```
//oiOSoul

//By Fahed Riachi

// fadlab 2014


#include <Servo.h>

#include <Adafruit_NeoPixel.h>




////////////// Eyes (NeoPixels) initialization ///////////


Adafruit_NeoPixel strip = Adafruit_NeoPixel(2, 12, NEO_GRB + NEO_KHZ800);


// Set some standard colors to be use, they become handy instead of remeering and typing each time
the RGB codes

uint32_t black = strip.Color(0, 0, 0);

uint32_t white = strip.Color(255, 255, 255);

uint32_t magenta = strip.Color(255, 0, 255);

uint32_t red = strip.Color(255, 0, 0);

uint32_t green = strip.Color(0, 255, 0);

uint32_t blue = strip.Color(0, 0, 255);

uint32_t blueMar = strip.Color(0, 157, 255);

uint32_t orange = strip.Color(250, 175, 0);

uint32_t yellow = strip.Color(255, 255, 0);

uint32_t nightBlue = strip.Color(35,   0,   135);


////////////// Servos initialization ////////////////////
```

```
// Define servos/ joints (leg, waste,head)

Servo servoLeg;

Servo servoWaste;

Servo servoHead;


//Define reference positions


//leg

int centerLeg=1580;

int leftLeg=2300;

int rightLeg=670;

//waste

int centerWaste=1300;

int forwardWaste=1670;

int backwardWaste=777;

//head

int centerHead=1150;

int leftHead=2400;

int rightHead=600;



//define variable to manage proximity detection

int sensor;

long unsigned detectionTime = 0;
long unsigned prevDetectionTime = 0;

int detectionCountFast = 0;

int detectionCountSlow = 0;

int fastDetection = 7000; // time between 2 consequetive detections where detectionCountFast is
incremented.

boolean proxDetected = false;



//define variables to manage sound detection

int sound;

float rhythm = 0.3;  //0.2 and 0.3 is nice for oiO

int countOfDetectedBeats = 0;

long unsigned lastBeat = 0;

int allowedPauseBetweenSequences = 2000;


//time reflexes timer

long unsigned lastAction = 0;

int sneezeTimer = 65000; // each 65 seconds, oiO will sneeze

int sleepIfNoActionTimer = 45000; // oiO will dose off and sleep if no action done by user for more than
30 sec


void setup() {


  //Serial.begin(9600);
 //Attach servos to their corresponding control pins

  servoLeg.attach(3);

  servoWaste.attach(5);

  servoHead.attach(6);



  //attache Sound sensor (digital input) to Pin 8

  pinMode(8, INPUT);




  //Start with 'oiO' standing posture (all servos at center)

  servoLeg.writeMicroseconds(centerLeg);

  servoWaste.writeMicroseconds(centerWaste);

  servoHead.writeMicroseconds(centerHead);

  delay(1000);

  //start the eyes object

  strip.begin();

  strip.setBrightness(180);

  strip.show(); // Initialize all pixels to 'off'

  setEyesColor(white);




}


void loop() {
```

```
//get fresh values for oiO 3 senses: proximity, sound and ime

sensor = analogRead(A0);

sound = digitalRead(8);


//Time based reflexes


//oiO will go to sleep if no action by user for more than 30 seconds

if((millis() - lastAction) > sleepIfNoActionTimer) {

  doseOff();

}


//oiO will sneeze each 1 min (will not be recorded as a user action)

if(millis() % sneezeTimer == 0) {

  sneeze();

}




/* ##### Sound reflex procedure ####

oiO waits to hear a loud sound, and moves to it, it expects 10 sounds to by played in a row

if less than 10 sounds and greater than 4 were played, then it will turn red, and blink to the user as if
asking

for more, and it will keep waiting untill the user start making sounds again. Once more than

10 sounds are played, oiO will "hang" by doing repetetive 8 head bangs on his own, then freezes

suddenly, turns his head left for 2 seconds, then shy away to the right as if he is embaressed.

This will end the sound reflex procedure. oiO can do other stuff now, including repeating the
```

```
sound reflex if it hears a sound again */


if (sound == HIGH) {
    delay(100);
    if (sound == HIGH) {


      lastBeat = millis(); //record time of beat (since arduino was started/Reset)
      int x = random(1,5);
      //debug
      /*
      Serial.println("I heard a sound!! ");
      Serial.println(x);
      */
      switch(x){
        case 1:
        setEyesColor(magenta);
        move3Servos(servoLeg,centerLeg,servoWaste,1350,servoHead,1650,rhythm);
        move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,rhythm);
        break;


        case 2:
        setEyesColor(blue);
        move3Servos(servoLeg,centerLeg,servoWaste,1350,servoHead,800,rhythm);
        move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,rhythm);
        break;
        case 3:

        setEyesColor(red);

        move3Servos(servoLeg,1320,servoWaste,centerWaste,servoHead,800,0.5);

        move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,rhythm);

        break;


        case 4:

        setEyesColor(yellow);

        move3Servos(servoLeg,1900,servoWaste,centerWaste,servoHead,1800,rhythm);

        move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,rhythm);

        break;


      }

  setEyesColor(white);

  countOfDetectedBeats++; //count detected beats

  lastAction = millis(); //stores endtime of this action

  }

}


if((millis()-lastBeat) > allowedPauseBetweenSequences && countOfDetectedBeats >=4 &&
countOfDetectedBeats <= 10)

{ //protest and ask for more rhythm

  rhythm = 0.5; //speed up oiO's movement in next soundSequence

  setEyesColor(red);

  move3Servos(servoLeg,1700,servoWaste,1350,servoHead,1580,0.1);
  delay(1000);

  setEyesColor(black);

  delay(80);

  setEyesColor(red);

  delay(700);

  setEyesColor(black);

  delay(80);

  setEyesColor(red);

  lastBeat = millis();

  countOfDetectedBeats = 0;

  lastAction = millis(); //stores endtime of this action

}


 if(countOfDetectedBeats > 10)

  {//user has performed more than 10 beats in a row, thus oiO will hang on the beat for a while , stop,
then shy away, then reset all related variables

   for(int i=1;i<=8;i++)

    {

      move3Servos(servoLeg,centerLeg,servoWaste,1350,servoHead,1650,rhythm);

      move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,rhythm);

    }

   move3Servos(servoLeg,centerLeg,servoWaste,1100,servoHead,centerHead,0.3);

   delay(1000);

   move3Servos(servoLeg,centerLeg,servoWaste,1100,servoHead,1580,0.08);

   delay(2000); //pause to realize that the user has stoped

   setEyesColor(orange);
   move3Servos(servoLeg,1000,servoWaste,1600,servoHead,750,5.0); //shy away

   delay(1000);

   move3Servos(servoLeg,1000,servoWaste,1600,servoHead,2200,0.05);

   delay(600);

   move3Servos(servoLeg,1000,servoWaste,1600,servoHead,750,0.2);

   countOfDetectedBeats = 0;

   rhythm = 0.3;

   lastBeat = 0;

   detectionCountFast = 1;

   detectionCountSlow = 1;

   lastAction = millis(); //stores endtime of this action

  }
```

/* ##### Proximity reflex procedure ####

oiO waits until the user approaches his head/neck, it will react by moving away (leaning backwards), in parallel

he started keeping track of time between each time the user approaches him. If the user approaches him quickly

after the first time, his eyes will turn yellow trying and again leans backward more quickly trying to indicate

his is not happy by the harrasement of the user. the 3rd time he is approached quickly, his eyes become red, he

leans very quickly to the back and anxiously nod his head left and right in disagreement, then go back to standing

position, with his eyes breathing fading red. when he calms down, he lights up in white.

oiO can do other stuff now, including repeating the sound or proximity reflex depending on the user behavior.

```
If the user approaches him with more time between approaches, oiO behavior will be differernt, his
eyes go magenta

indicating that he likes the soft moves, and do a bit of a show, then looks at the user a blinks asking for
another

slow approach. if this happens, then oiO will go into fast, color changing movement expressing his
happines, eyes goes

through all rainbow color, and at the end he leans down slowley to rest. Then he lights up in white.

*/


// Procedure to detect if the user is frequencly or seldomly approaching to oiO. Accordingly manages
(set and resets (for each 3 detections)) detectionCountFast & detectionCountSlow counters

if(sensor < 250)

 {

 delay(150); //debouncing delay and recheck if object still close, if yes then consider it a real detection

 if(sensor < 250)

 {



  detectionTime = millis();

  if (detectionCountFast !=0)

  {

   if (detectionTime-prevDetectionTime < fastDetection )

    {

    detectionCountFast++;

    detectionCountSlow = 1;

    }

   else
}

    {
    detectionCountSlow++;
    detectionCountFast = 1;
    }
  }
  else
  {
  detectionCountFast++;
  detectionCountSlow++;
  }
  if(detectionCountFast > 3) detectionCountFast=1; //reset detectionCountFast counter each 3 fast
detections in a row
  if(detectionCountSlow > 3) detectionCountSlow=1; //reset detectionCountSlow counter each 3 slow
detections in a row



  prevDetectionTime = detectionTime;
  proxDetected = true;


  /*
  Serial.print("countFast: ");
  Serial.print(detectionCountFast);
  Serial.print(" CountSlow: ");
  Serial.println(detectionCountSlow);
  */
 }
```

```
}//end if & of detection counters procedure



//proxility reflex execution


if(proxDetected == true && detectionCountFast == 1 && detectionCountSlow == 1) //indiffererrnt
 {
  setEyesColor(white);
  //randomize this move as it has more probability to happen
  int x = random(1,2);
  if(x == 2)
  {
    move3Servos(servoLeg,centerLeg,servoWaste,1000,servoHead,centerHead,0.05);
    delay(1000);
    move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);
  }
  else
  {
    move3Servos(servoLeg,1850,servoWaste,1000,servoHead,centerHead,0.05);
    delay(1000);
    move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);
  }
  proxDetected = false;
  lastAction = millis(); //stores endtime of this action
 }
```

```
else if (proxDetected == true && detectionCountFast == 2 && detectionCountSlow == 1) //tensed
{
  setEyesColor(orange);
  move3Servos(servoLeg,1700,servoWaste,900,servoHead,centerHead,0.05);
  delay(1000);
  move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.08);
  proxDetected = false;
  lastAction = millis(); //stores endtime of this action
}
else if(proxDetected == true && detectionCountFast == 3 && detectionCountSlow == 1) //angry
{
setEyesColor(red);
move3Servos(servoLeg,centerLeg,servoWaste,800,servoHead,1580,0.08);
move3Servos(servoLeg,centerLeg,servoWaste,800,servoHead,730,0.3);
move3Servos(servoLeg,centerLeg,servoWaste,800,servoHead,1580,0.3);
move3Servos(servoLeg,centerLeg,servoWaste,800,servoHead,centerHead,0.3);
delay(1000);
move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);
dimmer(10,255,255,0,0,3,3);
dimmer(10,255,255,0,0,3,3);
delay(2000);
setEyesColor(white);
proxDetected = false;
lastAction = millis(); //stores endtime of this action
}
```

```
else if(proxDetected == true && detectionCountFast == 1 && detectionCountSlow == 2)  //joyful

{

setEyesColor(magenta);

delay(500);

move3Servos(servoLeg,centerLeg,servoWaste,1500,servoHead,centerHead,0.03);

move3Servos(servoLeg,centerLeg,servoWaste,1300,servoHead,1580,0.05);

move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,750,0.05);

move3Servos(servoLeg,1100,servoWaste,centerWaste,servoHead,1900,0.5);

delay(1000);

setEyesColor(black);

delay(100);

setEyesColor(magenta);

delay(800);

setEyesColor(black);

delay(80);

setEyesColor(magenta);

delay(2000);

proxDetected = false;

lastAction = millis(); //stores endtime of this action

}


else if(proxDetected == true && detectionCountFast == 1 && detectionCountSlow == 3)  //happy

{

setEyesColor(yellow);

move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);
```

```
  move3Servos(servoLeg,670,servoWaste,centerWaste,servoHead,2300,0.09);

  move3Servos(servoLeg,2300,servoWaste,centerWaste,servoHead,600,0.09);

  delay(1000);

  move3Servos(servoLeg,2300,servoWaste,800,servoHead,600,0.05);

  move3Servos(servoLeg,2300,servoWaste,1250,servoHead,600,0.05);

  delay(700);

  setEyesColor(blueMar);

  move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);

  move3Servos(servoLeg,2300,servoWaste,centerWaste,servoHead,600,0.1);

  move3Servos(servoLeg,670,servoWaste,centerWaste,servoHead,2300,0.1);

  delay(1000);

  move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);

  rainbow(20);

  move3Servos(servoLeg,centerLeg,servoWaste,1520,servoHead,centerHead,0.01);

  setEyesColor(white);

  proxDetected = false;

  lastAction = millis(); //stores endtime of this action

  }



/*

rainbowCycle(20);

theaterChase(strip.Color(127,   0,   0), 50);

theaterChaseRainbow(50);

dimmer(1,30,255,0,255,0,3,3);
```

```
*/
}



/////// Call back functions ///////


//Time-based reflexe functions, triggered by timers initialized in setup();
void sneeze()
{
move3Servos(servoLeg,centerLeg,servoWaste,900,servoHead,centerHead,0.04);
setEyesColor(black);
move3Servos(servoLeg,centerLeg,servoWaste,1500,servoHead,centerHead,0.3);
move3Servos(servoLeg,centerLeg,servoWaste,1150,servoHead,centerHead,0.3);
rainbowCycle(2);
move3Servos(servoLeg,centerLeg,servoWaste,centerWaste,servoHead,centerHead,0.05);
setEyesColor(white);
}


void doseOff()
{
dimmer(10,255,35,0,135,2,25); //fade out nightBlue eyes..slowly
move3Servos(servoLeg,centerLeg,servoWaste,1400,servoHead,centerHead,2.0);
delay(1500);
move3Servos(servoLeg,centerLeg,servoWaste,1700,servoHead,1000,2.0);
delay(1500);
```

```
move3Servos(servoLeg,centerLeg,servoWaste,1850,servoHead,1000,2.0);

delay(4000);

move3Servos(servoLeg,centerLeg,servoWaste,1070,servoHead,centerHead,0.1);

//dimmer(10,120,35,0,135,1,25);

setEyesColor(black);

delay(500);

setEyesColor(nightBlue);

delay(500);

move3Servos(servoLeg,centerLeg,servoWaste,1070,servoHead,1000,0.08);

setEyesColor(black);

delay(300);

setEyesColor(nightBlue);

delay(300);

move3Servos(servoLeg,centerLeg,servoWaste,1070,servoHead,centerHead,0.08);

setEyesColor(black);

delay(200);

setEyesColor(nightBlue);

delay(5000);

move3Servos(servoLeg,1200,servoWaste,1060,servoHead,1500,0.02);

dimmer(10,255,35,0,135,2,25); //fade out nightBlue eyes..slowly

setEyesColor(black); //shut eyes

move3Servos(servoLeg,1200,servoWaste,1400,servoHead,1500,3.0);

delay(1500);

move3Servos(servoLeg,1200,servoWaste,1600,servoHead,1500,3.0);

delay(1500);
```

```
//move3Servos(servoLeg,1000,servoWaste,1900,servoHead,1000,3.0);

for(int i=1600;i<=2300;i+=2){

servoWaste.writeMicroseconds(i);

delay(100);

}

setEyesColor(black);//close eyes

  while(true){ //Deep sleep untill oiO hears a sound

    sound = digitalRead(8);

    if (sound == HIGH)

    {

    delay(100);

      if (sound == HIGH) {

      move3Servos(servoLeg,centerLeg,servoWaste,1070,servoHead,centerHead,0.1);

      rainbowCycle(5);

      break;

      }

    }


  }

}




//////////////////////////// Servo functions ////////////////////////////////

void move3Servos(Servo servo1Name, float target_pos1,Servo servo2Name, float target_pos2,Servo
servo3Name, float target_pos3,float easing){
  //  target_posx: has limits on start and end positions, between 600 and 2400, yet other servos are between 1000 and 2000...

  //  easing:

  //  is a value between 0.0 and 5.0, where 0.0 to 1.0 cause the servos to start movement fast but end slowely, values around zero (ex: 0.05) gives much slower/desired response of smoothing, value of 1 is fast

  //  For values between 1 > easingx > 5 , the easing behavior changes, causing servos to move linearly start to stop, the easing controls the speed to move from current to target positions


    //Prevent values out of range

    target_pos1 = constrain(target_pos1,600,2400);

    target_pos2 = constrain(target_pos2,600,2400);

    target_pos3 = constrain(target_pos3,600,2400);


    easing = constrain(easing,0.0,5.0);


   //get current servo pulse width (i.e position)

   float current_pos1 = servo1Name.readMicroseconds();

   float current_pos2 = servo2Name.readMicroseconds();

   float current_pos3 = servo3Name.readMicroseconds();



   //difference of positions

   float diff1;

   float diff2;

   float diff3;
```

```
while(true){
  // calculate the required travel
  diff1 = target_pos1 - current_pos1;
  diff2 = target_pos2 - current_pos2;
  diff3 = target_pos3 - current_pos3;

  // Move only when target position is not reached
  if( fabs(diff1) > 5.0 ) { //then Move
    if(easing <= 1.00) current_pos1 += diff1 * easing;
      else { //when easing value > 1.0, then behavior is to move individual increments
        if(diff1 > 0.0) current_pos1 += easing;
        else current_pos1 -= easing;
    }

     servo1Name.writeMicroseconds( (int)current_pos1 );
  }
  if( fabs(diff2) > 5.0 ) { //then Move
    if(easing <= 1.00) current_pos2 += diff2 * easing;
      else { //when easing value > 1.0, then behavior is to move individual increments
        if(diff2 > 0.0) current_pos2 += easing;
        else current_pos2 -= easing;
    }
     servo2Name.writeMicroseconds( (int)current_pos2 );
  }
 if( fabs(diff3) > 5.0 ) { //then Move
```

```
     if(easing <= 1.00) current_pos3 += diff3 * easing;
       else { //when easing value > 1.0, then behavior is to move individual increments
         if(diff3 > 0.0) current_pos3 += easing;
         else current_pos3 -= easing;
     }
      servo3Name.writeMicroseconds( (int)current_pos3 );
   }
   //when all positions are reached, then stop
   else if(fabs(diff1)<=5.0 && fabs(diff2) <=5.0 && fabs(diff3) <=5.0){
      // if diff almost 0, this means target position reached, thus remain in place and exit loop
      break;
     }

   // Delay so the servos can physically reach the positions (important, but open to experimentation!)
   delay(15);

  }//end while

}//end function
////////////////////////////////////// End of Servos functions ////////////////////////////////////




//////////////////////////////////////    Eyes functions   /////////////////////////////////////////
```

```
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
}
```

```
//Theatre-style crawling lights.

void theaterChase(uint32_t c, uint8_t wait) {

  for (int j=0; j<10; j++) {  //do 10 cycles of chasing

    for (int q=0; q < 3; q++) {

      for (int i=0; i < strip.numPixels(); i=i+3) {

        strip.setPixelColor(i+q, c);    //turn every third pixel on

      }

      strip.show();


      delay(wait);


      for (int i=0; i < strip.numPixels(); i=i+3) {

        strip.setPixelColor(i+q, 0);        //turn every third pixel off

      }

    }

  }

}


//Theatre-style crawling lights with rainbow effect

void theaterChaseRainbow(uint8_t wait) {

  for (int j=0; j < 256; j++) {     // cycle all 256 colors in the wheel

    for (int q=0; q < 3; q++) {

        for (int i=0; i < strip.numPixels(); i=i+3) {

          strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third pixel on
        }
        strip.show();


        delay(wait);


        for (int i=0; i < strip.numPixels(); i=i+3) {
          strip.setPixelColor(i+q, 0);        //turn every third pixel off
        }
    }
  }
}


// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

```
void dimmer(int a, int b, int red, int green, int blue,int fadeDirection, int stepSpeed){

//takes a color 'RGB' and a range of brighness 'start' & 'end' and goes up=1, down=2, or updown="3
according to 'fadeDirection', with a given 'stepSpeed' between steps


switch(fadeDirection){


  case 1:

  for(int i=a;i<=b;i++){

   strip.setPixelColor(0, (i*red/255) , (i*green/255), (i*blue/255));

   strip.setPixelColor(1, (i*red/255) , (i*green/255), (i*blue/255));

   strip.show();

   delay(stepSpeed);

  }

  break;


  case 2:

  for(int i=b;i>=a;i--){

  strip.setPixelColor(0, (i*red/255) , (i*green/255), (i*blue/255));

  strip.setPixelColor(1, (i*red/255) , (i*green/255), (i*blue/255));

  strip.show();

  delay(stepSpeed);

  }

  break;


  case 3:
```

```
  for(int i=a;i<=b;i++){
   strip.setPixelColor(0, (i*red/255) , (i*green/255), (i*blue/255));
   strip.setPixelColor(1, (i*red/255) , (i*green/255), (i*blue/255));
   strip.show();
   delay(stepSpeed);
  }
  for(int i=b;i>=a;i--){
  strip.setPixelColor(0, (i*red/255) , (i*green/255), (i*blue/255));
  strip.setPixelColor(1, (i*red/255) , (i*green/255), (i*blue/255));
  strip.show();
  delay(stepSpeed);
  }
  break;


}
}


void setEyesColor(uint32_t color)
{
strip.setPixelColor(0,color);
strip.setPixelColor(1,color);
strip.show();
}
```