ÍNDICE DE IMÁGENES

Contenido

ARDUINO

Fuente:

https://www.youtube.com/watch?v=RaEfpLnAxh8&list=PLEzmH7aN82FEh2JjYuNCvFu6wFolHai32&index=4

En lugar de la versión MAC, en mi caso me descargue la versión de Windows. Esta explicado en el Word/PDF del día 1. Hay cables y cables para conectar el Arduino con el ordenador. Algunos solo tienen los filamentos de potencia y otros tienen los de potencia y para enviar datos. En la Imagen 1 podemos ver el LED encendido:

Imagen 1: LED encendido

Una vez que hayamos hecho esto, vamos a cargar un programa de ejemplo que es el “Blink”. Este programa nos sirve para determinar si el Arduino funciona o no.

Para poder cargarlo nos vamos a “File” → “Example” → “Basic” → “Blink”. Tal y como se muestra en la :

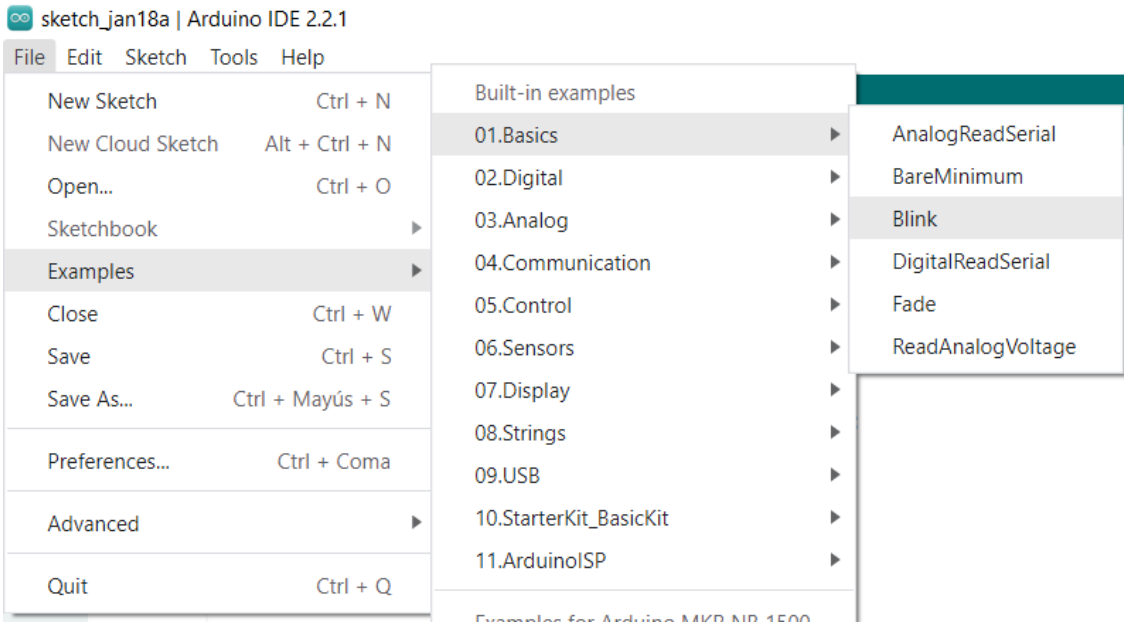

Imagen 2: LED encendido

Y lo que nos queda es lo siguiente:

```
24
25  // the setup function runs once when you press reset or power the board
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
34    delay(1000);                      // wait for a second
35    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
36    delay(1000);                      // wait for a second
37  }
38
```

Imagen 3: Programa “Blink” cargado

Para poder ejecutarlo hemos de seleccionar la placa en la que estemos trabajando. Para ello nos vamos a:

"Tools" → "Board: MKR NB 1500" en nuestro caso ya que ese es el modelo de Arduino que tenemos, (ver PDF día 1). Quedaría como en la imagen:

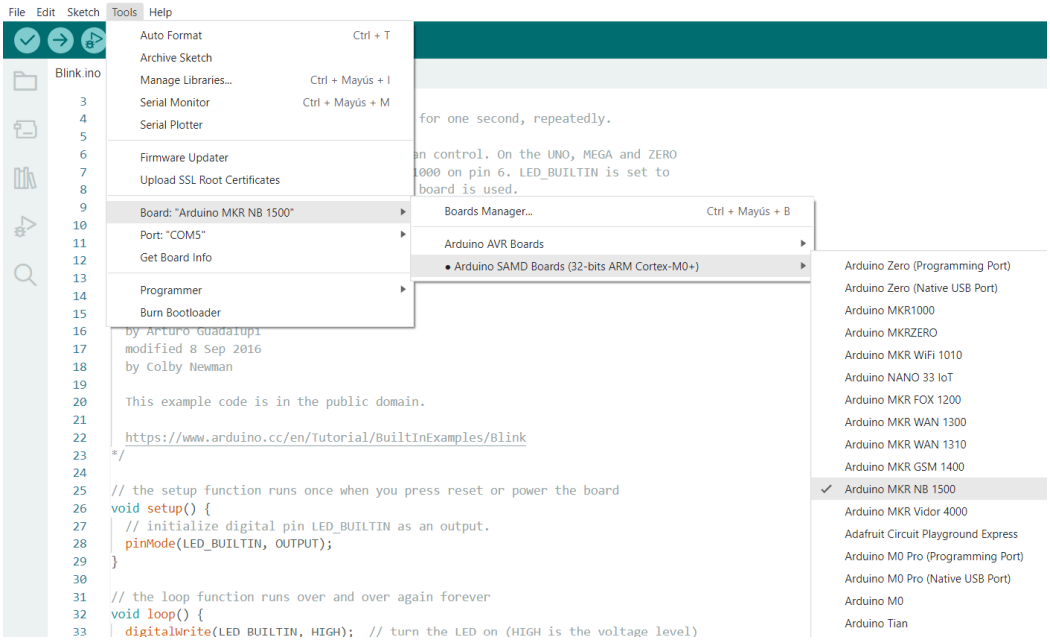

Imagen 4: Selección de la placa, ("Board")

Se nos enciende y apaga otro piloto que tiene el Arduino que es el que se muestra en la Imagen 5:

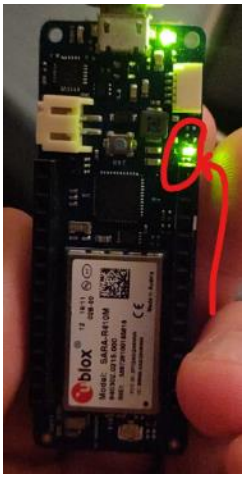

Imagen 5: Piloto que se enciende y se apaga con "Blink"

En el video sale como los LED correspondiente a T_x y a R_x se encienden al darle al botón "Upload" que es el botón de en medio de los 3 botones azules que están justo por debajo de "File", "Edit", … En nuestro modelo al no tener esos pilotos no se puede comprobar.

Si le damos a "Upload" se carga una serie de información que al menos por ahora no sé interpretar, pero nos dice que el Arduino funciona, (drivers bien instalados, puertos y placa bien configurada,…

Fuente:
https://www.youtube.com/watch?v=3DtstTfLwfY&list=PLEzmH7aN82FEh2JjYuNCvFu6wFolHai32&index=5

En todo Arduino existen 2 funciones básicas: Setup() y Loop() que vienen puestas por defecto al abrir la pantalla del IDE.

El Setup() solo se ejecuta al inicio nada más abrir el IDE. En esta función definimos pines de entrada, de salida, variables,… Es la programación que **no debe cambiar durante todo el programa**. El Loop() en cambio es la parte que se va a ir repitiendo todo el tiempo en bucle.

Para probarlo usaremos la misma función que utilizamos anteriormente en este mismo archivo, es decir, la del "Blink", solo que esta vez la escribiremos manualmente. En el Setup() escribimos los pines de entrada y de salida que en este caso serán el 13, (salida correspondiente a R_x), y el 14, (entrada correspondiente a T_x).

Para ello escribimos lo que viene en la Imagen 6:

```
1   void setup() {
2     // put your setup code here, to run once:
3   pinMode(13,OUTPUT);
4   }
5
6   void loop() {
7     // put your main code here, to run repeatedly:
8   digitalWrite(13, HIGH);
9   delay(1000);
10  digitalWrite(13, LOW);
11  delay(1000);
12  }
13
```

Imagen 6: Programa "Blink" escrito a mano

Voy a explicar paso a paso la Imagen 6.

El primer paso es tener claro que queremos hacer con el Arduino. En este ejemplo lo que queremos es que el piloto de apagado y encendido de nuestro Arduino se encienda y se apague con 1 segundo de espera entre ambos estados, (ON y OFF).

Para ello definimos en el Setup() lo que vamos a utilizar, (repito, el programa solo define 1 vez al inicio lo que hay dentro del Setup()).

En el video de la fuente, Edgar Pons escogió el **pin digital 13** como pin de entrada al programa. No sé si lo escogió por ser el pin de transmisión o si fue al azar. En cualquier caso en la línea 3 dentro de la Imagen 6 ha de definirse el/los pin/es a usar.

Para ello se ha de escribir el comando "pinMode(…)" sin las "", con la "p" minúscula y con la "M" mayúscula. Los puntos suspensivos son las variables que van dentro de "pinMode".

Dichas variables son el pin a usar, (13), y si es de entrada o salida, (en este caso salida, (OUTPUT)). Para terminar la frase, (y todas las que involucren comandos), hay que usar punto y coma, (;). Quedaría como la línea 3 de la Imagen 6: pinMode(13,OUTPUT);.

Lo del punto y coma es porque el lenguaje de programación de Arduino está basado en C++. Para este programa en concreto hemos acabado con el Setup(), no es necesario definir ninguna otra variable.

Para el Loop() en cambio hemos de definir unas cuantas. La primera es si el pin esta apagado, (0V), o encendido, (5V). Para apagado puede usarse 0 o LOW y para encendido puede usarse 5 o HIGH. LOW y HIGH deben de estar en mayúsculas.

Para definir que **es un pin digital y no uno analógico**, hemos de escribir el comando "digitalWrite" una vez mas en la primera palabra, (digital), la primera letra va en minúscula y en la segunda, (Write), la primera letra va en mayúscula.

Entonces, en la línea 8 de la Imagen 6 escribimos digitalWrite(…); Como vamos a definirlo como encendido quedaría: digitalWrite(13, HIGH);

El Arduino enciende y apaga el piloto 16 millones de veces por segundo, (el ojo no lo ve), por lo que de querer comprobarse que el aparato está encendiéndose y apagándose cada segundo, se ha de definir este retraso.

Para ello se ha de usar el comando delay cuya variable es el tiempo **que esta en milisegundos**. Quedaría como en la línea 9 de la Imagen 6: delay(1000);

En la línea 10 hacemos lo mismo que en la línea 8 pero apagado. Quedaría como: digitalWrite(13, LOW);

Y por último definimos el segundo delay para el proceso inverso, (de apagado a encendido). Quedaría como: delay(1000);

Fin de la explicación de la Imagen 6.

Conforme se van complicando los códigos es necesario ir comentándolos para no perderse. Para ello en Arduino se usa "//". La parte del código que se comenta no se ejecuta, solo son comentarios para que tu u otra persona que tengáis que trabajar con el código lo entendáis.