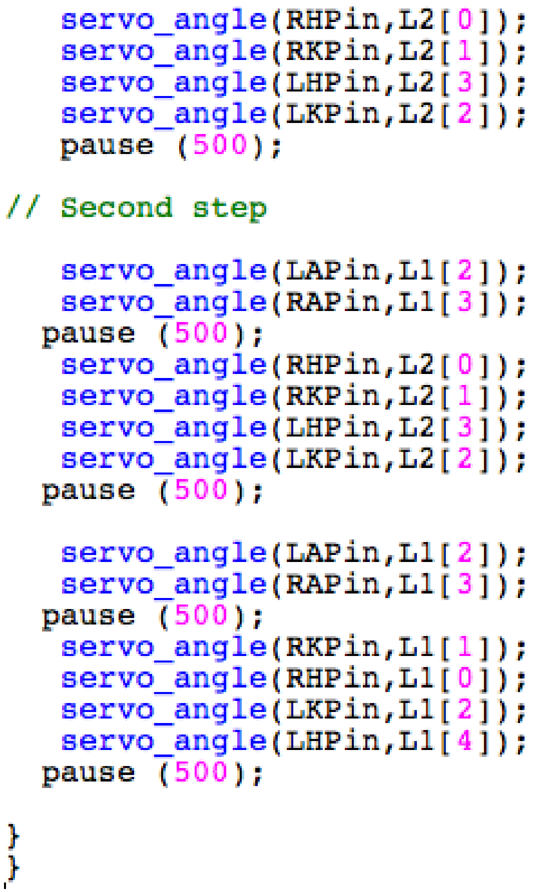


```
// Forth step

   servo_angle(LAPin,W1[4]);
   servo_angle(RAPin,W1[3]);
   pause (500);

   servo_angle(RHPin,W1[0]);
   servo_angle(RKPin,W1[1]);
   servo_angle(LHPin,W1[4]);
   servo_angle(LKPin,W1[3]);
   pause (500);
}
}
```

Code Left Turn

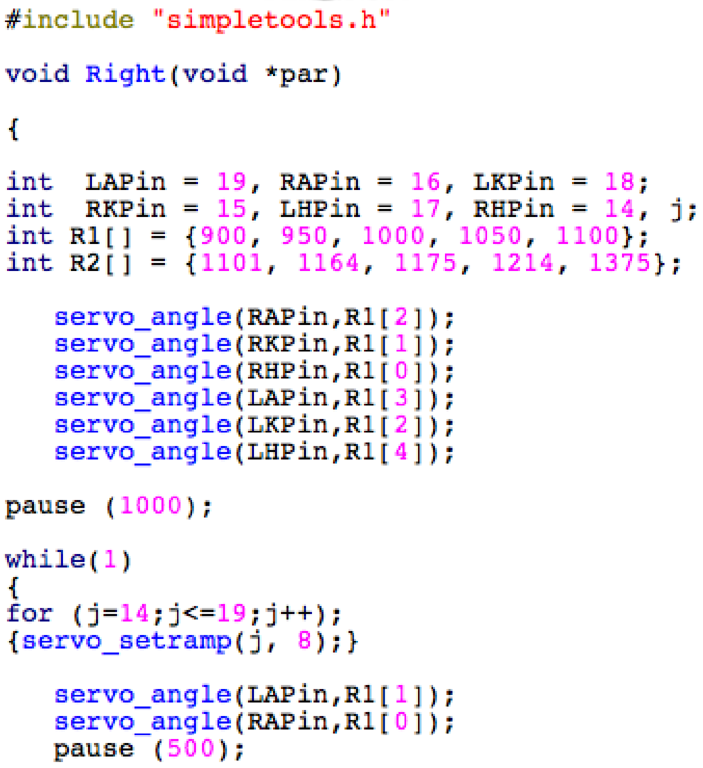


```
#include "simpletools.h"

void Left(void *par)

{


int  LAPin = 19, RAPin = 16, LKPin = 18;
int  RKPin = 15, LHPin = 17, RHPin = 14, j;
int L1[] = {900, 950, 1000, 1050, 1100};
int L2[] = {625, 736, 786, 825, 890};


   servo_angle(RAPin,L1[2]);
   servo_angle(RKPin,L1[1]);
   servo_angle(RHPin,L1[0]);
   servo_angle(LAPin,L1[3]);
   servo_angle(LKPin,L1[2]);
   servo_angle(LHPin,L1[4]);

pause (1000);

while(1)
{
for (j=14;j<=19;j++);
{servo_setramp(j, 8);}

   servo_angle(LAPin,L1[0]);
   servo_angle(RAPin,L2[4]);
   pause (500);
```

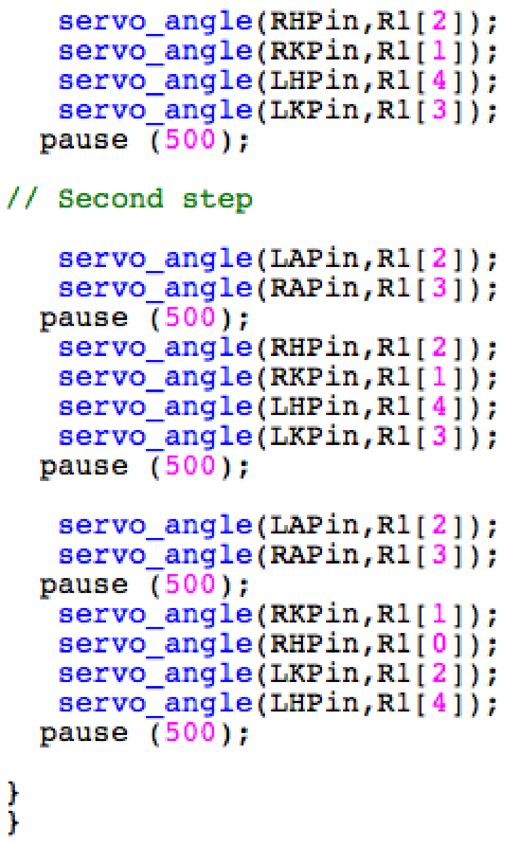


```
    servo_angle(RHPin,L2[0]);
    servo_angle(RKPin,L2[1]);
    servo_angle(LHPin,L2[3]);
    servo_angle(LKPin,L2[2]);
    pause (500);

// Second step

    servo_angle(LAPin,L1[2]);
    servo_angle(RAPin,L1[3]);
   pause (500);
    servo_angle(RHPin,L2[0]);
    servo_angle(RKPin,L2[1]);
    servo_angle(LHPin,L2[3]);
    servo_angle(LKPin,L2[2]);
   pause (500);

    servo_angle(LAPin,L1[2]);
    servo_angle(RAPin,L1[3]);
   pause (500);
    servo_angle(RKPin,L1[1]);
    servo_angle(RHPin,L1[0]);
    servo_angle(LKPin,L1[2]);
    servo_angle(LHPin,L1[4]);
   pause (500);


}
}
```

Code Right Turn

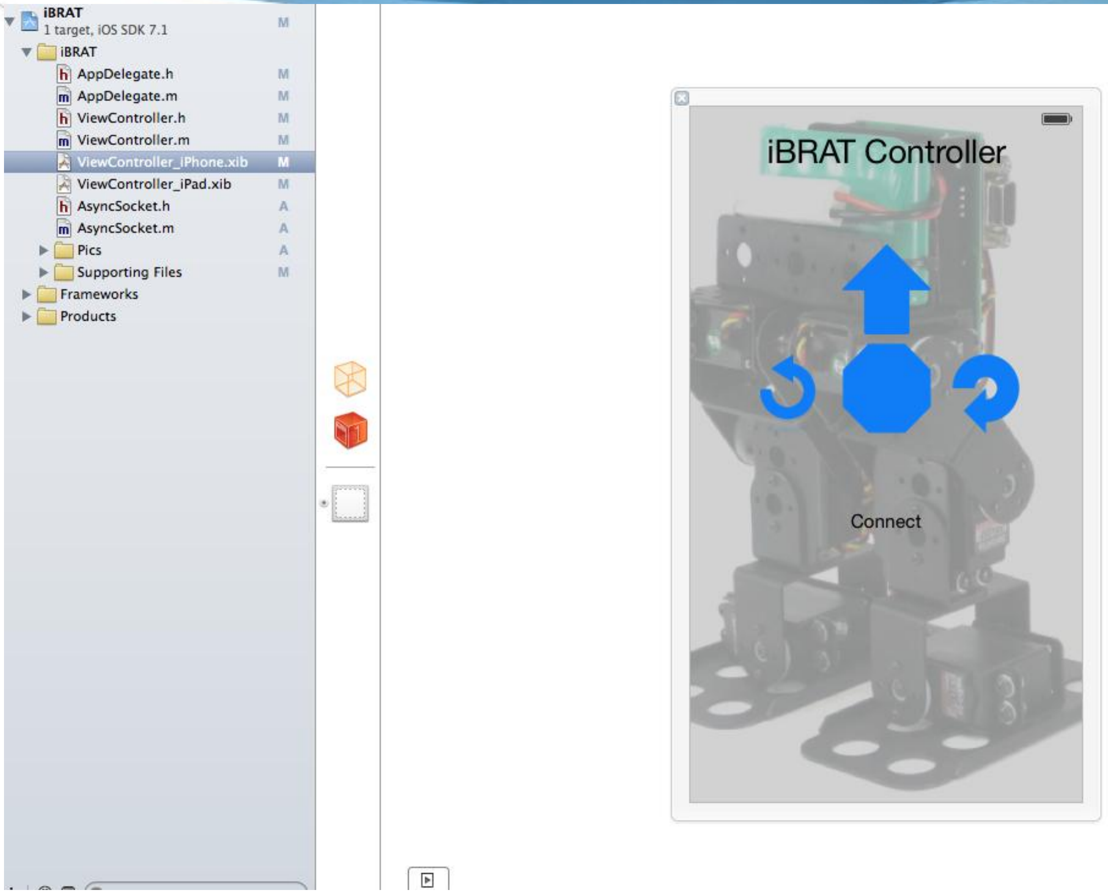


```
#include "simpletools.h"

void Right(void *par)

{

int  LAPin = 19, RAPin = 16, LKPin = 18;
int  RKPin = 15, LHPin = 17, RHPin = 14, j;
int R1[] = {900, 950, 1000, 1050, 1100};
int R2[] = {1101, 1164, 1175, 1214, 1375};

   servo_angle(RAPin,R1[2]);
   servo_angle(RKPin,R1[1]);
   servo_angle(RHPin,R1[0]);
   servo_angle(LAPin,R1[3]);
   servo_angle(LKPin,R1[2]);
   servo_angle(LHPin,R1[4]);

pause (1000);

while(1)
{
for (j=14;j<=19;j++);
{servo_setramp(j, 8);}

   servo_angle(LAPin,R1[1]);
   servo_angle(RAPin,R1[0]);
   pause (500);
```



```
   servo_angle(RHPin,R1[2]);
   servo_angle(RKPin,R1[1]);
   servo_angle(LHPin,R1[4]);
   servo_angle(LKPin,R1[3]);
  pause (500);

// Second step

   servo_angle(LAPin,R1[2]);
   servo_angle(RAPin,R1[3]);
  pause (500);
   servo_angle(RHPin,R1[2]);
   servo_angle(RKPin,R1[1]);
   servo_angle(LHPin,R1[4]);
   servo_angle(LKPin,R1[3]);
  pause (500);

   servo_angle(LAPin,R1[2]);
   servo_angle(RAPin,R1[3]);
  pause (500);
   servo_angle(RKPin,R1[1]);
   servo_angle(RHPin,R1[0]);
   servo_angle(LKPin,R1[2]);
   servo_angle(LHPin,R1[4]);
  pause (500);

}
}
```

BRAT App



```
iBRAT                                   M
1 target, iOS SDK 7.1
  iBRAT
    AppDelegate.h                       M
    AppDelegate.m                       M
    ViewController.h                    M
    ViewController.m                    M
    ViewController_iPhone.xib           M
    ViewController_iPad.xib             M
    AsyncSocket.h                       A
    AsyncSocket.m                       A
    Pics                                A
    Supporting Files                    M
  Frameworks
  Products
```