Detailed design of demo lectures using Arduino

Miguel Ángel Rubio Escudero, Rocío Romero Zaliz (translator) - Computer Science Department - University of Granada, Spain.

Table of Contents

1. Introduction

In this section we describe the set of demos that have been designed for the lectures. We begin with a description of the platforms developed and continue with a brief description of each demo and the required software.

The two platforms that have been developed for the classroom demos are based on the Arduino board. It is recommended, but not mandatory, to mount the card and prototyping board on a common platform to facilitate the overall experience.

2. Platform #1

The first platform is designed to perform a simple lab session at the beginning of the course. It comprises a `LDR-VT90N2` photoresistor, a `Sparkfun TMP36` temperature sensor and five LEDs of different colours whose description is attached.

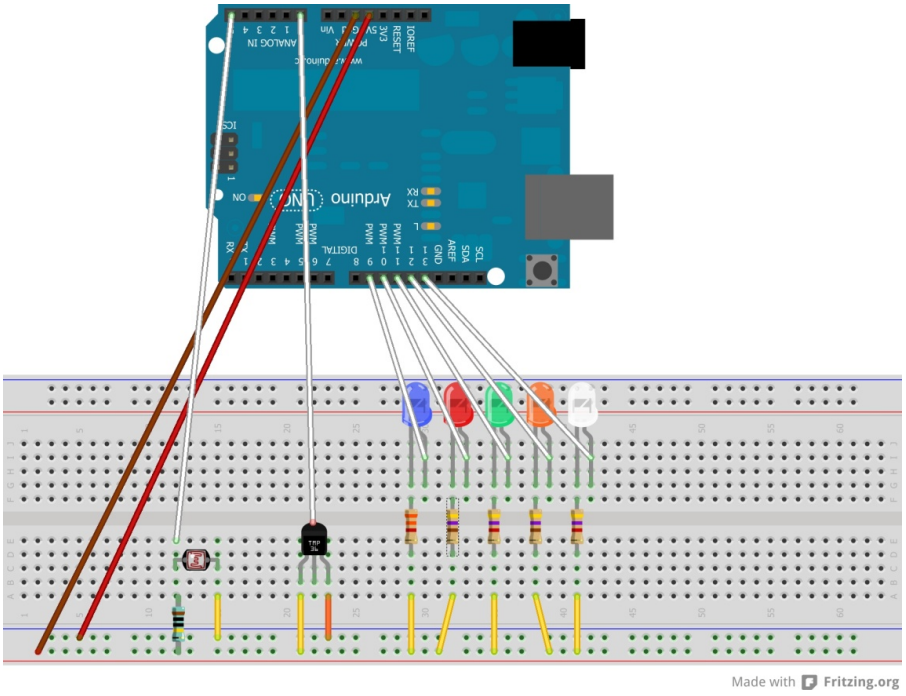

Figure 2-1. *Diagram of platform #1*

It is important to use LEDs with different colours since students may have problems to distinguish them only based on their position from far away in the lab room. The only way they can discriminate them accurately is if they have different colours.

Code	Reference	Description
LEDWC503B	WAN-CBADA151	5 mm white. IF= 25mA VF=3.2 V 9000mcd 15º
619-4937	L-7113SRD-H	5 mm diffused red. IF=20mA. VF=1.85V 1000mcd. 30º
262-2955	L-53SED	5mm diffused orange. IF=20mA. VF=2V 800mcd. 60º
466-3548	L-7113QBC-D	5mm blue. IF=20mA. VF=3.5V 2200mcd. 15º
466-3510	L-7113VGC-H	5mm green. IF=20mA. VF=3.7V. 14000mcd. 20º

Table 2-1. *Characteristics of the LEDs used in platform #1.*

During demos the students commented that sometimes the intensity of the LEDs was excessive and decided to calibrate its intensity. After conducting a study with students in the classroom we obtained the following calibration data.

Resistance used with each LED:

- Diffused red LED -> 470 ohms
- Diffused orange LED -> 470 ohms
- White LED -> 5 Kohms
- Blue LED -> 3 Kohms
- Green LED -> 5 Kohms

It is expected that for other classrooms it may be necessary to adjust the resistance values.

3. Platform #2

The second platform is designed to perform more sophisticated lab sessions as the course progresses. It consists of a `Sparkfun ROB-09065` servomotor, a `Sparkfun COM-07950` mini-speaker and a `LV-MaxBotix-EZ1` distance sensor. The servomotor has two

LEDs attached that allow students to perceive the movement of the servomotor from back in the classroom.

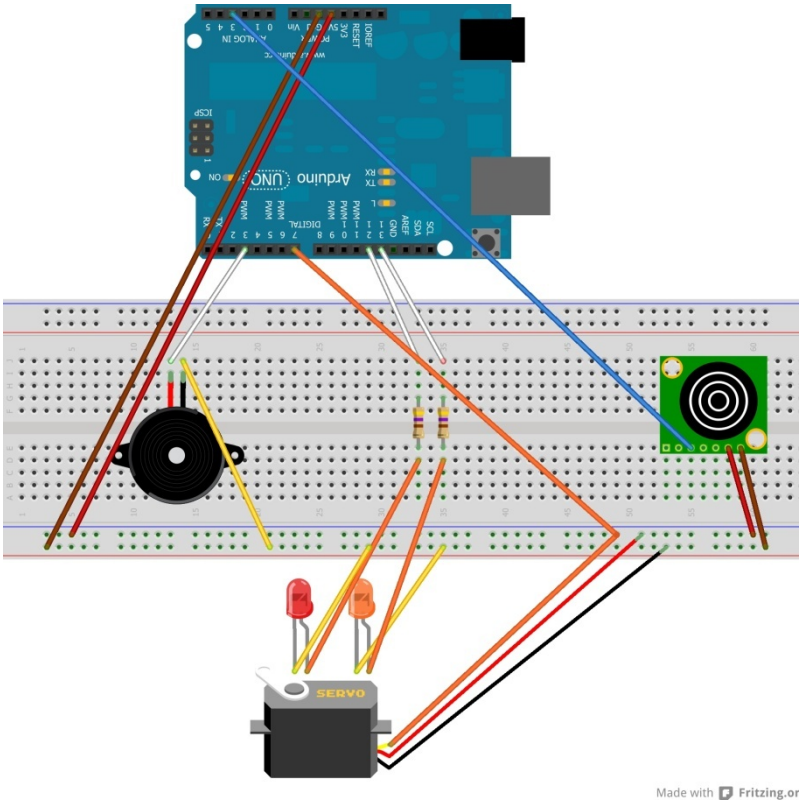

Figure 3-1. *Diagram of platform #2*

619-4937	L-7113SRD-H	5 mm diffused red. IF=20mA. VF=1.85V 1000mcd. 30º
262-2955	L-53SED	5mm diffused orange. IF=20mA. VF=2V 800mcd. 60º

Table 3-1. *Characteristic of the LEDs used in platform #2.*

Resistance used with each LED:

- Diffused red LED -> 470 ohms
- Diffused orange LED -> 470 ohms

4. Software library common to both platforms

Both platforms use a set of libraries developed during the project. These libraries contain a set of functions with names in Spanish to ease the understanding of the students.

Below is the code of these libraries:

```
/*
  PhysicalComputingDemonstrations.h - Physical Computing for Lectures library
  Written by Miguel Angel Rubio

*/


#ifndef PHYSICAL_COMPUTING_DEMONSTRATIONS_H_
#define PHYSICAL_COMPUTING_DEMONSTRATIONS_H_

#include <Arduino.h>
//Limitations of Arduino IDE...
#include "../Servo/Servo.h"

////////////////////////////////////
// Read Voltage
////////////////////////////////////

int lee_voltaje(int num_pin)
{
    int valor;
    valor = analogRead(num_pin);

    return valor;
}
```

Figure 4-1. *Code of the Arduino common library (Part 1).*

```
//////////////////////////////////////
// Turn off LEDs
//////////////////////////////////////

// Turn off a single LED

void apaga_led(int num_led)
{
    pinMode(num_led, OUTPUT);
    digitalWrite(num_led, LOW);
}

// Turn off multiple LEDs

void apaga_led(int num_led[], int size_led)
{
    for (int cont = 0; cont < size_led; cont++){
        pinMode(num_led[cont], OUTPUT);
        digitalWrite(num_led[cont], LOW);
        }
}

//////////////////////////////////////
// Turn on LEDs
//////////////////////////////////////

// Turn on a single LED

void enciende_led(int num_led)
{
    pinMode(num_led, OUTPUT);
    digitalWrite(num_led, HIGH);
}

// Turn on multiple LEDs

void enciende_led(int num_led[], int size_led)
{
    for (int cont = 0; cont < size_led; cont++){
        pinMode(num_led[cont], OUTPUT);
        digitalWrite(num_led[cont], HIGH);
        }
}

//////////////////////////////////////
// Move Servo
//////////////////////////////////////

// Move servo to a certain position
// It assumes the servo is plugged in

void mover_servo(Servo mi_servo, int posicion)
{
    mi_servo.write(posicion);
```

Figure 4-2. *Code of the Arduino common library (Part 2).*

```
////////////////////////////////////
// Pause for some seconds
////////////////////////////////////

void para (float segundos)
{
    delay(segundos*1000);
}

//////////////////////////////////
// Read temperature
//////////////////////////////////

// Reads temperatura using Arduino.
// The sensor is TMP36 from the Arduino Starter Kit
// Inpunt analog pin: The pin that is going to be used to read the temperature
// Output: The temperatura

int lee_temperatura(int analog_pin){

 char str[256];

 // We perform 20 measurements to minimize error rate
 float temp = 0;
 int num_medidas = 20;
 float voltaje;

 for (int cont = 0; cont < 20; cont++) {
    voltaje = lee_voltaje(analog_pin); //Medida directa
    voltaje = voltaje*5/1024; //Voltaje normalizado
    temp= temp+voltaje/0.01 - 50; //En C
 }

 // Take the mean
 int temperatura = temp/num_medidas;
 return temperatura;

}

//////////////////////////////////
// Read brightness
//////////////////////////////////

// See the brightness fo the room
// Th sensor is LDR-VT90N2
// See the pdf attached
// Input analog_pin: The pin that is going to be used to read the brightness
// Output: The value of brightness normalized between 0 and 1

float lee_luminosidad (int analog_pin){
  float voltaje = lee_voltaje(analog_pin);
  voltaje = voltaje/1024.;
  return voltaje;
}

#endif // PHYSICAL_COMPUTING_DEMONSTRATIONS_H_
```

Figure 4-3. *Code of the Arduino common library (Part 3).*

5. Demo #1: Variables

The demos are designed to cover different aspects of the teaching of programming. For the demos to be useful it is not enough to perform the demos in the classroom, you need to engage students in the demo by voting, small group discussions, etc. (Crouch et al. 2004)

The purpose of this demo is twofold. On one hand we want to present the Arduino board and explain why we do this kind of demos. On the other hand we want students to face some of the most common conceptual misunderstandings. Through voting and small group discussions we encourage students to resolve their doubts.

One way to raise the discussion would be:

c = 3;
run destellos(c); -> there are three flashes

*a = c*2;*
run destellos(a); -> there are six flashes

c = 2; -> discussion: How many flashes will there be? 4 or 6? Why?
run destellos(a)

*a = a*2 - 5; -> discussion: How many flashes will there be? 6 or 7? Why?*
run destellos(a)

The code to be used in this demo can be seen in Figure 5-1.

```
/*
  Example of the use of variables
  This example code is in the public domain.
 */
#include <PhysicalComputingDemonstrations.h>

void setup() {

  int leds[5] = {9,10,11,12,13};
  apaga_led(leds,5);

  int a = 3;
  int b = 7;

  b = a * 3 + 2;
  a = 1;
  destellos(b);
}

// The loop routine runs over and over again forever
void loop() {

}

// Function apagar: Turns off two LEDs (13 y 9) in Arduino
void apagar() {
  int led1 = 13;
  int led2 = 9;

  apaga_led(led1);
  apaga_led(led2);
}

// Function encender: Turns on two LEDs (13 y 9) in Arduino
void encender() {
  int led1 = 13;
  int led2 = 9;

  enciende_led(led1);
  enciende_led(led2);
}

// Function destellos: Makes num flashes
// Input num_destellos: The number of flashes to make
void destellos(int num_destellos) {
  int led1 = 13;
  int led2 = 9;

  for (int cont = 0; cont < num_destellos; cont++) {
    enciende_led(led1);
    apaga_led(led1);
    para(1);

    enciende_led(led2);
    apaga_led(led2);
    para(1);
  }

  apaga_led(led1);
  apaga_led(led2);
}
```

Figure 5-1. *Code of Demo #1.*

6. Demo #2: Arrays

In this demo we will work with arrays. We will use an array to store a song and proceed to work with. Several authors (Moor 2010; Ruthmann et al. 2010) have shown that music can be very effective in presenting concepts such arrays, repeat loops or functions. In this demo is necessary to use the platform #2.

The tune we have selected is the Imperial March soundtrack from Star Wars. Our experience indicates that students of this generation recognize it and gets their attention. Attached is the sheet music for this soundtrack.

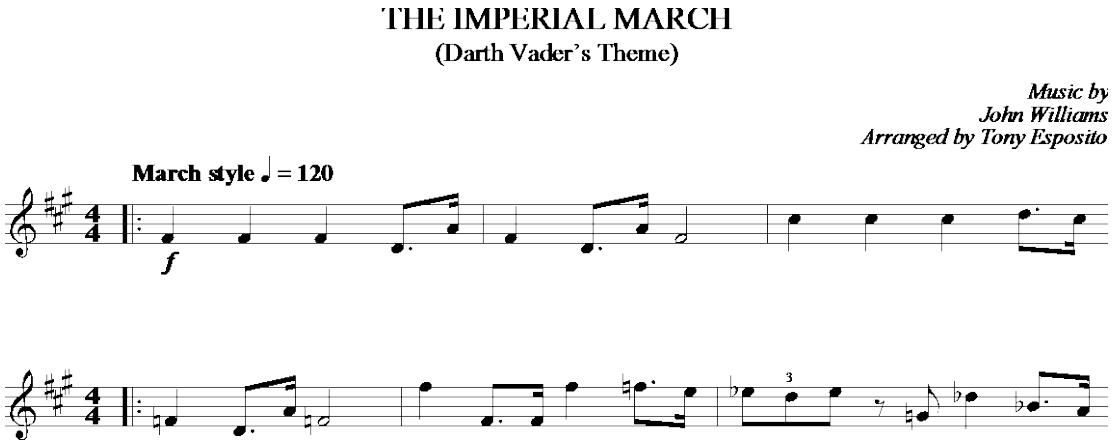

Figure 6-1. *Sheet music for the Imperial March (© 1980 WARNER-TAMERLANE PUBLISHING CORP. & BANTHA MUSIC, under the conditions of fair use).*

One way to raise the discussion would be:

> *Play multiple notes one statement at a time.*
>
> *Play the whole song using the array.*
>
> *Select different subarrays and play them.*

The code to be used in this demo can be seen in Figure 6-3.

```
/*
  Example of th euse of arrays
  See the instructions.txt file
  This example code is in the public domain. */

#include <PhysicalComputingDemonstrations.h>
#include "tonos.h"

void setup() {

  // Song's notes
  int melodia[] = {
  NOTA_SI_3, NOTA_SI_3, NOTA_SI_3, NOTA_SOL_3, NOTA_RE_4, NOTA_SI_3, NOTA_SOL_3, NO
TA_RE_4, NOTA_SI_3, NOTA_FA_SOST_4,
  NOTA_FA_SOST_4, NOTA_FA_SOST_4, NOTA_SOL_4, NOTA_RE_4, NOTA_SI_3, NOTA_SOL_3, NOT
A_RE_4, NOTA_SI_3};

  int tempo = 2000;
  int pin_altavoz = 3;
  int redonda = tempo;
  int blanca = tempo/2;
  int negra = tempo/4;
  int corchea = tempo/8;

  // Song's duration
  int duracion_nota[] = {negra, negra, negra, negra, corchea, negra, negra,
     corchea, blanca, negra, negra, negra, negra, corchea, negra,  negra, corchea,
blanca};


  // Play a note
  tocar (pin_altavoz, NOTA_SI_3, blanca);

  // Play several notes
  tocar (pin_altavoz, melodia, duracion_nota, 18);

}
```

Figure 6-2. *Code for demo #2 (Part 1).*

```
void loop() {
}


// Play a note
void tocar (int pin_altavoz, int nota, int duracion) {

    tone(pin_altavoz, nota,duracion);
    delay(duracion*1.3);
    noTone(pin_altavoz);
}

// Play several notes
void tocar (int pin_altavoz, int* melodia, int* duracion, int long_cancion) {

  // Play one note at a time
  for (int pos = 0; pos < long_cancion; pos++) {

    tone(pin_altavoz, melodia[pos],duracion[pos]);

    // Pause to space notes
    int pausa = duracion[pos] * 1.2;
    delay(pausa);

    // Stop the speaker
    noTone(pin_altavoz);
   }
}
```

Figure 6-3. *Code for demo #2 (Part 2).*

7. Demo #3: Inputs and outputs

In this demonstration we will work the concepts of input and output of a computer. To do so, we will distinguish between inputs and outputs of the computer. In this demo is necessary to use platform #1.

One way to raise the discussion would be:

1) *Read temperature*

   Run the function and see the temperature value

   Put your fingers on the sensor about 30 seconds

   Run the function and see that the temperature has risen

   Run it again after a while to see that the temperature has dropped.

2) *Write with LEDs*

   Tell them they have to guess how the LEDs are turned on or off

   Keep running tests until they guess

   Discuss different strategies of guessing (binary search, systematic ...)

The code to be used in this demo can be seen in Figure 7-1 and Figure 7-2.

```
/*
  Example of uso of inputs and outputs
  See instructions.txt file
  This example code is in the public domain.
*/

#include <PhysicalComputingDemonstrations.h>

void setup() {

Serial.begin(9600);
escribe(16);
}


void loop() {
  int temper = lee_temp_salida();
  Serial.println(temper);
  para(1);
}


// Function lee_temp_salida. Reads temperature using Arduino
// The sensor is the TMP36 of the Arduino Starter Kit
// Output: the temperature
int lee_temp_salida(){

 char str[256];
 // The pin we are going to use to reed the temperature
 int analog_pin = 0;

 // We run 20 measures to minimize the error
 float temp = 0;
 int num_medidas = 20;
 float voltaje;

 for (int cont = 0; cont < 20; cont++) {
    voltaje = lee_voltaje(analog_pin); // Direct measure
    voltaje = voltaje*5/1024; // Normalized voltage
    temp= temp+voltaje/0.01 - 50; // In °C
 }

 // Do the mean and return it
 int temperatura = temp/num_medidas;
 return temperatura;
}
```

Figure 7-1. *Code of demo #3 (Part 1).*

```
// Function lee_temp_salida. Reads temperature using Arduino
// The sensor is the TMP36 of the Arduino Starter Kit
// See the pdfs attached
void lee_temp(){

 char str[256];
 // The pin we are going to use to reed the temperature
 int analog_pin = 0;

 // We run 20 measures to minimize the error
 float temp = 0;
 int num_medidas = 20;
 float voltaje;

 for (int cont = 0; cont < 20; cont++) {
    voltaje = lee_voltaje(analog_pin); // Direct measure
    voltaje = voltaje*5/1024; // Normalized voltage
    temp= temp+voltaje/0.01 - 50; // In °C
 }

 // Do the mean and show it
 int temperatura = temp/num_medidas;
 sprintf(str, "La temperatura es    %d C",temperatura);
 Serial.println(str);

}

// LED output of the input binary number
// Maximun number 31
// We use all pins form 9 to 13
// Pin 13 is the least significant
// Pin 9 is the most significant
// To do the guessing game we need to use LEDs
// with different colours.
// Input num: the number we are going to transform into binary
void escribe (int num) {

// The LEDs which we are going to work with
int rango_led [5]= {9,10,11,12,13};
int val;

for (int cont = 0; cont < 5; cont++){
    val = num%2; // Calculate the remainder of the division by 2
    num = num/2; // Calculate the quotient and assign it to num

    // If binary value is 1
    if (val == 1)
        enciende_led(rango_led[cont]); // turn on LED
    // If not
    else
        apaga_led(rango_led[cont]); // Turn off LED
}

}
```

Figura 7-2. *Code of demo #3 (Part 2).*

8. Demo #4: Conditionals

In this demo we show how to use conditional structures. This demo takes the measure of the ambient light. If it's dark, the lights turn on, if there is enough, they turn off. This demos requires platform #1.

One way to raise the discussion would be:

> *Show the conditional code and show its operation*
>
> *Vary the environment conditions to trigger the light on and off*
>
> *Discuss the drawbacks of this implementation (conditional only runs once) to introduce the need for repeat loops*

The code to be used in this demo can be seen in Figure 8-1.

```
/*
  Example of the use of conditionals

  See instructions.txt file
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {

float luz = lee_luminosidad();

if (luz < 0.5)
  enciende_leds();
else
  apaga_leds();

}

void loop() {
}


// Function lee_luminosidad. Sense the brightness of the room
// The sensor is the LDR-VT90N2
// See the pdf attached
// Output: the brightness value normalized between 0 and 1float lee_luminosidad
(){
  int analog_pin = 5;
  float voltaje = lee_voltaje(analog_pin);
  voltaje = voltaje/1024.;
  return voltaje;
}

void apaga_leds() {
  int leds[5] = {9,10,11,12,13};
  apaga_led(leds,5);
}

void enciende_leds() {
  int leds[5] = {9,10,11,12,13};
  enciende_led(leds,5);
}
```

Figure 8-1. *Code of demo #4.*

9. Demo #5: Repetition

This demo is a continuation of demo #4. We extend the code of demo #4 to include repeat structures. Thus, the code runs for about a minute and you can see the changes that cause variation in brightness. We will use this platform #1.

One way to raise the discussion would be:

> *Remember demo #3 (conditionals)*
>
> *Include the loop and show its operation*
>
> *Add an additional condition using else if and function encender_mitad show its operation.*

The code to be used in this demonstration can be seen in Figure 9-2.

```
/*
  Example of the use of loops
  See instructions.txt file
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {


//for (int cont = 1; cont < 100; cont ++) {
//
//  float luz = lee_luminosidad();
//  if (luz < 0.5)
//     enciende_leds();
//   else
//     apaga_leds();
//  delay(200);
//}
//

for (int cont = 1; cont < 100; cont ++) {

  float luz = lee_luminosidad();
  if (luz < 0.8 && luz >= 0.5
  )
     enciende_mitad_leds();
   else if (luz < 0.5)
     enciende_leds();
   else
     apaga_leds();

  delay(200);
}
}
```

Figure 9-1. *Code of demo #5 (Part 1).*

```
// The loop routine runs over and over again forever:
void loop() {
}

// See the brightness fo the room
// Th sensor is LDR-VT90N2
// See the pdf attached
// Input analog_pin: The pin that is going to be used to read the brightness
// Output: The value of brightness normalized between 0 and 1
float lee_luminosidad (){
  int analog_pin = 5;
  float voltaje = lee_voltaje(analog_pin);
  voltaje = voltaje/1024.;
  return voltaje;
}

void apaga_leds() {
  int leds[5] = {9,10,11,12,13};
  apaga_led(leds,5);
}

void enciende_leds() {
  int leds[5] = {9,10,11,12,13};
  enciende_led(leds,5);
}

void enciende_mitad_leds() {
  int leds[5] = {9,11,13};
  enciende_led(leds,3);
}
```

Figure 9-2. *Code of demo #5 (Part 2).*

10. Demo #6: Advanced repetition

In this demo we reinforce the concepts of working with repeat loops and nested loops using the while statement. In this case we can show how to rotate the servomotor (we can present it as a beacon) using nested loops. We present the usefulness loops controlled by a condition by implementing a proximity alarm using the servomotor and the distance sensor. We will use platform #2.

One way to raise the discussion would be:

Present the code of the beacon and discuss its operation

Present the code of the beacon code with flashes and discuss its operation

Present the code of the proximity alarm. Discuss advantages and disadvantages of using a while structure

Do the demo.

The code to be used in this demo can be seen in Figure 10-1, Figure 10-2 and Figure 10-3.

```
/*
  Advanced examples of loops
  See instructions.txt file
  This example code is in the public domain.
 */
#include <PhysicalComputingDemonstrations.h>
#include <Servo.h>

void setup() {

// The name of the servomotor
int pin_servo = 7;
Servo mi_servo;
mi_servo.attach(pin_servo);

// The names of the LEDs
int pin_rojo = 12;
int pin_naranja = 13;

//// Turn on the LEDs
//enciende_led(pin_rojo);
//enciende_led(pin_naranja);


//// Repeat the rotation three times
//for (int cont = 1; cont <= 3; cont++){
//    // Rotate in one direction
//    for (int angulo=0; angulo <= 180; angulo = angulo +5){
//        mover_servo(mi_servo,angulo);
//        para(.1);
//    }
//
//    // Rotate in the other direction
//    for (int angulo=180; angulo <= 0; angulo = angulo - 5){
//        mover_servo(mi_servo,angulo);
//        para(.1);
//    }
```

Figure 10-1. *Code of demo #6 (Part 1).*

```
//// Complete version with flashes

//// Turn on the LEDs
//enciende_led(pin_rojo);
//enciende_led(pin_naranja);


//// Repeat the rotations three times
//for (int cont = 1; cont <= 3; cont++){
//
//    // Change the flash
//    enciende_led(pin_rojo);
//    apaga_led(pin_naranja);
//
//
//    // Rotate in one direction
//    for (int angulo=0; angulo <= 180; angulo = angulo +5){
//        mover_servo(mi_servo,angulo);
//
//        if (angulo > 90){
//          // Change the flash
//          apaga_led(pin_rojo);
//          enciende_led(pin_naranja);
//        }
//        para(.1);
//    }
//
//    // Rotate in the other direction
//    for (int angulo=180; angulo <= 0; angulo = angulo - 5){
//        mover_servo(mi_servo,angulo);
//        if (angulo > 90){
//          // Change the flash
//          apaga_led(pin_rojo);
//          enciende_led(pin_naranja);
//        }
//        para(.1);
//    }
//}

// See the distance value
float distancia = lee_distancia();

// Keep watching...
while (distancia > 30){
    Serial.println(distancia);
    // See the distance value
    distancia = lee_distancia();
    para(.5);
}
```

Figure 10-2. *Code of demo #6 (Part 2).*

```
// Turn on the LEDs
enciende_led(pin_rojo);
enciende_led(pin_naranja);

// Repeat the rotations three times
for (int cont = 1; cont <= 3; cont++){
    // Rotate in one direction
    for (int angulo=0; angulo <= 180; angulo = angulo +5){
        mover_servo(mi_servo,angulo);
        para(.1);
    }

    // Rotate in the other direction
    for (int angulo=180; angulo <= 0; angulo = angulo - 5){
        mover_servo(mi_servo,angulo);
        para(.1);
    }
}

apaga_led(pin_rojo);
apaga_led(pin_naranja);
mover_servo(mi_servo,90);

}


void loop() {
}

// Function lee_distancia. Reads the distance using Arduino
// The sensor is the LV-MaxSonar EZ1
float lee_distancia(){

 // The pin we are going to use to read the distance value
 int analog_pin = 3;
 // Number of measures
 int num_medidas = 15;

 // We take num_medidas measures to stabilize the result
 // The sensor values are not very stable
 float distancia_inst = 0;
 float voltaje;

 for (int cont = 0; cont < num_medidas; cont++){
    voltaje = lee_voltaje(analog_pin); // Direct measure
    voltaje = voltaje/2; // In units, normalized
    distancia_inst = distancia_inst + voltaje*2.54; // In cm

 }

 // We take the mean and return it
 float distancia = distancia_inst/num_medidas;
 return distancia;
}
```

Figure 10-3. *Code of demo #6 (Part 3).*

Bibliography

Sorva, Juha. 2012. “Visual Program Simulation in Introductory Programming Education.”