Diseño detallado de las demostraciones de teoría con Arduino

Miguel Ángel Rubio Escudero- Dpto. Ciencias de la Computación

Universidad de Granada

Índice

1. Introducción

En este apartado vamos a proceder a describir el conjunto de demostraciones que se han desarrollado para las clases de teoría. Comenzaremos con una descripción de las plataformas que se han diseñado para realizar las demostraciones y seguiremos con una breve descripción de cada demostración y el software necesario.

Las dos plataformas que se han desarrollado para las demostraciones en clase de teoría están basadas en la tarjeta Arduino. Es recomendable, aunque no necesario, montar la tarjeta y la placa de prototipado en una plataforma común. Esto facilita en gran medida la realización de las experiencias.

2. Diseño 1

La primera plataforma está diseñada para la realización de prácticas sencillas al principio de curso. Consta de un fotoresistor `LDR-VT90N2` un sensor de temperatura `Sparkfun TMP36` y de cinco leds de colores distintos cuya descripción se adjunta.

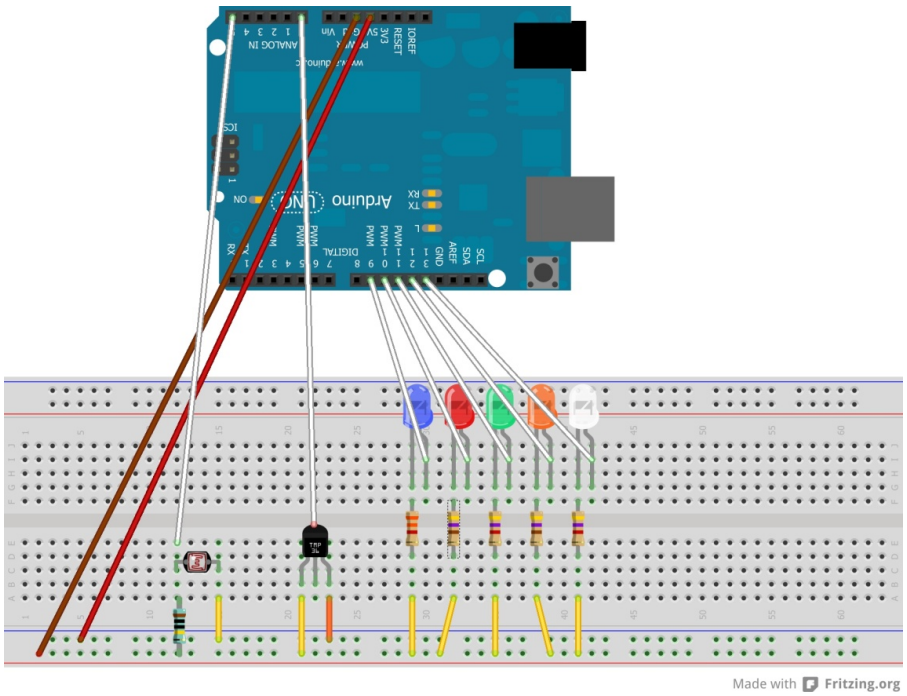

Figura 2-1. *Diagrama de la plataforma 1*

Es importante que los cinco leds sean de colores distintos. Dado el tamaño de una clase es muy difícil para los estudiantes distinguir los LEDS en función de su posición y la única manera de que puedan distinguirlos es si son de distintos colores.

Codigo	Ref	Descripción
LEDWC503B	WAN-CBADA151	5 mm blanco. IF= 25mA VF=3.2 V 9000mcd 15º
619-4937	L-7113SRD-H	5 mm difuso rojo. IF=20mA. VF=1.85V 1000mcd. 30º
262-2955	L-53SED	5mm difuso naranja. IF=20mA. VF=2V 800mcd. 60º
466-3548	L-7113QBC-D	5mm azul. IF=20mA. VF=3.5V 2200mcd. 15º
466-3510	L-7113VGC-H	5mm verde. IF=20mA. VF=3.7V. 14000mcd. 20º

Tabla 2-1. *Características de los LEDs utilizados*

Durante las demostraciones los estudiantes comentaron que a veces la intensidad de los LEDs era excesiva y decidimos calibrar su intensidad. Después de realizar un estudio con los estudiantes en el aula obtuvimos la siguiente calibración

Resistencias a utilizar con los distintos LEDs

- Led rojo difuso -> 470 ohmios
- Led naranja difuso -> 470 ohmios
- Led blanco Arduino -> 5 Kohmios
- Led azul -> 3 Khomios
- Led verde ultraeficiente -> 5 Khomios

Es de esperar que en otras aulas pueda ser necesario ajustar las resistencias.

3. Diseño 2

La segunda plataforma está diseñada para realizar prácticas más sofisticadas a medida que vaya avanzando el curso. Consta de un servomotor `Sparkfun ROB-09065`, un mini-altavoz `Sparkfun COM-07950` y un sensor de distancia `LV-MaxBotix-EZ1`. El sevomotor tiene acoplados dos LEDs que permiten a los estudiantes percibir el movimiento del servomotor desde la distancia.

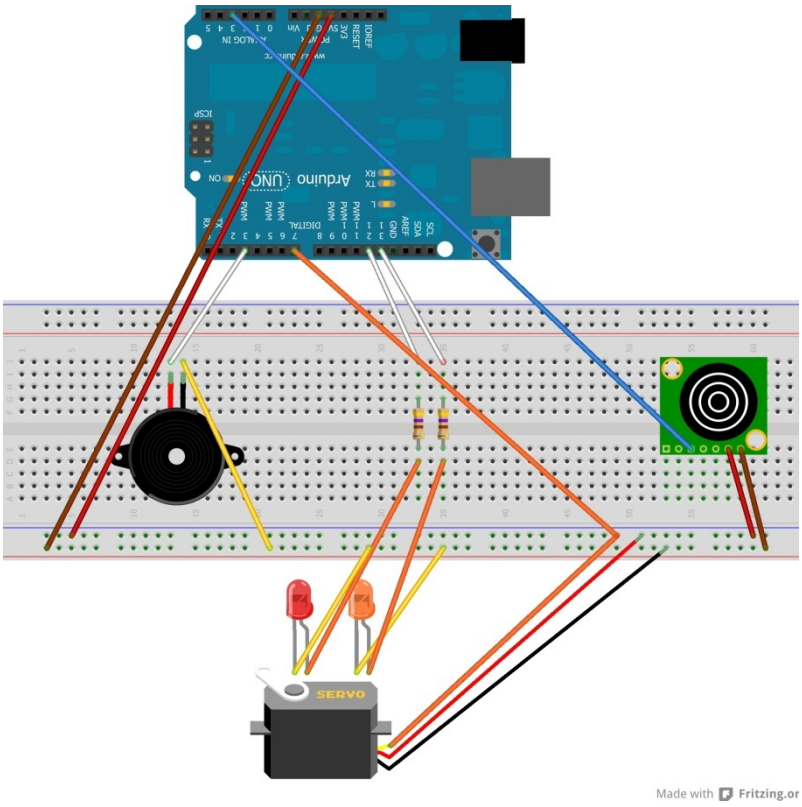

Figura 3-1. *Diagrama de la plataforma 2*

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

Tabla 3-1. *Características de los LEDs utilizados*

Resistencias a utilizar con los distintos LEDs

- Led rojo difuso -> 470 ohmios
- Led naranja difuso -> 470 ohmios

4. Librería software común a las dos plataformas

Ambas plataformas utilizan un conjunto de librerías desarrolladas durante el proyecto. Estas librerías contienen un conjunto de funciones con nombres en español que facilitan las explicaciones y la comprensión por parte de los alumnos.

A continuación se muestra el código de estas librerías.

```
/*
  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"

////////////////////////////////////
//Leer Voltaje
////////////////////////////////////

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

    return valor;
}
```

Figura 4-1. *Código de la librería común de Arduino (1ª parte).*

```
////////////////////////////////////
//Apagar LEDs
////////////////////////////////////

//Sirve para apagar un nico LED

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

//Sirve para apagar multiples 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);
        }
}

////////////////////////////////////
//Encender LEDs
////////////////////////////////////
//Sirve para encender un unico LED

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

//Sirve para encender multiples 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);
        }
}

////////////////////////////////////
//Mover Servo
////////////////////////////////////
//Mueve el servo a una cierta posición
//Se presupone que el servo esta conectado

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

Figura 4-2. *Código de la librería común de Arduino (2ª parte).*

```
////////////////////////////////////
//Pausar unos segundos
////////////////////////////////////

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

//////////////////////////////////
//Leer temperatura
//////////////////////////////////
//Funcion lee temp. Lee la temperatura mediante Arduino
//El sensor es el TMP36 del Arduino Starter Kit
//Entrada analog_pin: El pin que vamos a usar para leer la temperatura
//Salida: la temperatura
int lee_temperatura(int analog_pin){

 char str[256];

 //Realizamos 20 medidas para minimizar el error.
 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
 }

 //Hacemos la media y lo mostramos
 int temperatura = temp/num_medidas;
 return temperatura;

}

//////////////////////////////////
//Leer luminosidad
//////////////////////////////////
//Funcion luminosidad. Ve la luminosidad de la habitacion
//El sensor es el LDR-VT90N2
//Ver los pdf adjuntos
//Entrada analog_pin: El pin que vamos a usar para leer la luminosidad
//Argumento de salida
//El valor de luminosidad normalizado de 0 a 1
float lee_luminosidad (int analog_pin){
  float voltaje = lee_voltaje(analog_pin);
  voltaje = voltaje/1024.;
  return voltaje;
}

#endif // PHYSICAL_COMPUTING_DEMONSTRATIONS_H_
```

Figura 4-3. *Código de la librería común de Arduino (3ª parte).*

5. Demostración 1: Variables

Las demostraciones se han diseñado para cubrir distintos aspectos de la enseñanza de la programación. Para su efectiva utilización no basta con hacer las demostraciones en clase, es necesario involucrar a los alumnos en las mismas mediante votaciones, discusiones en pequeños grupos… (Crouch et al. 2004)

El objetivo de esta demostración es doble. Por un lado queremos presentar la tarjeta Arduino y explicar por qué hacemos este tipo de demostraciones. Por otro lado queremos enfrentar a los alumnos a algunas confusiones conceptuales bastante comunes. Mediante votaciones y discusiones en pequeños grupos motivamos a los estudiantes a que resuelvan sus dudas.

Una manera de plantear la discusión sería:

c = 3;
ejecutamos destellos(c); -> hay tres destellos

*a = c*2;*
ejecutamos destellos(a); -> hay seis destellos

c = 2;
discusión: ¿Cuántos destellos habrá? ¿4 o 6? ¿por qué?
Ejecutamos destellos(a)

*a = a*2 - 5*
discusión: ¿Cuántos destellos habrá? ¿6 o 7? ¿por qué?
Ejecutamos destellos(a)

El código que se debe utilizar en esta demostración se puede ver en la Figura 5-1.

```
/*
  Ejemplo de uso de 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() {

}

//Funcion apagar. Apaga dos leds (13 y 9) en el arduino
void apagar() {
  int led1 = 13;
  int led2 = 9;

  apaga_led(led1);
  apaga_led(led2);
}

//Funcion encender. Enciende dos leds (13 y 9) en el arduino
void encender() {
  int led1 = 13;
  int led2 = 9;

  enciende_led(led1);
  enciende_led(led2);
}

//Función destellos. Da num destellos
//Argumento de entrada
//num_destellos: el número de destellos a hacer

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);
}
```

Figura 5-1. *Código de la demostración 1.*

6. Demostración 2: Arreglos

En esta demostración vamos a trabajar con los arreglos. Vamos a usar un arreglo para almacenar una melodía y procederemos a trabajar con ella. Diversos autores (Moor 2010; Ruthmann et al. 2010) han mostrado que la música puede resultar muy efectiva en la presentación de conceptos como arreglos, bucles de repetición o funciones. En esta demostración es necesario utilizar la plataforma 2.

La melodía que hemos seleccionado es la marcha imperial de la banda sonora de la guerra de las galaxias. Nuestra experiencia indica que los estudiantes de esta generación la reconocen sin problemas y les llama la atención. Adjuntamos la partitura de la misma.

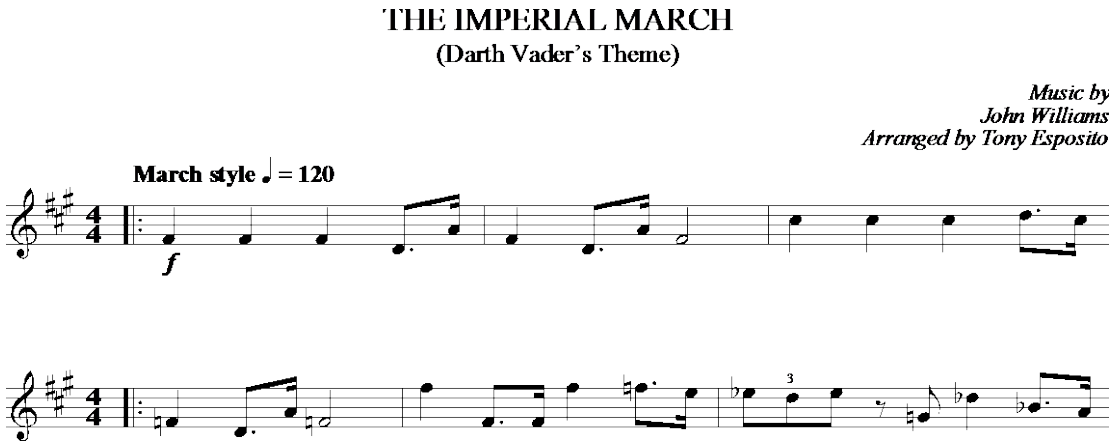

Figura 6-1. *Partitura de la marcha imperial (© 1980 WARNER-TAMERLANE PUBLISHING CORP. & BANTHA MUSIC, para uso académico)*

Una manera de plantear la discusión sería:

1. *Tocamos varias notas sentencia a sentencia*
2. *Tocamos toda la melodía usando el arreglo*
3. *Seleccionamos distintos subarreglos y hacemos que suenen*

El código que se debe utilizar en esta demostración se puede ver en la Figura 6-2.

```
/*
  Ejemplo de uso de arreglos
  Mirar el documento de instrucciones
  This example code is in the public domain.
 */

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

void setup() {

  // notas melodia:
  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;

  // duracion de la melodia:
  int duracion_nota[] = {negra, negra, negra, negra, corchea, negra, negra,
     corchea, blanca, negra, negra, negra, negra, corchea, negra,  negra, corchea,
blanca};


  //Tocamos una nota
  tocar (pin_altavoz, NOTA_SI_3, blanca);

  // Toca varias notas
  tocar (pin_altavoz, melodia, duracion_nota, 18);

}
```

Figura 6-2. *Código de la demostración 2 (1ª parte).*

```
void loop() {
}


//Toca una nota
void tocar (int pin_altavoz, int nota, int duracion) {

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

//Toca varias notas
void tocar (int pin_altavoz, int* melodia, int* duracion, int long_cancion) {

  // vamos nota a notas
  for (int pos = 0; pos < long_cancion; pos++) {

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

    // Incluimos una pausa para separar las notas
    int pausa = duracion[pos] * 1.2;
    delay(pausa);

    // Paramos el altavoz
    noTone(pin_altavoz);
   }
}
```

Figura 6-3. *Código de la demostración 2 (2ª parte).*

7. Demostración 3: Entradas y salidas

En esta demostración vamos a trabajar los conceptos de entrada y salida de un ordenador. Para ello vamos a distinguir entre las entradas y salidas del ordenador. En esta demostración es necesario utilizar la plataforma 1.

Una manera de plantear la discusión sería:

1) Leer temperatura

Ejecutar la función y ver cuál es la temperatura

Poner los dedos sobre el sensor unos 30 segundos

Ejecutar la función y ver que la temperatura ha subido

Ejecutarla al poco para ver que la temperatura ha bajado.

2) Escribe con leds

Decir que tienen que adivinar cómo se decide qué leds encienden

Ir haciendo pruebas hasta que lo adivinen

Discutir sobre distintas estrategias de adivinación (búsqueda binaria, sistemática..)

El código que se debe utilizar en esta demostración se puede ver en la Figura 7-1 y la Figura 7-2.

```
/*
  Ejemplo de uso de entradas y salidas
  Mirar el documento de instrucciones
  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);
}


//Función lee_temp_salida. Lee la temperatura mediante Arduino
//El sensor es el TMP36 del Arduino Starter Kit
//Salida: la temperatura
int lee_temp_salida(){

 char str[256];
 //El pin que vamos a usar para leer la temperatura
 int analog_pin = 0;

 //Realizamos 20 medidas para minimizar el error.
 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
 }

 //Hacemos la media y la devolvemos
 int temperatura = temp/num_medidas;
 return temperatura;
}
```

Figura 7-1. *Código de la demostración 3 (1ª parte).*

```
//Función lee_temp. Lee la temperatura mediante Arduino
//El sensor es el TMP36 del Arduino Starter Kit
//Ver los pdf adjuntos
void lee_temp(){

 char str[256];
 //El pin que vamos a usar para leer la temperatura
 int analog_pin = 0;

 //Realizamos 20 medidas para minimizar el error.
 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
 }

 //Hacemos la media y lo mostramos
 int temperatura = temp/num_medidas;
 sprintf(str, "La temperatura es    %d C",temperatura);
 Serial.println(str);

}
//Saca por led el número que introduzcamos en binario
//Máximo número 31
//Se utilizan los pines del 13 al 9
//13 menos significativo
//9 más significativo
//Para poder hacer el juego de adivinar el método es necesario
//que los leds tengan distintos colores.
//Argumento de entrada
//num: el número que vamos a pasar a binario

void escribe (int num) {

//Los leds con los que voy a trabajar
int rango_led [5]= {9,10,11,12,13};
int val;

for (int cont = 0; cont < 5; cont++){
    val = num%2; //Calculo el resto de dividir por dos
    num = num/2; //Calculo el cociente y lo asigno a num

    //Si la cifra en binario vale 1
    if (val == 1)
        enciende_led(rango_led[cont]); //enciendo el led
    //Si no
    else
        apaga_led(rango_led[cont]); //apago el led
}

}
```

Figura 7-2. *Código de la demostración 3 (2ª parte).*

8. Demostración 4: Condicionales

En esta demostración mostramos como se pueden utilizar las estructuras condicionales. En esta demostración se toma la medida de luminosidad ambiental. Si hay poca luz se encienden las luces, si hay bastante luz se apagan. Utilizaremos para ello la plataforma 1.

Una manera de plantear la discusión sería:

> *Mostrar el código condicional y mostrar su funcionamiento*
>
> *Variar las condiciones para que se enciendan y apaguen las luces*
>
> *Discutir los inconvenientes de esta implementación (sólo se ejecuta el condicional una vez) para introducir la necesidad de los bucles de repetición*

El código que se debe utilizar en esta demostración se puede ver en la Figura 8-1.

```
/*
  Ejemplo de uso de condicionales
  Mirar el documento de instrucciones
  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() {
}


//Función luminosidad. Ve la luminosidad de la habitación
//El sensor es el LDR-VT90N2
//Ver los pdf adjuntos
//Argumento de salida
//El valor de luminosidad normalizado de 0 a 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);
}
```

Figura 8-1. *Código de la demostración 4.*

9. Demostración 5: Repetición

Esta demostración es una continuación de la demostración cuatro. Extenderemos el código de la demostración anterior para incluir estructuras de repetición. De esta manera el funcionamiento durante cerca de un minuto y se pueden ver las variaciones que provocan la variación en luminosidad. Utilizaremos para ello la plataforma 1.

Una manera de plantear la discusión sería:

> *Recordar la demostración de los condicionales*
>
> *Incluir el bucle for y mostrar su funcionamiento*
>
> *Añadir una condición adicional usando un else if y la función encender_mitad y mostrar su funcionamiento.*

El código que se debe utilizar en esta demostración se puede ver en la Figura 9-2.

```
/*
  Ejemplo de uso de bucles
  Mirar el documento de instrucciones
  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);
}
}
```

Figura 9-1. *Código de la demostración 5 (1ª parte).*

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

//Función luminosidad. Ve la luminosidad de la habitación
//El sensor es el LDR-VT90N2
//Ver los pdf adjuntos
//Argumento de salida
//El valor de luminosidad normalizado de 0 a 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);
}
```

Figura 9-2. Continuación del *código de la demostración 5 (2ª parte).*

10. Demostración 6: Repetición avanzada

En esta demostración reforzamos los conceptos de los bucles de repetición trabajando con bucles anidados y con la estructura while. En este caso mostraremos como podemos hacer girar el servomotor (podemos presentarlo como un faro) utilizando bucles anidados. Presentaremos la utilidad de los bucles controlados por condición implementado una alarma de proximidad utilizando el servomotor y el sensor de distancia. Utilizaremos para ello la plataforma 2.

Una manera de plantear la discusión sería:

> *Presentar el código del faro y discutir su funcionamiento*
>
> *Presentar el código del faro con destellos y discutir su funcionamiento*
>
> *Presentar el código de la alarma de proximidad. Discutir ventajas y desventajas de utilizar while. Realizar la demostración.*

El código que se debe utilizar en esta demostración se puede ver en la Figura 10-1, Figura 10-2 y Figura 10-3.

```
/*
  Ejemplos avanzados de uso de bucleas
  Mirar el documento de instrucciones
  This example code is in the public domain.
 */
#include <PhysicalComputingDemonstrations.h>
#include <Servo.h>

void setup() {

//Cual es el nombre del Servo Motor
int pin_servo = 7;
Servo mi_servo;
mi_servo.attach(pin_servo);

//Cual es el nombre de los leds
int pin_rojo = 12;
int pin_naranja = 13;

////enciendo los leds
//enciende_led(pin_rojo);
//enciende_led(pin_naranja);
////Repito los giros tres veces
//for (int cont = 1; cont <= 3; cont++){
//    //Giro en un sentido
//    for (int angulo=0; angulo <= 180; angulo = angulo +5){
//        mover_servo(mi_servo,angulo);
//        para(.1);
//    }
//
//    //Giro en el otro sentido
//    for (int angulo=180; angulo <= 0; angulo = angulo - 5){
//        mover_servo(mi_servo,angulo);
//        para(.1);
//    }
//}
```

Figura 10-1. *Código de la demostración 6 (1ª parte).*

```
////Versión con destellos
////enciendo los leds
//enciende_led(pin_rojo);
//enciende_led(pin_naranja);
////Repito los giros tres veces
//for (int cont = 1; cont <= 3; cont++){
//
//    //Cambio el destello
//    enciende_led(pin_rojo);
//    apaga_led(pin_naranja);
//
//
//    //Giro en un sentido
//    for (int angulo=0; angulo <= 180; angulo = angulo +5){
//        mover_servo(mi_servo,angulo);
//
//        if (angulo > 90){
//          //Cambiar destello...
//          apaga_led(pin_rojo);
//          enciende_led(pin_naranja);
//        }
//        para(.1);
//    }
//
//    //Giro en el otro sentido
//    for (int angulo=180; angulo <= 0; angulo = angulo - 5){
//        mover_servo(mi_servo,angulo);
//        if (angulo > 90){
//          //Cambiar destello...
//          apaga_led(pin_rojo);
//          enciende_led(pin_naranja);
//        }
//        para(.1);
//    }
//}

//veo la distancia
float distancia = lee_distancia();

//Vigilando...
while (distancia > 30){
    Serial.println(distancia);
    //veo la distancia
    distancia = lee_distancia();
    para(.5);
}
```

Figura 10-2. *Código de la demostración 6 (2ª parte).*

```
//enciendo los leds
enciende_led(pin_rojo);
enciende_led(pin_naranja);

//Repito los giros tres veces
for (int cont = 1; cont <= 3; cont++){
    //Giro en un sentido
    for (int angulo=0; angulo <= 180; angulo = angulo +5){
        mover_servo(mi_servo,angulo);
        para(.1);
    }

    //Giro en el otro sentido
    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() {
}

//Función lee_distancia. Lee la distancia mediante Arduino
//El sensor es el LV-MaxSonar EZ1


float lee_distancia(){

 //El pin que vamos a usar para leer la distancia
 int analog_pin = 3;
 //Numero de medidas
 int num_medidas = 15;

 //Realizamos num_medidas medidas para estabilizar el resultado
 //Las medidas del sensor no son muy estables
 float distancia_inst = 0;
 float voltaje;


 for (int cont = 0; cont < num_medidas; cont++){
    voltaje = lee_voltaje(analog_pin); //Medida directa
    voltaje = voltaje/2; //En unidades normalizadas
    distancia_inst = distancia_inst + voltaje*2.54; //En cm

 }

 //Hacemos la media y lo mostramos
 float distancia = distancia_inst/num_medidas;
 return distancia;
}
```

Figura 10-3. *Código de la demostración 6 (3ª parte).*

Bibliografía

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