Detailed design of the lab sessions 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 proceed to describe the set of activities that have been developed for the lab sessions. We begin with a description of the designed platform and continue with a brief description of each demo and the necessary software.

The platform that has been developed for lab sessions is based on the Arduino board. In the case of lab sessions, it is necessary to mount the card and prototyping board on a common platform, reducing the occurrence of errors and malfunctions.

The platform designed for the lab sessions is based on the first platform used in the lectures. The reason is simple: our experience indicates that students are often left with many questions about the demos performed in lectures. We can take advantage of this unsatisfied curiosity if we introduce in lab sessions similar experiments to the ones given in lectures.

The platform we present consists of a `LDR-VT90N2` photoresistor, a `TMP36 Sparkfun` temperature sensor and five red LEDs (description attached).

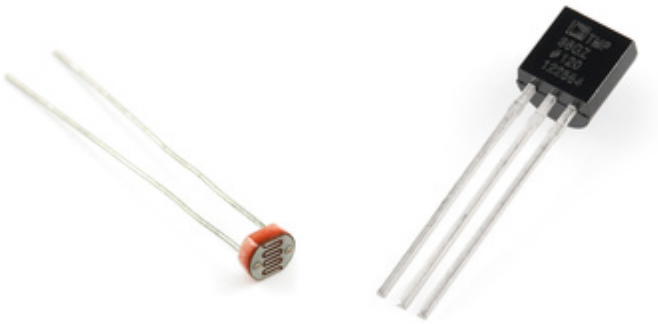

Figure 1-1. *`LDR-VT90N2` photoresistor and `TMP36 Sparkfun` temperature sensor.*

Code	Reference	Description
619-4937	L-7113SRD-H	5 mm diffused red. IF=20mA. VF=1.85V 1000mcd. 30º

Figure 1-2. *Characteristics of the LEDs used.*

Resistance used with each LED:

- Diffused red LED -> 470 ohms

- Green LED -> 5 Kohms

The design includes some aspects that facilitate teaching. We try to be consistent with the colours: red for power, brown for earth and yellow for the collection of information. One of the LEDs has a longer cable in blue colour, which allows to perform some experiments that show students that the underlying mechanics are simple.

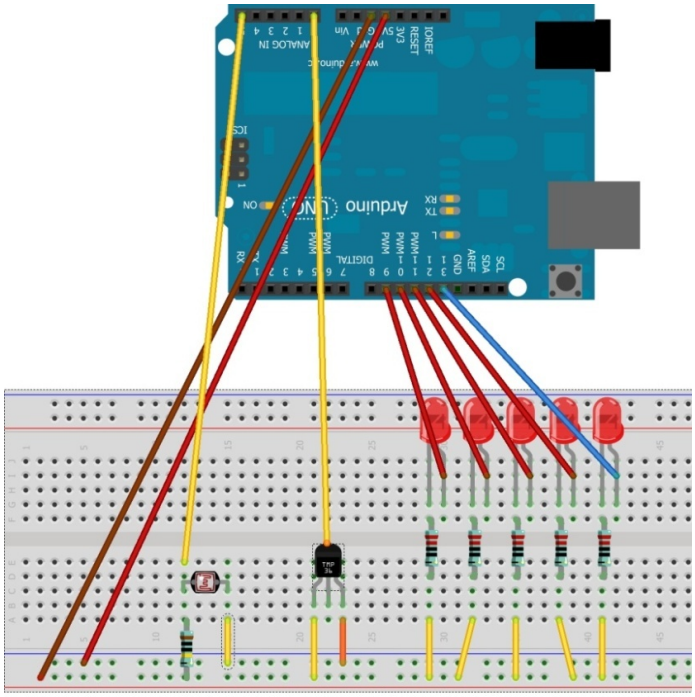

Figure 1-3. *Diagram of the platform.*

The exercises have been designed to present different concepts of programming and use of computers using the Arduino board. Our goal consists in students to see how what they have learned in the lectures allow them to program interesting things in real devices.

The design aims to help students at different stages of learning, also known as scaffolding (Sorva 2012). Although the activities are related, they were created to be as independent as possible. The teacher can choose -depending on the time available and the knowledge of their students- which activities to include in the lab session and which not.

It is assumed that students are already familiar with the Arduino board and programming using Arduino IDE. Their teacher may decide to present these principles in lectures, at the beginning of the lab session or in a previous lab session.

2. Exercise #1: Variables

We are going to start the lab session by turning LEDs on and off. First, we will turn on LED 10. Copy the following code, compile it and load it into Arduino.

```
/*
  Example of the use of LEDs
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {
enciende_led(10);
}

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

}
```

- Now try to modify the code to turn LED 11 instead of LED 10.

Let's proceed by turning on LEDs 9, 11 and 13 simultaneously.

```
 /*
  Example of the use of LEDs
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {
int leds[3] = {9,11,13};

enciende_led(leds,3);
}

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

}
```

- Now try to modify the code to turn LEDs 10, 11 and 12 simultaneously.

To turn off a LED connected to pin 9 after turning on LED 9, 11 and 13, the code must be:

```
 /*
  Example of the use of LEDs
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {
int leds[3] = {9,11,13};

enciende_led(leds,3);
para(1); // Stops the execution for a second
apaga_led(9);
}

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

}
```

- Now, try to modify the code to turn off LED 13.

To turn off LEDs connected to pins 9, 11 and 13, the code will be:

```
 /*
  Example of the use of LEDs
  This example code is in the public domain.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {
int leds[3] = {9,11,13};

enciende_led(leds,3);
para(1); // Stops the execution for a second
apaga_led(leds,3);
}

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

}
```

- Now, try to modify the code to turn on LEDs 9, 11 and 13 and turn off LEDs 10, 11 and 12. What happens?

At least one of the LEDs is connected to Arduino using the long cable. Disconnect this cable and try to turn on the LED. What happens?

Next, copy the following code and save it. What do you think it does?

```
 /*
  Example of the use of LEDs
  Program that executes a number of flashes.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {

int destellos = 4; // Number of flashes
int pinLed1 = 13;
int pinLed2 = 9;

for (int cont = 0; cont < destellos; cont++) {
   apaga_led(pinLed1);
   enciende_led(pinLed2);
   para(1);
   apaga_led(pinLed2);
   enciende_led(pinLed1);
   para(1);
}
   apaga_led(pinLed1);
   apaga_led(pinLed2);
}

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

}
```

Compile it and load it into Arduino. Does it do what you expected?

1. Modify the previous code to use all available LEDs instead of only two of them (like a Christmas tree, a clyon o the knight rider).
2. Modify the value for `para(1)` to make it flash quicker.
3. What will happen if we use `para()` instead of `para(1)`?

3. Exercise #2: Sensing the outside world

Once we have learned how to use a program to turn on and off LEDs, let's see how to perceive the outside world using electronic sensors. To do this, we will start by measuring the temperature of the room.

To read the temperature you must search for the analog pin connected to the sensor. You can see it on the diagram below; although it is better that you see it directly on the board that we have been given you. Your temperature sensor should be connected to *analog pin 0*.

Once we have identified the pin to which it is connected, we can begin to measure the temperature. If the sensor is connected to pin 0, then the code would be:

```
/*
  Example of usage of temperature and brightness sensors.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {

  Serial.begin(9600); // allows to see the output on the screen
}



void loop() {
  int pin_temperatura = 0;
  int temper = lee_temperatura(pin_temperatura);
  Serial.println(temper); // shows the temperatur in the serial monitor
  para(1);
}
```

To see the output you need to open the serial monitor window. It's in *Tools-> Serial Monitor*.

Note that the output of the function `lee_temperatura` will depend on the temperature of the room. Now, put your fingers on the black section of the temperature sensor. Wait a few seconds and repeat the temperature measurement. What happened? Describe the results you have obtained.

Now we proceed to measure the brightness of the room. To do this we will use a sensor that provides a measure between zero and one. Zero is total darkness and one is maximum brightness.

To read the brightness you must search for the analog pin connected to this new sensor. You can see it on the diagram below; although it is better that you see it directly on the board that we have been given you. Your brightness sensor should be connected to *analog pin 5*.

Once you have identified the pin to which it is connected, we can begin to measure the brightness. If the sensor is connected to pin 5, then the code will be:

```
/*
  Example of usage of temperature and brightness sensors.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {

  Serial.begin(9600); // allows the output on the screen
}


void loop() {
  int pin_luminosidad = 5;
  float luz = lee_luminosidad(pin_luminosidad);
  Serial.println(luz); // shows the brightness in the serial monitor
  para(1);
}
```

Note that the output of the function `lee_luminosidad` will depend on the brightness of the room. Now, cover the sensor with your fingers and repeat the brightness measurement. Is the new value higher of lower than the previous one? Comment on the results obtained.

4. Exercise #3: Additional activities

Finally, a set of exercises is proposed to the students to strengthen the acquired knowledge. Likewise, it is expected that students could imagine possible real applications using what they have learned so far.

1. Modify the code of the function `encendido_automatico` to make it work properly. Then describe what it does and how did you implement it.

```
/*
  Example of usage of temperature and brightness sensors.
 */

#include <PhysicalComputingDemonstrations.h>

void setup() {
}


void loop() {

// Program encendido_automatico
// Turn on the LEDs if it is dark
// If there is light, turn them off
//-------------------------------------

    // See the brightness
    float luz = lee_luminosidad(5);
    int leds[5] = {9,10,11,12,13};

    // If it is lower tan a given value,
    // then turn on the mappost
    if (luz < 0.9)
      enciende_led(leds,5);
    else // Turn off the lamppost
      enciende_led(leds,5);

  para (.3); // stop a few thenth of a second before repeating
}
```

2. Modify the program we used to make flashes and make the flashing frequency depend on the brightness of the environment (less brightness will produce higher frequency of flashes).
3. Analyse the code of the function `escribe` and indicates what it does. To do it you should state its function and how to implement it.

```
// 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
 }

}
```

The function is called using the following command: `escribe(16)`

4. Write a function called `escribe_temperatura` that uses the function `lee_temperatura` and `escribe`. This new function reads the temperature from the temperature sensor and writes it in binary using the available LEDs. Create an alternative version called `lecturaContinua` that shows the temperature using the LEDs continuously.

Bibliography

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