SESION 4-
Circuito con múltiples LEDs

4.1 Objetivos

- Conocer las Iteraciones en C++, Instrucción for.
- Primeras variables en C++.
- Circuito con múltiples LEDs.
- Aprendiendo a programar…programando.

4.2 Material requerido.

Arduino Uno o similar.
Una **Protoboard**.
8 x **diodos LED**.
Una **resistencia** de 330 Ohmios.
Algunos **cables** de Protoboard.

4.3 Un circuito con varios LED: Iteraciones For

En las sesiones anteriores vimos como gobernar que un diodo LED externo. Si quisiéramos montar un circuito que tuviera 8 LEDs y en el que la luz se desplazara de uno a otro, una posibilidad sería repetir varias veces las mismas secuencias de instrucciones que ya conocemos.

Por ejemplo si conectamos distintos LEDs a distintos pines digitales de Arduino, deberíamos declararlo en nuestra Función de setup() que podría ser:

```
void setup()
  {
       // initialize the digital pins as an output
      pinMode( 13, OUTPUT) ;
      pinMode( 12, OUTPUT) ;
      pinMode( 11, OUTPUT) ;
      ……
      pinMode( 6, OUTPUT) ;
  }
```

Y a su vez nuestro loop() debería repetir tantas veces como LEDs tengamos el juego de encender y apagar cada uno de los LEDs en secuencia desde el pin 13 hasta el 6.

Esta solución es la que podríamos describir como de fuerza bruta, pero no es muy elegante, es trabajosa y probablemente cometeríamos más de un error al escribirla, porque las personas tendemos a equivocarnos haciendo tareas repetitivas aburridas (y esta lo es mortalmente, imaginad un circuito de de 16 LEDs).

En cambio los ordenadores no se aburren y además C++ nos ofrece un medio cómodo de indicarle que debe repetir algo un número definido de veces. Este medio es la instrucción For que podemos usar en combinación con una variable.

- Una variable es un contenedor que puede tomar varios valores, en nuestro caso aceptará todos los valores entre 6 y 13.
- C++ nos exige declarar el **tipo** de las variables antes de usarlas. En nuestro caso usaremos el tipo entero que se escribe **int** para indicar que esta variables es **numérica** y entera, sin decimales.
- Iremos viendo que existen otros tipos de variables. Volveremos sobre este tema en próximas sesiones.

Así por ejemplo, para inicializar en nuestro setup() los pines desde el 13 hasta el 6 como salidas (requerido por nuestro Arduino) podríamos usar la instrucción for de la siguiente manera:

```
void setup()
  {
    int i = 0 ; // Inicializamos la variable i como un entero
    for ( i = 6 ; i < 14 ; i++)
        pinMode( i , OUTPUT) ;
  }
```

Aunque la sintaxis parece complicada al principio, uno se acostumbra con rapidez. Aquí lo importante es que for necesita 3 parámetros separados por un carácter de punto y coma.

Estos parámetros son y en éste orden:

- Una **variable** que ira tomando valores según una cierta regla, y a la que asignamos un valor inicial. En este caso: **i = 6** .
- El ciclo continúa mientras se cumpla esta condición. En nuestro caso mientras la i sea menor que 14, o sea hasta el 13: i <14
- Como cambia la variable en cada iteración. En nuestro caso **i++** que es pedirle a C++ que incremente en uno la variable i, al final de cada iteración.

Con el mismo criterio podríamos escribir la función loop() así Descargar:

```
void loop()
  {
    int i = 0 ; // Inicializamos la variable i como un entero
    for ( i = 6 ; i < 14 ; i++)
        {
          digitalWrite( i , HIGH) ;
          delay (500) ;
          digitalWrite( i , LOW);
          delay (500) ;
        }
  }
```

En la sesión 3 el código era muy similar excepto en que escribíamos el valor 13 para el único pin que tenía un LED conectado. Aquí asignamos el pin con una variable i , que va tomando los valores de 6 hasta el 13 para el pin.

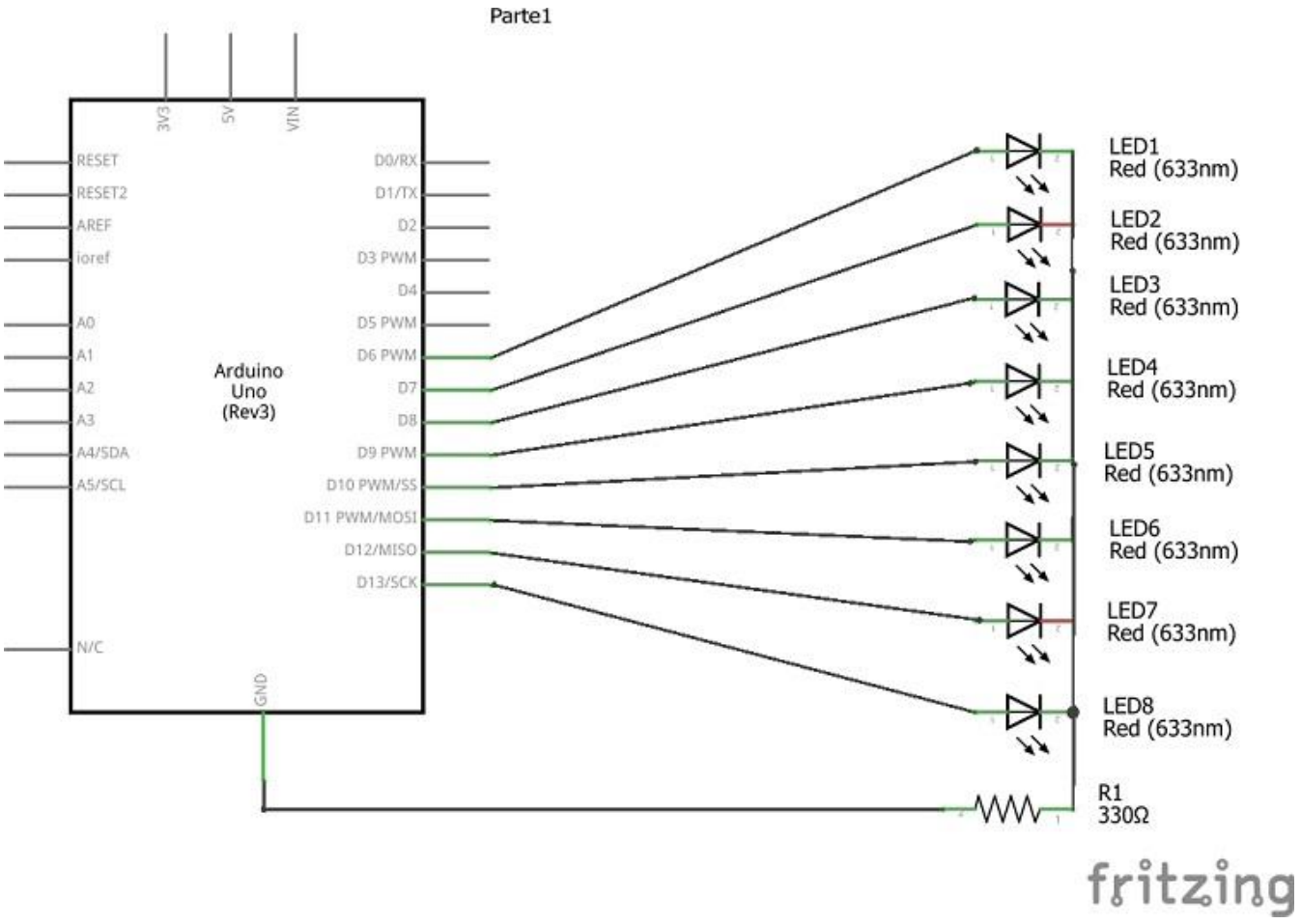
- *Nótese que la instrucción for no lleva un punto y coma al final. Esto es porque se aplica al bloque de instrucciones que le siguen entre llaves, como es el caso del loop() La iteración realiza las cuatro instrucciones que siguen a la línea del for, porque están dentro de un bloque de instrucciones.*
- Las instrucciones que se aplican a bloques de código, no llevan punto y coma al final.
- *En el caso de particular de que el bloque lleve una única línea de código, las llaves pueden ser omitidas, como en el caso de la instrucción for en la función setup() de arriba.*

4.4 Esquema electrónico del circuito

El esquema del circuito es muy similar al de la sesión 3, salvo por el hecho de que colocamos en la Protoboard 8 LEDs.

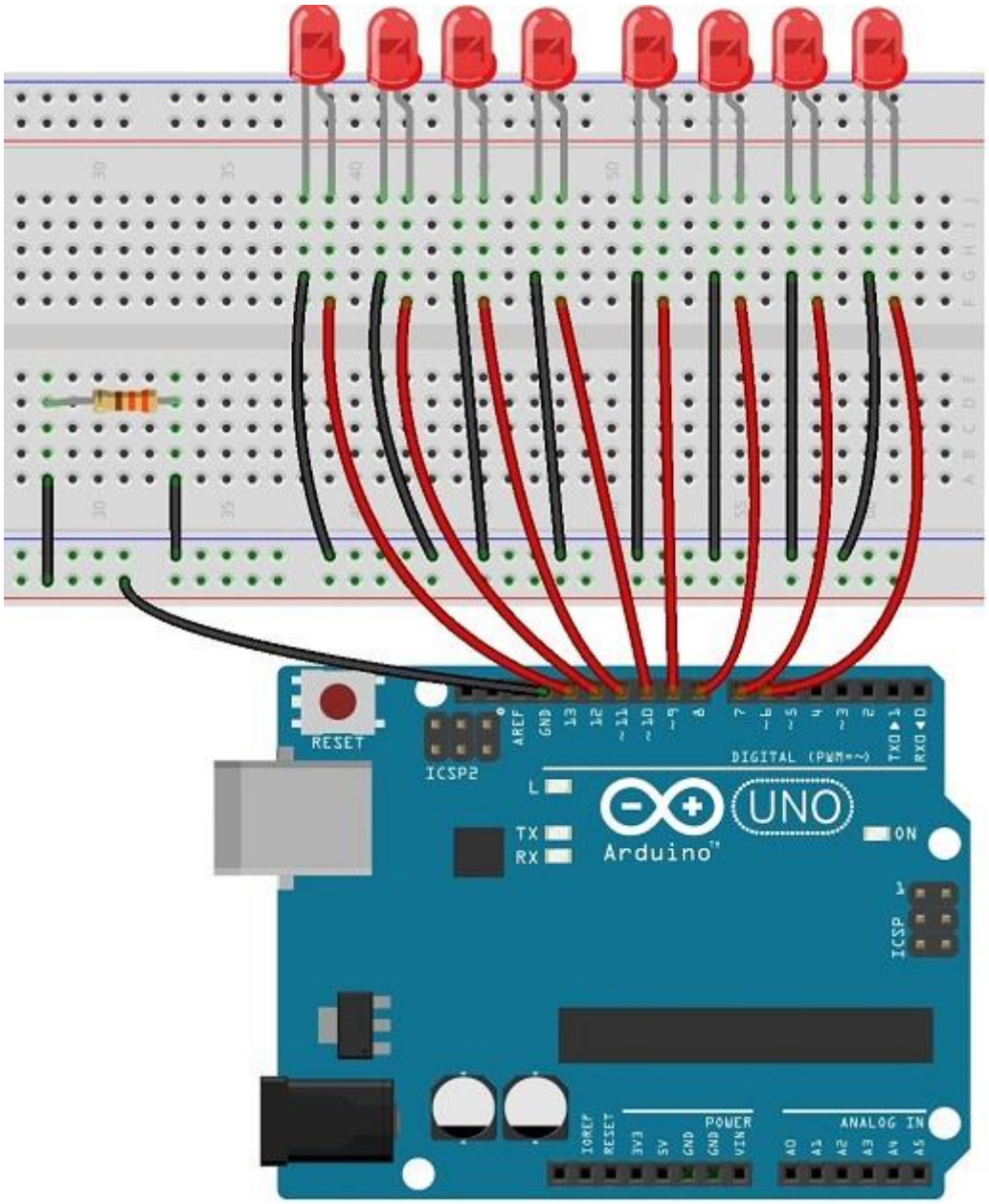
La única novedad es que dado que la función de la resistencia es limitar la intensidad de la corriente que circula por el circuito, y puesto que todos los diodos tienen masa común, basta una única resistencia entre este punto y Ground.

Cuando nuestro programa levante el pin correspondiente a valor a HIGH, se cerrar el circuito iluminándose el LED asociado.



Con este circuito, y con el programa 4.1 descrito en las páginas anteriores, tendremos un efecto de luces similar al del “coche fantástico” (O de los Zylon para los aficionados a la ciencia ficción).

A continuación incluimos un esquema de conexión del circuito en una protoboard.



En general, se considera buena costumbre (la recomendamos), montar los circuitos que veamos a partir del esquema electrónico del mismo, más que a partir del diagrama de conexiones de la Protoboard.

- *La razón es que con el esquema, la comprensión del circuito es completa y se evita la tentación de copiar la práctica sin necesidad de entenderla.*
- *Además, el diagrama electrónico del circuito es su completa descripción y suele resultar más sencillo comprender la función del mismo. En cambio a medida que los circuitos se hacen más complejos, comprender su función desde un esquema de Protoboard puede complicarse mucho, y peor aún llevar a una interpretación errónea.*

4.5 Variantes del programa con el mismo circuito

Este montaje nos permite jugar con las luces y se presta a varios programas diferentes para conseguir distintos efectos.

Por ejemplo, con el programa anterior 4.1, el efecto no es exactamente el del coche fantástico porque cuando acabamos de iterar el for, el programa vuelve a empezar desde el principio, lo que hace que la luz salte desde el pin 6 hasta la del pin 13.

Así pues ¿ Podríamos hacer que la luz rebotara ? Pensadlo un poco.

Desde luego que sí, bastaría con usar dos ciclos for, similar a lo siguiente Descargar:

```
void loop() // Prog_4_2
   {
      for ( int i = 6 ; i < 14 ; i++) // Definimos la variable i sobre la marcha
          {
             digitalWrite( i , HIGH) ;
             delay (500) ;
             digitalWrite( i , LOW);
             delay (500) ;
          }
      for ( int i = 12 ; i >6 ; i--) // Definimos la variable i sobre la marcha
          {
            digitalWrite( i , HIGH) ;
            delay (500) ;
            digitalWrite( i , LOW);
            delay (500) ;
           }
   }
```

- El primer ciclo for hace que las luces se encienda en secuencia desde la 6 hasta la 13. El segundo bucle entra a continuación empezando con la luz 12 (para no repetir la 13) y finalizando con la 7(para no repetir la 6), y vuelta a empezar.
- En el segundo bucle hemos hecho una cuenta atrás diciéndole a la variable i que se **decrcmentara** en uno en cada iteración mediante la instrucción i– .
- También nos hemos aprovechado de que C++ nos permite definir **variables sobre la marcha** dentro de la propia instrucción **for**, sin necesidad de dedicarle una línea completa a la declaración e inicialización.

Otra variante seria, hacer un efecto de ola en al que las luces subieran dejando encendidos los LEDs previos hasta alcanzar el máximo y ahora descender apagando los LEDs superiores. Os recomendamos intentar resolver el problema como desafío, antes de buscar una solución.

- *Programar es en parte aprender las instrucciones de un lenguaje (la parte fácil), y otra más difícil que es aprender a resolver los problemas de un modo que nos permita darle instrucciones a un ordenador para que lo lleve a cabo.*
- *Estos procedimientos secuenciales de cómo resolver un cierto tipo de problemas es lo que se conoce como un **algoritmo**.*
- *Según el problema que abordemos el algoritmo será más o menos complicado pero aprender a programar tiene más que ver con desarrollar esta capacidad de resolver problemas lógicos en una secuencia de pasos que podamos codificar en un ordenador.*
- *Por cierto, cualquiera puede aprender a programar. No lo dudéis. Solo que como en todo, a unos les lleva más tiempo que a otros desarrollar la habilidad necesaria. Al principio muchos me dicen que les duele la cabeza de pensar en este tipo de cosas, pero os animo a continuar (poco a poco si es preciso) porque os encontrareis que vale la pena.*

4.6 Resumen de la sesión

En esta sesión hemos aprendido varias cosas importantes:

- La instrucción For, nos permite iterar un bloque de instrucciones tantas veces le indiquemos.
- Hemos visto uno de los tipos de variables que C++ acepta: los enteros.
- Hemos introducido el concepto de algoritmo, como un procedimiento secuencial para resolver un problema concreto y lo hemos aplicado a varios ejemplos de programas sencillos con luces.