Taller de Arduino

Conocimientos básicos y sensor de presencia

Guion elaborado por Victor Perez Domingo para el $CEEI$ (Centro Europeo de Empresas e Innovación).

El objetivo de esta lección es adquirir unos conocimientos básicos de cómo funciona Arduino y poder construir con ello un sensor de presencia que encienda una luz.

Tabla de contenido

¿Qué es Arduino?

"Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects"

www.arduino.cc

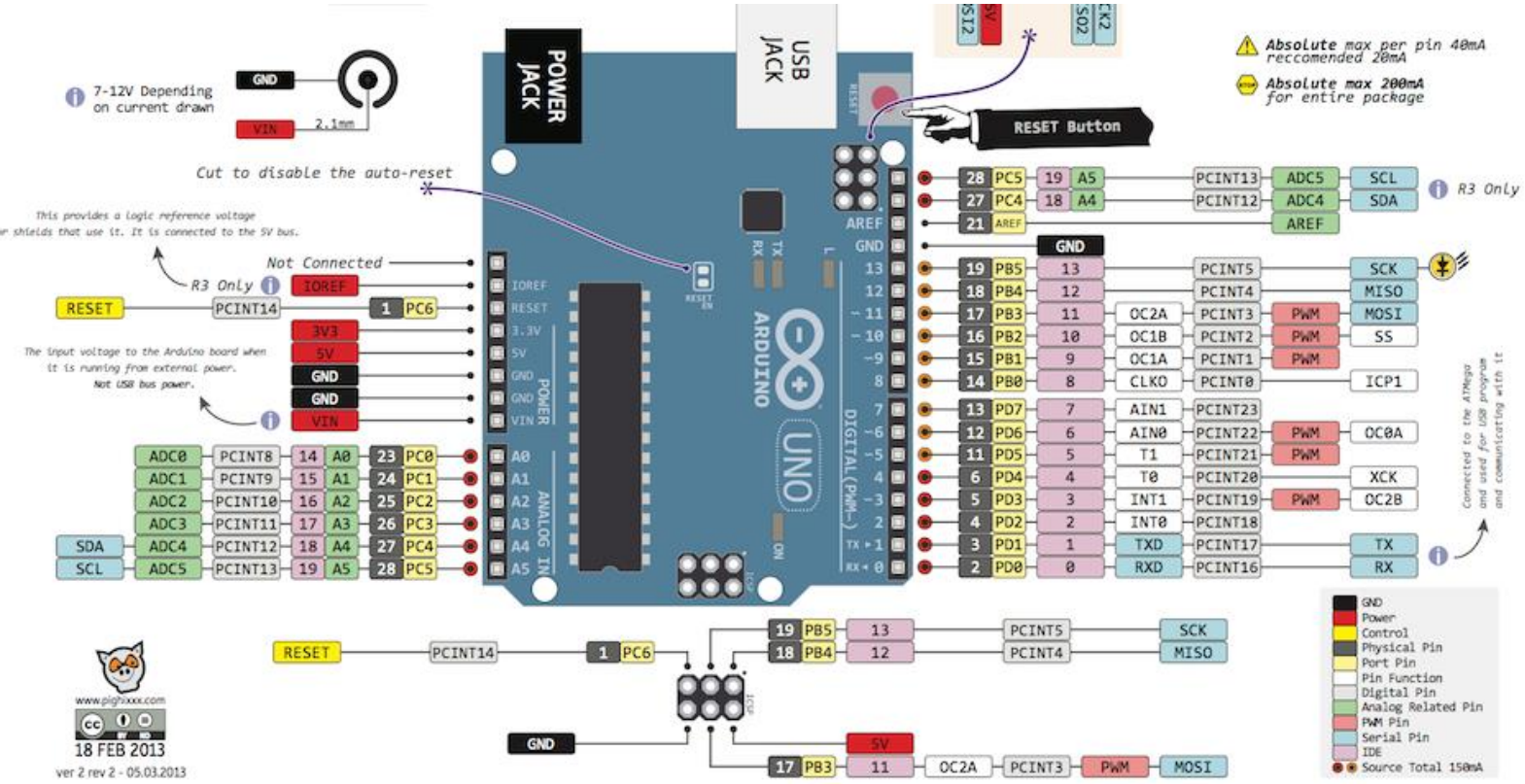
Esta definición es la que se puede obtener en la página oficial de Arduino y describe a la perfección lo que podemos conseguir y lo que nos ofrece un Arduino

- Open-source -> ¡barato!
- Easy to use -> ¡cualquiera puede aprender!

Por dar una definición un poco más técnica, Arduino es un microcontrolador (ATmega328P) del fabricante Atmel al cual se le ha cargado un bootloader que le permite ser programado fácilmente a través del puerto USB utilizando un programa conocido como Arduino IDE.

Gracias a este gran trabajo nosotros podemos programar un Arduino para que haga cosas increíbles sin tener que enfrentarnos a hojas de datos como la del ATmega328P que he dejado arriba.

Otra página que no puede faltar es la referencia del lenguaje de Arduino, existe una versión en español pero no la recomiendo.



Aquí podemos ver un buen resumen de que hace cada pin en un Arduino.

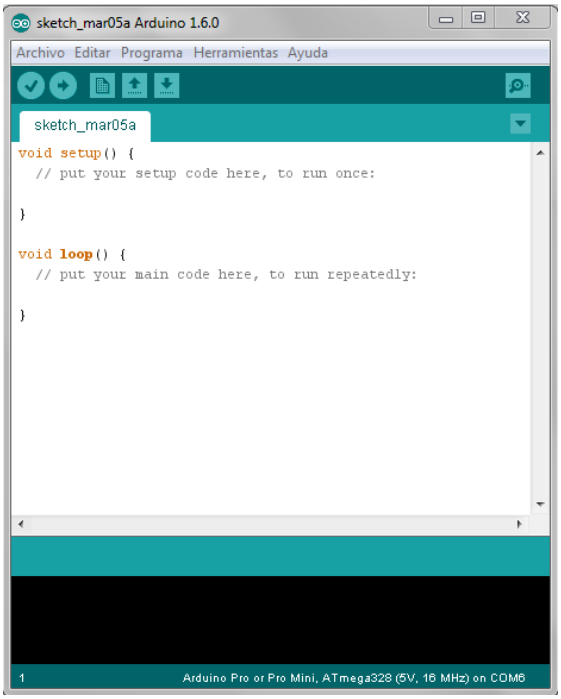
No hay porque agobiarse por entender todo, solo échale un vistazo.

Voy a destacar unos detalles de la imagen anterior.

- El límite de corriente es de 40mA (20mA mejor) por pin y 200mA en total.
- Los pines 0 y 1 los utiliza Arduino para la comunicación serie, tenerlo en cuenta a la hora de cargar un programa
- El pin 13 lleva un led integrado

Instalar y utilizar Arduino IDE

Hay que realizar la descarga desde la página oficial e instalarlo como haríamos con cualquier otro programa.



Al ejecutarlo por primera vez debería aparecer una imagen como esta.

Cargando nuestro primer programa

Para aprender a usar el IDE y de paso verificar que nuestro Arduino funciona correctamente vamos a realizar los pasos necesarios para cargar el programa más sencillo de Arduino conocido como blink.ino.

En el menú Archivo>Ejemplos disponemos de una gran cantidad de ejemplos, blink.ino se encuentra en Archivo>Ejemplos>Basics>Blink

Por ahora no vamos a entrar en entender cómo va el programa pero si lees la descripción nos indica que hace parpadear el led integrado en la placa Arduino en el pin 13.

Antes de pulsar el botón cargar, el segundo de la barra superior, tenemos que configurar arduino IDE para que pueda programar nuestro tipo de placa.

En el menú Herramientas>Placa vamos a seleccionar nuestro modelo que es “Arduino UNO”

En el menú Herramientas>Puerto tenemos que seleccionar el puerto en el que está conectado nuestro Arduino, si no aparece es porque faltan los drivers aunque esto no suele pasar si el Arduino es original. Si aparecen varios puertos y no sabes cuál es, en el administrador de dispositivos de Windows puedes ver donde está conectado.

En el menú Herramientas>Programador usaremos AVR ISP

Con todo esto configurado solo hay que darle a cargar y en unos segundos nuestro led 13 empezara a parpadear.

Ejemplo Blink.ino

En este ejemplo vamos a aprender los conceptos básicos de la programación de Arduino y como interactuar con salidas digitales.

La parte de código en color gris corresponde a los comentarios por lo que no afectan al funcionamiento.

Si observamos el código podemos distinguir fácilmente 2 zonas bien definidas. void setup() y void loop() las cuales se encontraran siempre en cualquier programa de Arduino.

- Void setup() es una función que se ejecuta al encender o reiniciar el Arduino una única vez.
- Void loop() es una función que se ejecuta de forma infinita de forma cíclica.

Conociendo esto ya podemos deducir que en setup inicializaremos el funcionamiento y en loop llevaremos a cabo las acciones que deseemos realizar.

Lo siguiente es ver que hacen las distintas funciones que aparecen en el código, podemos encontrar todas en la referencia del lenguaje de Arduino.

En este código aparecen 3 muy básicas.

- pinMode(pin,mode): nos permite seleccionar en modo de trabajo de un pin
- digitalWrite(pin,value): nos permite escribir un valor en un pin
- delay(time): nos permite retrasar la ejecución un tiempo determinado.

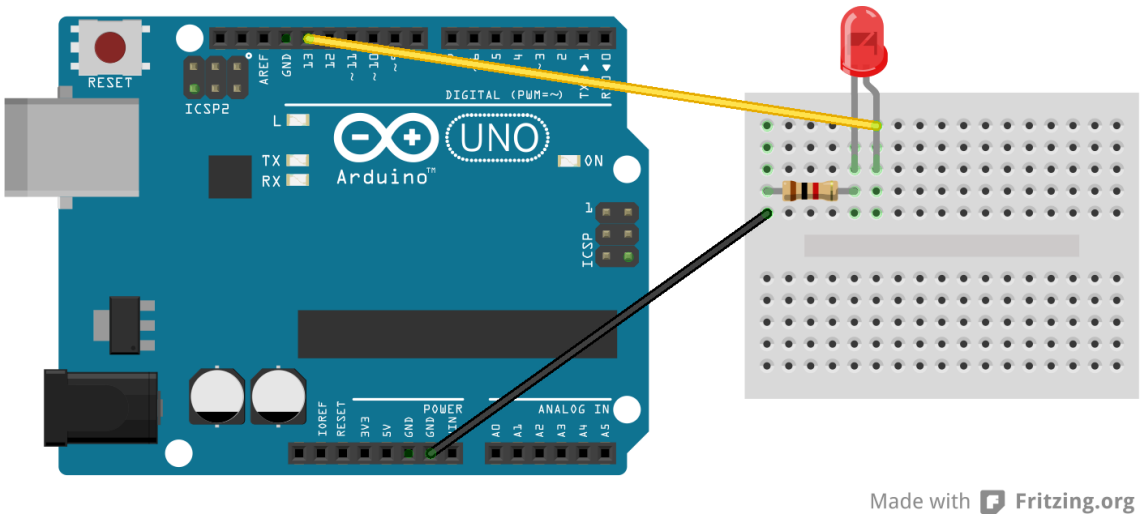
Dentro de cada una podemos encontrar toda la información necesaria para entender cómo funciona.

Practica

Es el momento de modificar el código, puedes intentar hacer que el led parpadee más rápido o más lento o incluso conectar un led en otro pin e intentar hacer que sea ese otro led el que parpadee.

Como conectar un led

La L en la patilla del led del dibujo representa la patilla larga del led real.



Ejemplo DigitalReadSerial.ino

En este ejemplo vamos a ver como leer entradas digitales y como ver datos a través del puerto serie en el ordenador.

A simple vista hay una gran diferencia con el código anterior, fuera del setup y del loop podemos ver una línea de código que dice “int pushbutton = 2”. Lo que hace esta línea es crear una variable global del tipo entero (int) de nombre pushbutton y con el valor 2 por defecto.

¿Para qué sirve una variable? Nos va a permitir guardar un valor y utilizarlo más adelante e incluso cambiarlo durante la ejecución del programa.

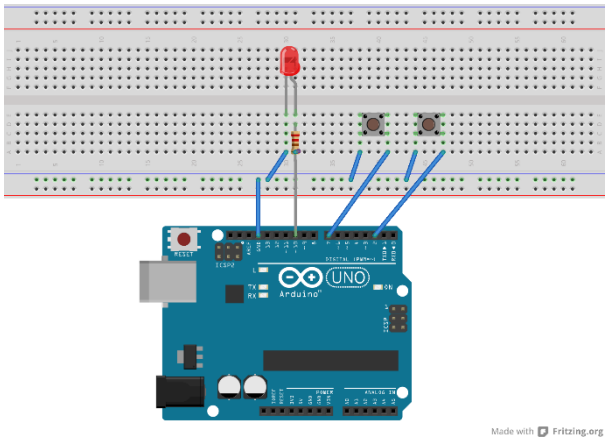
Hay que quedarse con la idea de que cuando nosotros ponemos pushButton Arduino entiende que queremos decir 2. Asi de sencillo.

Si echamos un vistazo a las funciones podemos ver 2 nuevas

- Serial.begin: empieza la comunicación.
- Serial.println: envía un dato seguido de un salto de línea.

Estas dos funciones son las que nos van a permitir enviar datos al ordenador a través del USB. Para verlos hay que abrir el monitor serie situado a la derecha arriba del IDE y seleccionar en el monitor la velocidad que le hemos puesto en Serial.begin a Arduino abajo a la derecha.

El objetivo de este sketch es leer el valor que hay en el pin 2 de Arduino, como se puede ver en el setup se configura el pin 2 para ser una entrada y en el loop se crea una variable buttonState que tiene como valor la lectura digital del pin 2 “digitalRead(pushButton)”



Como conectar el botón.

En esta imagen se puede ver un Arduino con 2 botones y un led conectados, simplemente fíjate en cómo están conectados los botones e intenta conectar tu un botón al pin 2.

Practica

Fácilmente podemos cambiar el pin que queremos leer, para esto solo hay que cambiar el valor de la variable pushbutton.

Practica

En vez de mandar los datos por el puerto serie al ordenador intenta encender el led 13 (u otro) cuando se pulse el botón.

Ejemplo AnalogReadSerial.ino

Este ejemplo es prácticamente igual que el anterior salvo por la diferencia de que estamos obteniendo el valor de una entrada analógica de Arduino.

Los valores leídos sobre una entrada analógica van de 0 a 1023 dependiendo de si en la entrada hay 0V o 5V

Nosotros contamos con una LDR que es una resistencia que cambia de valor dependiendo la cantidad de luz que incide sobre ella. Para poder medirlo necesitamos conectarla en un divisor de tensión y medir la tensión obtenida.

Como no disponemos de una resistencia externa del valor necesario vamos a utilizar una resistencia de pullup integrada en Arduino por lo que necesitamos modificar el código de AnalogReadSerial.ino añadiendo debajo de Serial.begin una línea que configure el puerto analógico con una resistencia de pullup.

- pinMode(A0,INPUT_PULLUP)

Con esta modificación solo tenemos que conectar la LDR entre el pin A0 y GND para obtener en el puerto serie una medida de la luz que esta midiendo.

Practica

Encender un led cuando la luz sea insuficiente sería algo útil. Recomiendo primero medir el valor obtenido con la luz de ambiente e ir buscando el valor en el cual queremos que se encienda la luz.

Ejemplo BlinkWithoutDelay.ino

Este ejemplo se encuentra en la sección digital en vez de basics.

Como bien dice su nombre hace parpadear un led sin utilizar la función delay lo que hace que el programa sea mucho más eficiente a la hora de realizar múltiples tareas ya que el procesador no se queda parado nunca.

El ejemplo es más complicado pero leído con atención se puede llegar a entender sin dificultad.

La parte más importante es la función millis() que lo único que hace es devolvernos el valor en milisegundos desde que arrancó el programa por lo que con una serie de comparaciones podemos usarla para el control de intervalos de tiempo.

Analizando la condición que viene en el programa podemos entender su funcionamiento

- if(currentMillis - previousMillis >= interval)
    - currentMillis -> tiempo actual
    - previous millis -> tiempo en el que se cumplió por última vez la condición
    - interval -> intervalo que deseamos conseguir

la mejor manera de entender esto es imaginar en tu cabeza como van a ir cambiando los valores de las variables.

Control de una luz automática

Antes de poder realizar este proyecto es necesario entender todos los ejemplos anteriores ya que necesitaremos un poco de cada uno.

El objetivo es conseguir que una luz se encienda cuando el detector vea una persona siempre que la luz de ambiente no sea suficiente para ver por lo que necesitamos:

- control digital de la luz para encender y apagar
- lectura digital del sensor para saber si hay persona
- lectura analógica de un LDR para saber la luz
- temporización para regular el tiempo que se mantiene encendida después de ver a alguien

Recomiendo abordar el proyecto por partes, lo primero que se puede hacer es conectar el sensor e intentar ver que señal nos da. Podéis encontrar información valiosa en la hoja de características de nuestro sensor que es el HCSR501.

Una vez tengáis este apartado claro podéis intentar añadir un led e intentar que imite la señal del HCSR501 o lo que sería mejor, que una vez detecte una señal se quede encendido durante 30 segundos antes de apagarse.

Si conseguís esto solo queda añadir que si el LDR detecta que hay luz no se pueda encender el led y ya tendríamos nuestro control automático de iluminación.