Programa del curso de Arduino

Víctor Pérez Domingo
victorperezdomingo@gmail.com

Febrero 2017

Índice

1. Curso **4**
1.1. Objetivo del bloque 1 . . . 4
1.2. Objetivo del bloque 2 . . . 4

2. Iniciación: Clase 1 **5**
2.1. Objetivo del bloque . . . 5
2.2. ¿Que es Arduino? (30 min) . . . 5
2.2.1. Descripción de la placa . . . 5
2.2.2. Arduino IDE . . . 6
2.3. Ejemplo Basics/Blink.ino (30min) . . . 8
2.3.1. LED BUILTIN . . . 8
2.3.2. LED Externo: . . . 8
2.3.3. Varios LED al mismo tiempo . . . 9
2.3.4. Ejemplo Digital/BlinkWithoutDelay.ino . . . 9
2.4. Ejemplo Digital/Button.ino (30 min) . . . 10
2.4.1. Simple . . . 10
2.4.2. Ejemplo Digital/DigitalInputPullup.ino . . . 10
2.4.3. Ejemplo Digital/Debounce.ino . . . 11
2.5. Extra: Blink y Button al mismo tiempo (30 min) . . . 12

3. Iniciación: Clase 2 **12**
3.1. Comunicación serie (30 min) . . . 12
3.1.1. Ejemplo Basics/DigitalReadSerial.ino . . . 12
3.1.2. Ejemplo Communication/SerialEvent.ino . . . 13
3.1.3. Ejemplos Strings . . . 13
3.2. Librerias (15 min) . . . 13
3.2.1. Instalación de librerías: Botón . . . 13
3.3. Dispositivos (45 min) . . . 13
3.3.1. Sensor de presencia . . . 13
3.3.2. Sensor de distancia . . . 13
3.3.3. Pantalla LCD . . . 13
3.4. Dudas generales (30 min) . . . 13

4. Intermedio: Clase 1 **13**
4.1. Objetivo del bloque . . . 13
4.2. Creación de librerías: Botón avanzado (30 min) . . . 14
4.2.1. Anti rebotes . . . 14
4.2.2. CallBacks . . . 14
4.3. Técnicas de programación: FSM (60 min) . . . 14
4.3.1. Teoría . . . 14
4.3.2. Libreria . . . 14
4.4. Semaforo . . . 14
4.4.1. Teoría . . . 14
4.4.2. Libreria . . . 14

5. Intermedio: Clase 2 **14**
5.1. Comunicación inalámbrica (30 min) 14
5.2. Dispositivos remotos (30 min) 14
5.3. Aplicar todo lo aprendido (60 min) 14

1. Curso

1.1. Objetivo del bloque 1

Al completa este bloque se obtienen los siguientes conocimientos:

- Control de pines Digitales
- Comunicación Serie
- Librerías
- Control de dispositivos

1.2. Objetivo del bloque 2

Al completa este bloque se obtienen los siguientes conocimientos:

- Creación de librerías para Arduino
- Programar utilizando la técnica FSM
- Comunicación inalámbrica por radio

2. Iniciación: Clase 1

2.1. Objetivo del bloque

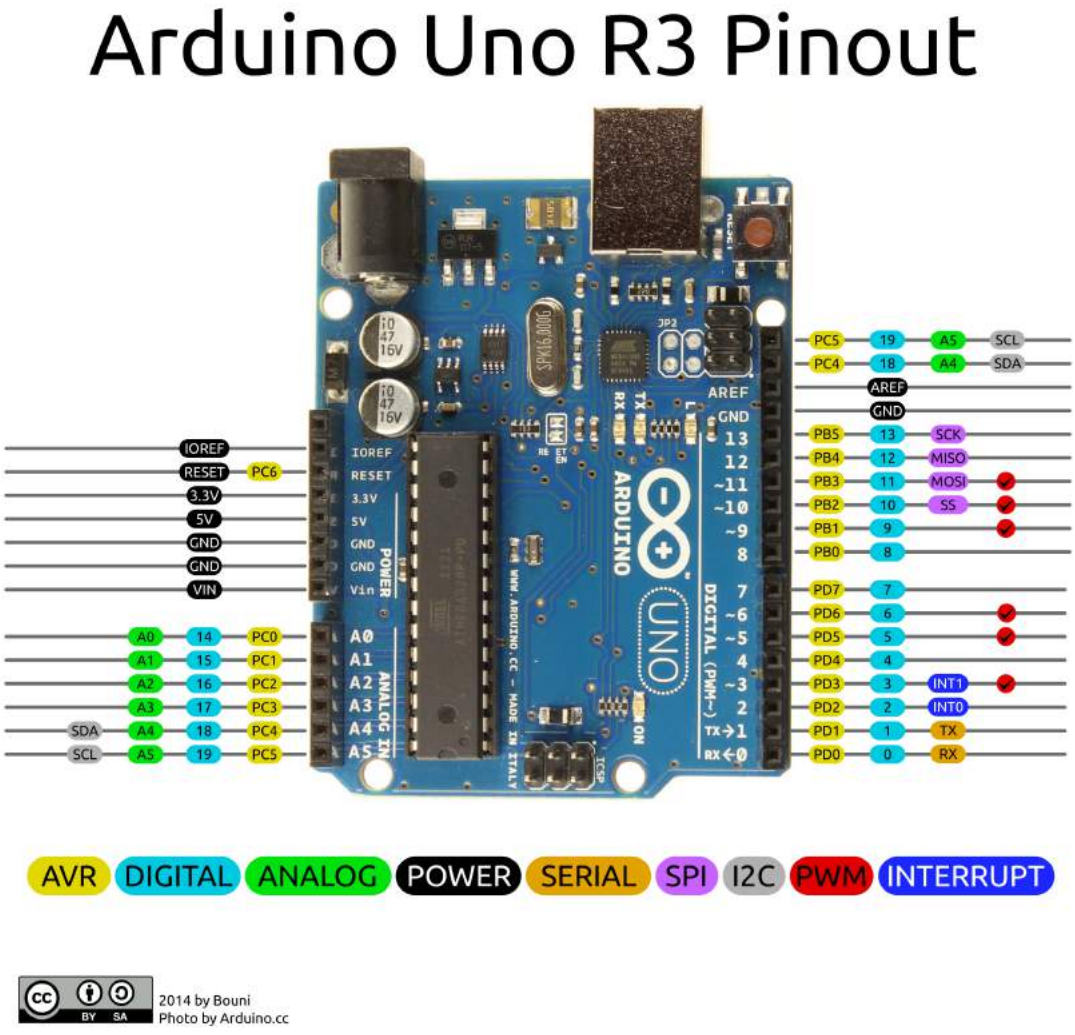
Al completa este bloque se obtienen los siguientes conocimientos:

- Control de pines Digitales
- Comunicación Serie
- Librerías
- Control de dispositivos

2.2. ¿Que es Arduino? (30 min)

Arduino es una placa de desarrollo libre que nos permite interactuar con entradas como botones, sensores de luz... y transformarlo en una salida como un led o el motor de una persiana. Con un poco de imaginación podremos lograr automatizar cualquier proceso que nos propongamos.

2.2.1. Descripción de la placa



El modelo Arduino UNO R3 es el modelo mas popular y en el que se suelen basar todos los ejemplos que se encuentran en la red.

Pines Digitales: Todos los pines de Arduino se pueden comportar como un pin Digital aunque se suelen denominar únicamente a los pines que van desde el pin 0 al pin 13. Pueden actuar como INPUT (entradas) o OUTPUT (salidas) y se caracterizan por tener únicamente dos estados, HIGH (activado) o LOW (desactivado).

Pines Analógicos: Estos pines empiezan por la letra A y van desde A0 hasta A5. Estos pines son capaces de medir un valor de tension entre 0 y 5 de manera precisa, lo que nos va a permitir obtener medidas de sensores analógicos como sensores de luz LDR

Pines con funciones especiales: En el esquema se puede apreciar que algunos pines tienen funciones alternativas,. Cuando utilicemos una de estas características no podremos utilizar el funcionamiento digital del pin.

Led 13: Hay un led integrado en la placa listo para ser utilizado en el pin 13. lo único que hay que hacer para hacerlo funcionar en configurar este pin como OUTPUT y elegir si queremos encender o apagar el led con las funciones HIGH y LOW.

Puerto Serie: En los pines 0 y 1 se ven las letras RX TX que son los pines básicos de comunicación serie. Es recomendable no utilizar estos pines durante nuestros programas ya que son los mismos que se utilizan para programar la placa o para comunicarse con el ordenador.

Puerto SPI: los pines del 10 al 13 tienen los nombres SCK, MISO, MOSI y SS que identifican los pines de comunicación SPI. Si vamos a utilizar un Shield de Arduino que utilize estos pines, como el Ethernet Shield, no podremos utilizarlos como pines digitales.

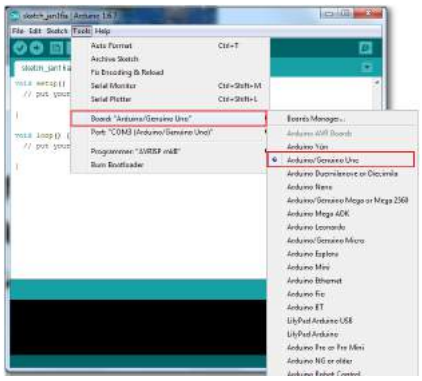
2.2.2. Arduino IDE

Existen muchas entornos de desarrollo que nos van a permitir programar Arduino como Visual Studio o Atmel Studio. Aun así, para principiantes lo mejor es utilizar Arduino IDE por ser muy simple e intuitivo.

Instalación: Lo primero es descargar el software desde la página web www.Arduino.cc. Cada uno deberá descargar la version acorde a su sistema operativo. La instalación es un proceso sencillo e idéntico a la instalación de cualquier otro programa.

Drivers: Para las placas Arduino UNO R3 oficiales que cuentan con el chip oficial para la comunicación con el ordenador por lo que no suele suponer ningún problema de drivers. Si utilizas una placa no oficial es posible que utilicen otro chip de comunicación por lo que tendrás que buscar los drivers. Normalmente en la carpeta de instalación de Arduino IDE encontramos una carpeta con los drivers mas comunes.

Configuración: Hay que configurar el IDE para programar la placa UNO R3, hay que acceder al menú de herramientas¿placas y seleccionar Arduino UNO R3. Luego hay que seleccionar el puerto en el que la placa está conectada al ordenador.



Ejemplos: Arduino IDE cuenta con un montón de ejemplos, puedes acceder a ellos en el menú Archivo>Ejemplos. Lo primero que vamos a hacer es cargar el programa de ejemplo "Basics>Blink" .

Cargar un programa: Si todo esta configurado correctamente, solo hay que pulsar la flecha en la parte superior izquierda para cargar el programa a la placa Arduino y empezar a ver como funciona.

2.3. Ejemplo Basics/Blink.ino (30min)

2.3.1. LED BUILTIN

Este es el programa mas sencillo de Arduino y por lo tanto es el primero que todo el mundo tiene que hacer funcionar. Las placas Arduino vienen siempre programadas de fabrica con el ejemplo Blink.

```
// La funcion setup se ejecuta una vez al conectar la placa
void setup() {
  // inicializa el pin LED_BUILTIN como SALIDA,
  // en la placa UNO es el pin 13.
  pinMode(LED_BUILTIN, OUTPUT);
}

// la funcion loop se ejecuta de manera ciclica para siempre.
void loop() {
  // Enciende el LED_BUILTIN
  digitalWrite(LED_BUILTIN, HIGH);
  // Espera 1 segundo
  delay(1000);
  // Apaga el LED_BUILTIN
  digitalWrite(LED_BUILTIN, LOW);
  // Espera 1 segundo
  delay(1000);
}
```

El programa se divide en dos funciones básicas, **setup** y **loop**. Todos los programas de Arduino tienen ambas funciones.

pinMode: Es una función con dos parámetros, el primero es el numero del pin el segundo indica el modo en el que va a trabajar el pin, OUTPUT para salida, INPUT para entrada e INPUT_PULLUP para activar el pullup interno del pin.

digitalWrite: Es una función con dos parámetros, el primero es el el numero del pin y el segundo puede ser HIGH para encender y LOW para apagar.

LED_BUILTIN: Es un macro definido en las librerías de Arduino que contiene el valor del numero del pin en el que esta conectado el led integrado de la placa. en Arduino UNO es el 13.

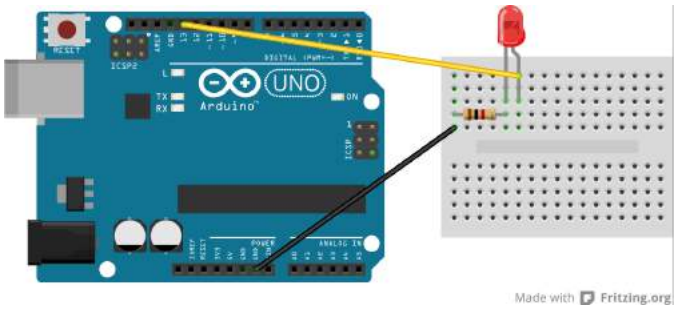
delay: Es una de las funciones mas populares entre los principiantes. te permite parar la ejecución del programa por un numero determinado de milisegundos.

2.3.2. LED Externo:

Para ir ganando soltura, lo primero es realizar pequeños cambios en este programa y ver que responden como nosotros esperamos. Un cambio sencillo es sustituir la variable LED_BUILTIN

por un numero indicando otro pin, donde conectaremos un led externo para comprobar que esta funcionando correctamente.

Conectar led externo:



En este ejemplo he añadido una variable llamada led con el valor 9. por lo tanto tendremos que conectar nuestro led en el pin 9.

```
int led = 9;
// La funcion setup se ejecuta una vez al conectar la placa
void setup() {
  // inicializa el pin led como SALIDA,
  //en la placa UNO es el pin 13.
  pinMode(led, OUTPUT);
}

// la funcion loop se ejecuta de manera ciclica para siempre.
void loop() {
  // Enciende el led
  digitalWrite(led, HIGH);
  // Espera 1 segundo
  delay(1000);
  // Apaga el led
  digitalWrite(led, LOW);
  // Espera 1 segundo
  delay(1000);
}
```

2.3.3. Varios LED al mismo tiempo

Si ya has conseguido hacer funcionar los dos ejemplos anteriores no debería ser muy difícil hacer un programa que parpadee dos leds al mismo tiempo. ¡Es el momento de experimentar!

2.3.4. Ejemplo Digital/BlinkWithoutDelay.ino

Es un buen momento para echar un vistazo al ejemplo Blink Witout Delay. Nos puede ayudar a hacer un programa que haga parpadear dos leds al mismo tiempo con frecuencias distintas.

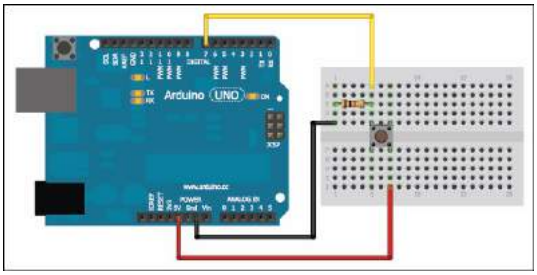
2.4. Ejemplo Digital/Button.ino (30 min)

2.4.1. Simple

Ya sabemos la forma de encender y apagar un led en función del tiempo. el siguiente paso es controlar este efecto utilizando un botón.

Busca el ejemplo Digital>Button.

```
// Configuracion de pines:
// El numero del boton
const int buttonPin = 2;
// El numero del led
const int ledPin =  13;

// variable temporal:
// Guarda el estado del boton
int buttonState = 0;

void setup() {
  // Configura el led como salida:
  pinMode(ledPin, OUTPUT);
  // Configura el boton como entrada
  pinMode(buttonPin, INPUT);

}

void loop() {
  // obtiene el valor del boton
  buttonState = digitalRead(buttonPin);

  // Comprueba si el boton esta pulsado.
  if (buttonState == HIGH) {
    // Enciende el led:
    digitalWrite(ledPin, HIGH);
  } else {
    // Apaga el led:
    digitalWrite(ledPin, LOW);
  }
}
```



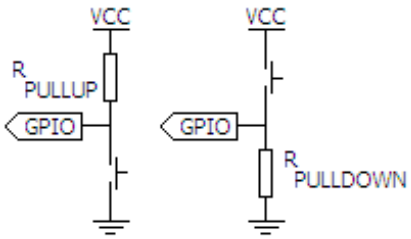
Ejemplo de conexión en el pin 7

digitalRead: Esta función tiene un único parámetro para identificar el numero del pin. Devuelve HIGH cuando el pin esta conectado a 5V o LOW cuando esta conectado a GND.

Este ejemplo no es la manera optima de leer un botón. recomiendo pasar al ejemplo siguiente ya que es mucho mejor.

2.4.2. Ejemplo Digital/DigitalInputPullup.ino

En el ejemplo anterior estábamos utilizando una configuración Pull-down utilizando una resistencia externa. En este ejemplo vamos a utilizar una resistencia interna de Arduino para obtener una configuración Pull-up.



Pull-up vs Pull-down

```
//Ejemplo modificado
void setup() {
  //Configurar el pin 2 como pullup
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  // int sensorVal = digitalRead(2);
  // Lectura invertida
  int sensorVal = !digitalRead(2);
  if (sensorVal == HIGH) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

Hay que tener en cuenta que, al usar una configuración Pull-up, la lógica del botón queda invertida. una rápida solución es invertir la lectura del pin para que el resto del código siga siendo el mismo.

2.4.3. Ejemplo Digital/Debounce.ino

Este ejemplo incluye dos nuevos conceptos que vamos a analizar por separado.

Detección de flancos: En vez de fijarnos en si un botón esta pulsado o no, suele ser mucho más útil fijarse en los momentos de cambio de estado (flancos). Esto nos va a permitir encender un led al pulsar un botón y apagarlo al volver a pulsar el botón.

Para detectar un flanco necesitamos dos variables, El estado actual del pin y el estado la ultima vez que comprobamos el pin. Al momento en el que se pulsa el botón se le denomina Flanco Positivo (FP) y al momento en el que se suelta el botón se le denomina Flanco Negativo (FN). Utilizando estas dos variables podemos obtener la siguiente tabla:

	Anterior	Actual
OFF	0	0
FP	0	1
ON	1	1
FN	1	0

```
int boton = 2;
```

```
bool actual; //estado actual
bool anterior; //estado anterior

bool FP = false; //Flanco positivo
bool FN = false; //Flanco negativo

void setup() {
  pinMode(boton, INPUT_PULLUP);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  //Reiniciamos variables
  FP = false;
  FN = false;
  //Guardamos estado anterior
  anterior = actual;
  actual = digitalRead(boton);
  if (actual == HIGH & anterior == LOW) {
    FP = true;
  }
  if (actual == LOW & anterior == HIGH) {
    FP = true;
  }

  if (FP) {
    //Invierte el estado del led
    digitalWrite(LED_BUILTIN,!digitalRead(LED_BUILTIN));
  }
}
```

Eliminación de rebotes:

2.5. Extra: Blink y Button al mismo tiempo (30 min)

3. Iniciación: Clase 2

3.1. Comunicación serie (30 min)

La comunicación con el ordenador es una de las mejores maneras de comprobar que nuestro programa está funcionando correctamente. Vamos a aprender lo básico para obtener una comunicación bidireccional.

3.1.1. Ejemplo Basics/DigitalReadSerial.ino

3.1.2. Ejemplo Communication/SerialEvent.ino

3.1.3. Ejemplos Strings

3.2. Librerias (15 min)

3.2.1. Instalación de librerías: Botón

Existe una gran cantidad de código listo para ser usado en nuestros programas que nos van a facilitar mucho las cosas. Vamos a buscar algunas de las librerías que existen en Arduino y vamos a aprender a instalarlas.

3.3. Dispositivos (45 min)

Es el momento de introducir dispositivos mas complejos que los botones y los leds.

3.3.1. Sensor de presencia

3.3.2. Sensor de distancia

3.3.3. Pantalla LCD

3.4. Dudas generales (30 min)

4. Intermedio: Clase 1

4.1. Objetivo del bloque

Al completa este bloque se obtienen los siguientes conocimientos:

- Creación de librerías para Arduino
- Programar utilizando la técnica FSM
- Comunicación inalámbrica por radio

4.2. Creación de librerías: Botón avanzado (30 min)

Vamos a convertir el código para leer botones que hemos creado antes en una librería para poder reutilizarlo en futuros proyectos.

4.2.1. Anti rebotes

4.2.2. CallBacks

4.3. Técnicas de programación: FSM (60 min)

Es el momento de aprender una técnica de programación para mantener nuestros programas ordenados.

4.3.1. Teoría

4.3.2. Libreria

4.4. Semaforo

4.4.1. Teoría

4.4.2. Libreria

5. Intermedio: Clase 2

5.1. Comunicación inalámbrica (30 min)

Objetivo final, mandar información por radio a otra persona.

5.2. Dispositivos remotos (30 min)

5.3. Aplicar todo lo aprendido (60 min)