Conceptos básicos de Arduino:

Cosas básicas de C:

Siempre que se acabe de escribir una línea, y esta no acabe en **}**, hay que poner punto y coma (**;**)

1. Variables:
2. Cómo declarar una variable:
    a. Se pueden declarar en cualquier momento, pero siempre se deben declarar ANTES de usarlas, así que por lo general, siempre se declaran antes del setup().
    b. int var; Declara una variable llamada var que solo puede tomar valores enteros(int es integer del inglés, y significa entero.)
    c. int var2 = 10; Declara la variable var2 y directamente le asigna el valor 10, pero este valor puede cambiar a lo largo del programa.
3. Cómo cambiar el valor de una variable:
    a. Si tenemos declarada la variable var2, basta con poner var2 = 5 para cambiar su valor. Ahora var2 valdrá 5, hasta que vuelva a cambiar.
4. Tipos de variable:
    a. Las numéricas hay enteros int , con decimal float, con decimal grande double, enteros grandes long int, y muchos más que ahora no son importantes.
    b. Para letras y caracteres, se usa char, por ahora.
5. Funciones:
    a. Las funciones en C son parecidas las matemáticas. Por ejemplo, una función que suma 5 a la variable se expresa como f(x) = x + 5. Pues en C, es parecido, se pondría: (los números en gris NO se ponen, es para la explicación)

```
1. int sumar(int x){
2. int suma;
3. suma = 5 + x;
4. return suma;
5. }
```

    b. En la línea 1., en primer int define la función, esto significa que la función devolverá un entero. Después, el nombre de la función, que en este caso es sumar. Después, entre paréntesis, al igual que en mates, la

variable, pero aquí hay que especificar si se trata de un entero u otra cosa.

c. En la línea 2., se declara una variable dentro de la función que solo existe para la función. Esta variable suma no se puede usar fuera de la función, ya que no existe fuera de esta.
d. En la línea 3., se escribe la operación matemática, que es muy sencilla.
e. En la línea 4., se especifica lo que devuelve la función, con return y la variable que se desea devolver, en este caso, suma.
f. En la línea 5., se cierra la función.

6. Tipos de funciones:
   a. Las funciones pueden devolver un valor, como la del ejemplo que devuelve un entero int, pero también puede devolver otro tipo de valor, como float o char o más. También puede ser que no devuelva nada, en ese caso se pone void. (vacío)
   b. La función del ejemplo solo tenia una variable de entrada, x, pero pueden tener todas las que se necesite, siempre que se especifique de que tipo son y se pongan en orden cuando se llame a la función.
7. Declarar y llamar a una función:
   a. Para declarar una función, se escribe como la del ejemplo FUERA del setup() y del loop(), arriba o abajo indistintamente. Solo hace falta declararla una vez, pero se la puede llamar todas las que haga falta.
   b. Para llamar a una función, basta con poner el nombre de la función y las variables. Por ejemplo:
   c. int var = 10;
   d. int var2;
   e. (declaración de la función sumar como en el ejemplo)
   f. var2 = sumar(var);
   g. Después de este pequeño código, var2 pasará a valer lo que devuelva sumar, que en este caso es 10+5=15. Dentro de la función, var pasará a llamarse x, pero SOLO dentro de la función, y la variable suma solo existe DENTRO de la función.

8. Librerías:
    a. Las librerías son archivos donde se almacenan muchas funciones. Por ejemplo, en C, o Arduino, no puedes calcular un seno o un valor absoluto sin haber llamado primero a la librería <Math.h> que contiene varias funciones matemáticas, por ejemplo, la función seno que seria la siguiente:
    b. float sen(float x); donde ya se sabe lo que significa cada palabra. Esta función, creada por otra persona e integrada ya en el lenguaje C, devuelve el seno del ángulo, en radianes, en tipo float, que es un numero con decimales.
    c. Para incluir una librería, se pone lo siguiente:
        i. #include <Math.h>
    d. Un ejemplo en Arduino es la librería <Servo.h>, la cual nos aporta las funciones necesarias para controlar un servo, las cuales se describen más adelante.

Funciones básicas:

- setup(){ } Sirve para configurar el programa. Solo se ejecuta una vez, cuando se inicializa el Arduino o se resetea.
- loop(){ } Todo lo que hay dentro de esta función se repite una vez se llega al final.
- delay(__); Sirve para hacer que transcurra un tiempo sin que pase nada. Es una forma de decirle al Arduino que espere. Son milisegundos, eso quiere decir que si se pone delay(1000); el Arduino se esperará 1segundo.
- pinMode(__,__); Sirve para configurar un pin. Puede ser de entrada de datos; pinMode(2, INPUT), o salida pinMode(2, OUTPUT).
- digitalWrite(__,__); Sirve para mandar una señal digital, 0 ó 1, al pin seleccionado. Por ejemplo, digitalWrite(2,1);, manda una señal de corriente al pin 2. Es lo mismo poner digitalWrite(2,1); que digitalWrite(2,HIGH); Para apagar la señal, se escribe digitalWrite(2,0);, o bien digitalWrite(2,LOW);.
- analogWrite(__,__); Sirve para mandar un pulso al pin seleccionado. Solo se puede usar para pines PWM (Pulse Width Modulation), los que tienen un ~. Simula una señal analógica, que puede estar entre 0 y 255. Por ejemplo, un LED conectado a un al pin ~3, que es PWM, brillará más con analogWrite(3,250); que con analogWrite(3, 100);.
- digitalRead(__); Sirve para leer una entrada digital. El Arduino leerá 1 si hay voltaje en el pin, y 0 si no lo hay. Solo hay que indicar el pin que se quiere leer.
- analogRead(__); Sirve para leer una entrada analógica. Solo se puede usar en los pines A0,A1,…. Ya que son los que pueden recibir entradas analógicas. El Arduino recibirá un valor entre 0(nada de señal) y 1023(toda la señal). Se puede hacer la prueba con un potenciómetro para ver como cambia fácilmente.

Funciones específicas:

- Servo servo1; Son esto se inicializa un servo. Se escribe ANTES de la función setup() y hay que incluir la librería servo. El nombre del servo, que en este caso es servo1, se puede cambiar al gusto, como si le quieres llamar Pepe a tu servo, pues se escribiría Servo Pepe;
- servo1.attach(__); Sirve para decirle al Arduino en que pin se conecta el servo.
- servo1.move(__); Sirve para mover el servo. Los servos se mueven entre 0º y 180º, aproximadamente. Si se escribe, servo1.move(60); el servo se moverá a su posición de 60º.
- Serial.begin(9600); Sirve para iniciar la comunicación por serial con el ordenador. Con esto, el Arduino se puede comunicar con el ordenador mandando información. Solo hace falta iniciarlo una vez en cada programa, asi que se escribe en el setup. El valor 9600 es el baudio de comunicación, que por ahora, siempre será ese, pero hay más.
- Serial.print(__); Sirve para que el Arduino envié al ordenador un mensaje. Si a es una variable, y se escribe Serial.print(a); aparecerá en el Serial el valor de a en el momento que se escribe la función. Para mandar un texto, se pone entre comillas: Serial.print("texto");.
- Serial.println(__); Es igual que la función anterior, solo que esta pone un salto de línea la final de lo que escribe.
- map(__,__,__,__,__); Sirve para cambiar un rango de valores por otro. Por ejemplo, si se lee var = analogRead(__); una entrada, var tendrá un valor entre 0 y 1023, pero a veces este rango no nos conviene por el motivo que sea, y queremos que nuestra entrada este entre 0 y 255, por ejemplo, pues con var2 = map(var,0,1023,0,255); el Arduino asignará a cada valor entre 0-1023 un valor entre 0-255, y lo guardará en var2.