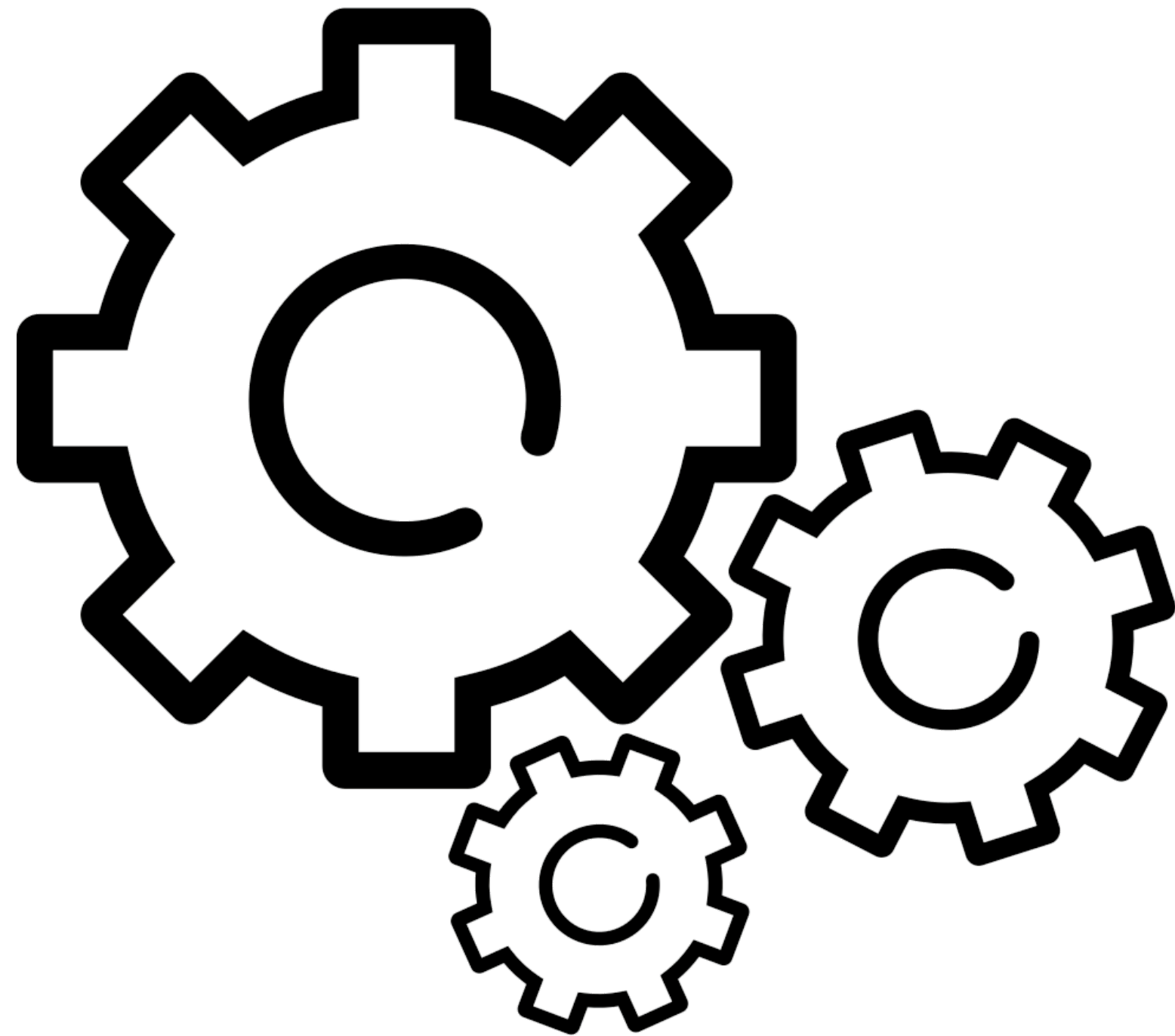


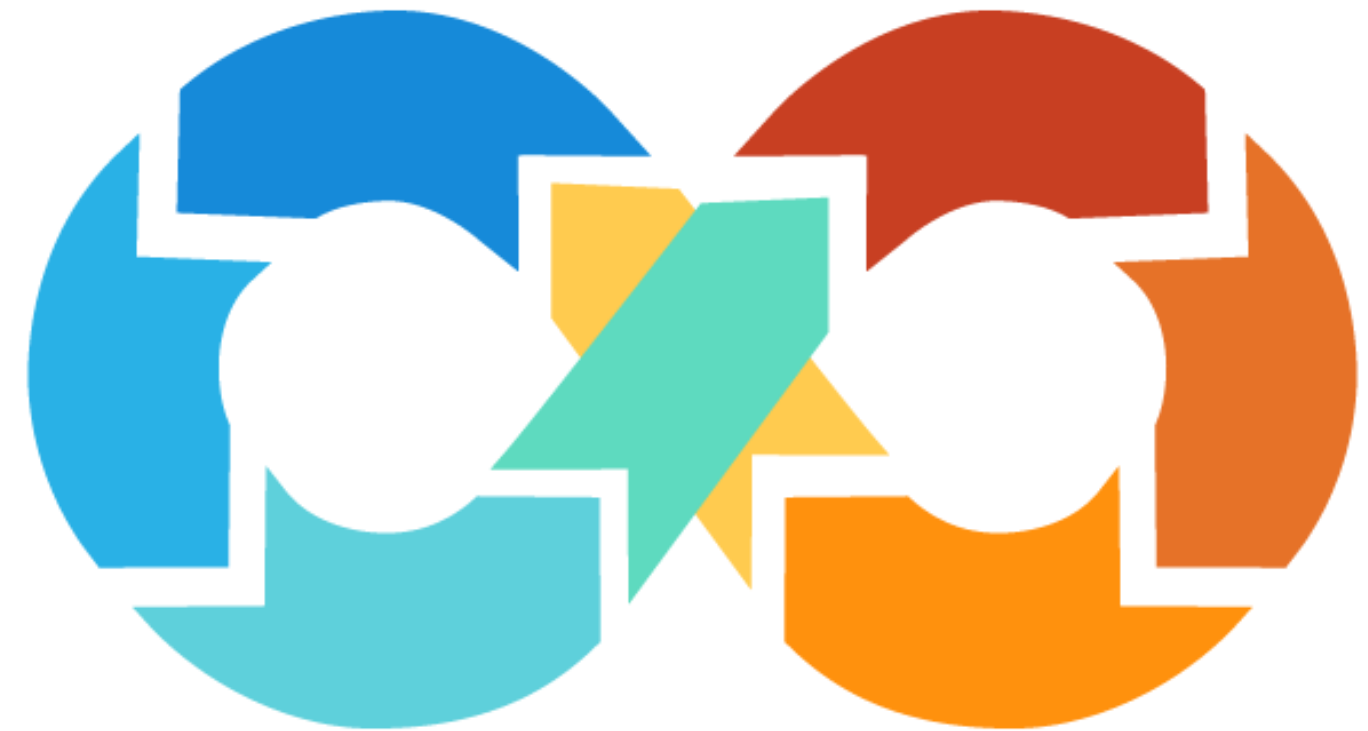
# Ciclos y funciones



# Ciclos iterativos



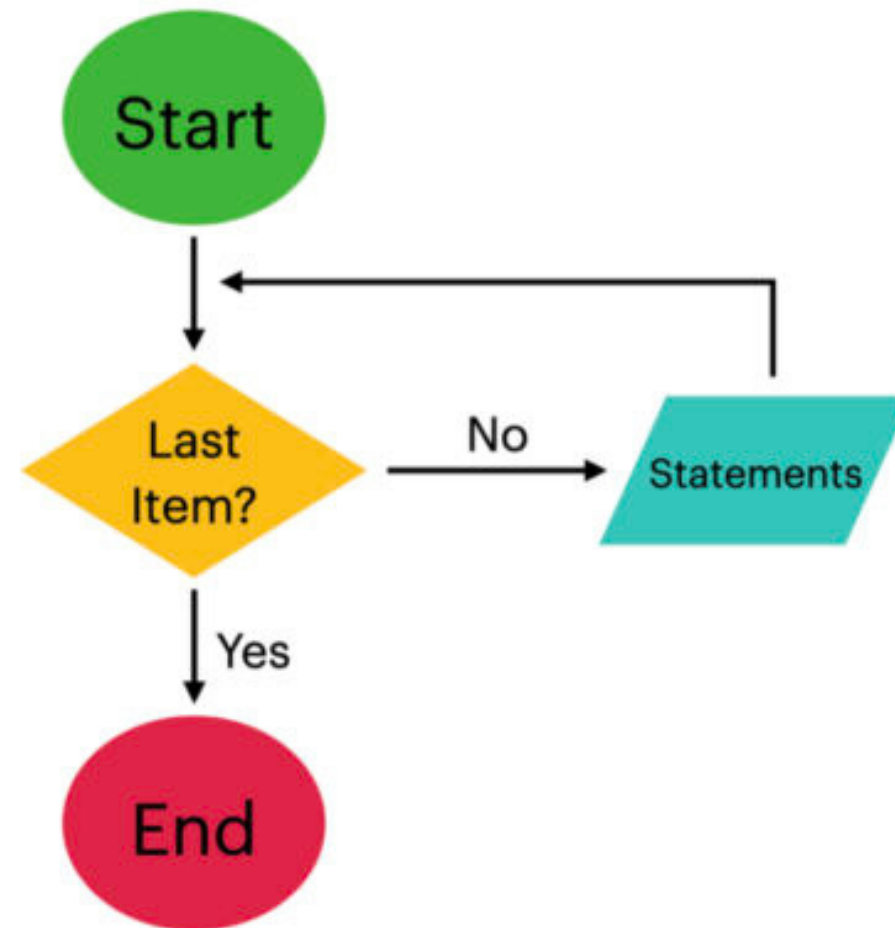
Un "loop" o ciclo iterativo es una estructura de control que ejecuta una o varias instrucciones de manera finita o infinita dependiendo de la validación de una o varias condiciones



# For



## For Loop



El ciclo For ó "For Loop" es una estructura de control cuya característica es que la principal condición de control es el constante monitoreo de una variable numérica del tipo int, a esta variable se le asigna un incremento o decremento dependiendo de las necesidades del programa.(1)

# ¿Cómo se usa For?



Declaramos la variable de "control", esta variable puede ser declara desde antes.

Mientras esta condición sea 'TRUE', el ciclo seguirá ejecutandose.

```
for(int i=0; i < 10; i++){  
    instruccion();  
}
```

Las instrucciones dentro de las llaves se ejecutarán el número de veces que el ciclo for valide la condición.

Incremento o razón de cambio de la variable de control.

# ¿Cómo se usa For?



```
for(int i=0; i < 51; i+=2)
```

Desde  $i = 0$ , mientras  $i$  sea menor a 51, incrementar  $i$  dos unidades.

```
for(j=21; j >= 0; j--)
```

Desde  $j = 21$ , mientras  $j$  sea mayor o igual 0, decrementar  $j$  una unidad.

```
for(int k=99; k>=12; k-=3)
```

Desde  $k = 99$ , mientras  $k$  sea mayor o igual 12, decrementar  $k$  tres unidades.

```
for(i=0; i < 100; i++)
```

Desde  $i = 0$ , mientras  $i$  sea menor a 100, incrementar  $i$  una unidad.

# Usos típicos del For



Contadores

```
for(i=0; i < 100; i++){  
    contador++;  
}
```

Llenado de arrays  
o arreglos

```
for(i=0; i < 100; i++){  
    arreglo[i]=valorSensor;  
}
```

Métodos  
numéricos

```
for(i=0; i <= 100; i++){  
    sumatoria+=i;  
}
```

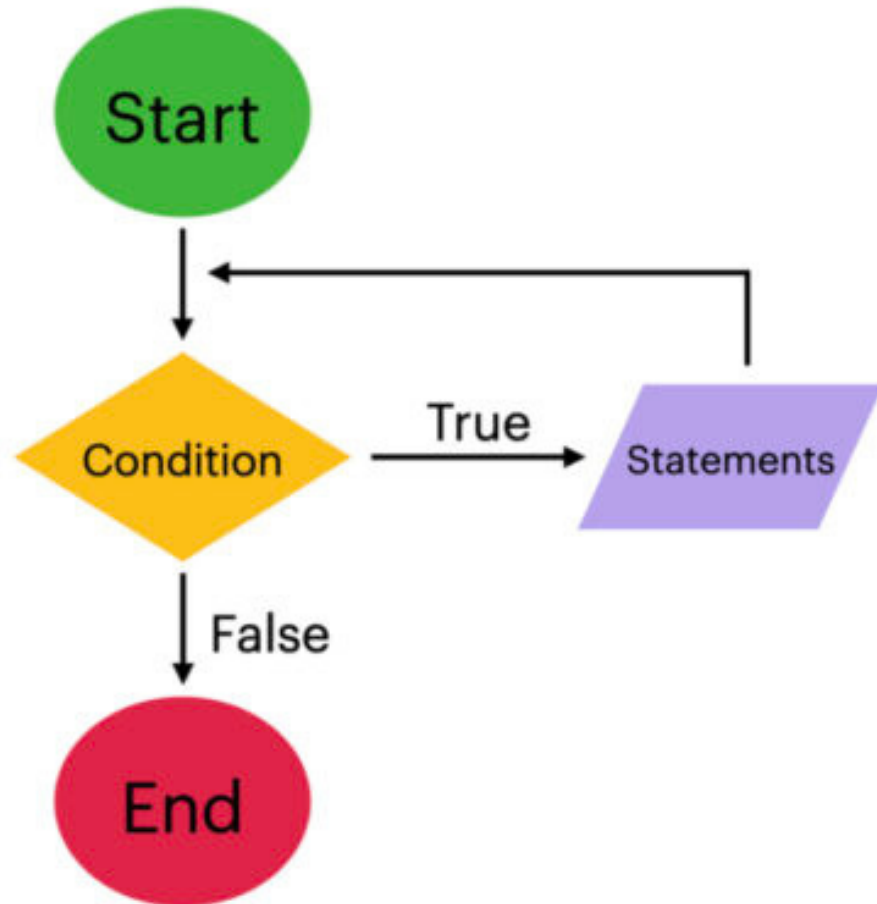
Suma de  
Gauss  
 $n(n+1)/2$



# While y Do...While



## While Loop



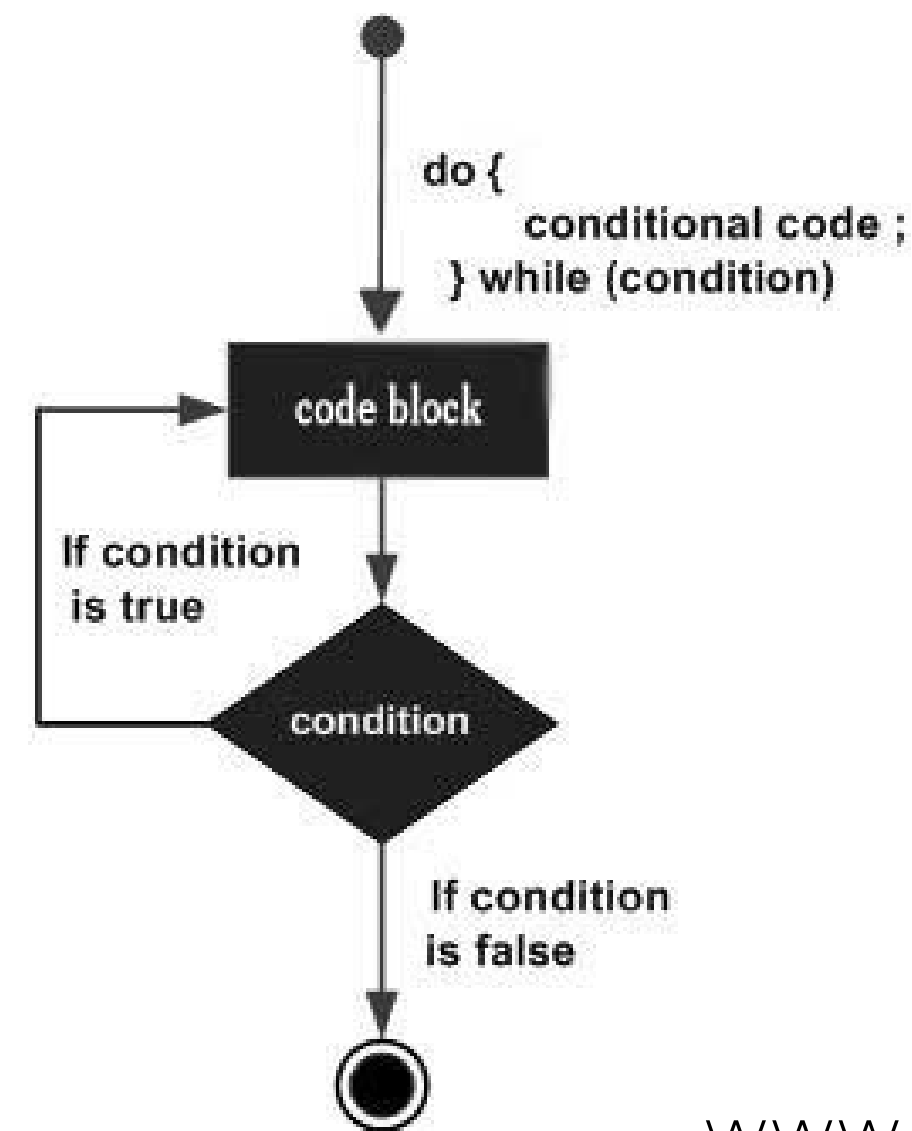
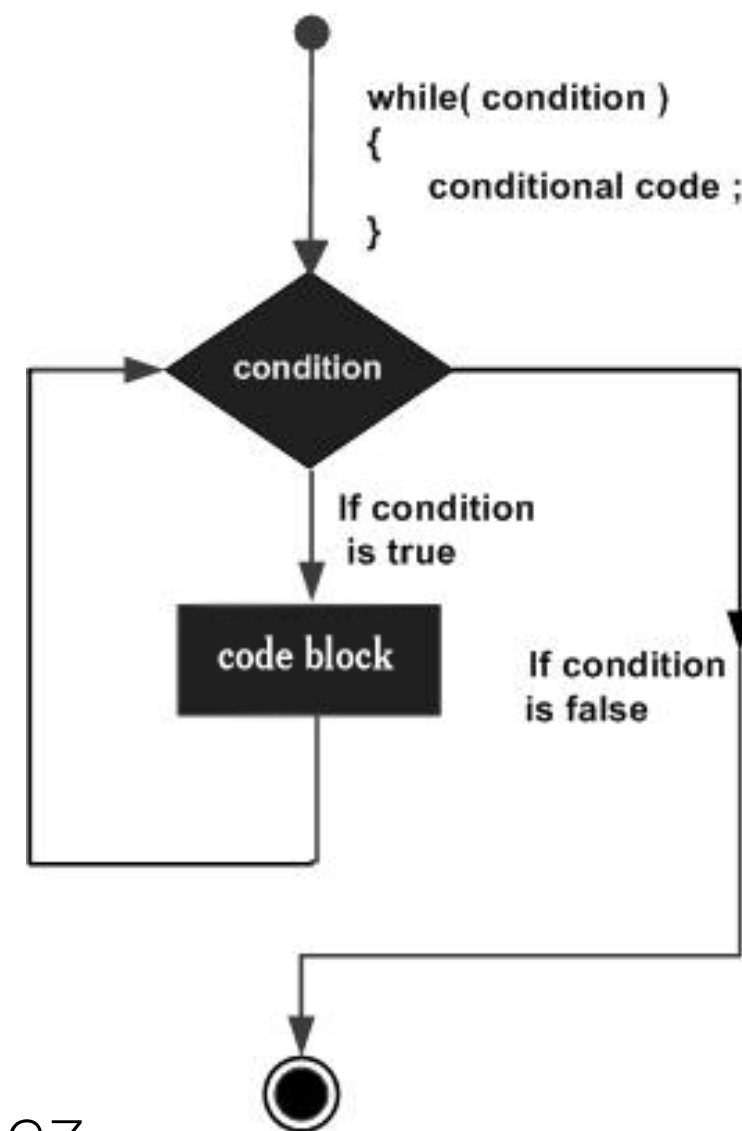
La siguiente estructura de control de programa es el ciclo while, este ciclo es útil para realizar instrucciones de manera repetitiva que no sabemos de manera exacta cuántas veces se repetirá el ciclo hasta que nuestra condición de iteración sea validada. Es decir, el ciclo se ejecutará infinitamente hasta que la condición de control NO se cumpla.(2)(3)



# While vs Do...While



La principal diferencia entre while y do while es la siguiente. While siempre "evaluará" primero la condición antes de entrar al ciclo, mientras que do while, primero entra al ciclo y después "evalúa".





# ¿Cómo se usa While y Do while?



```
while(condicion){  
    instruccion();  
}
```

```
do{  
    instruccion();  
}while(condicion);
```

Para ambas condiciones se necesita un resultado o argumento de tipo boolean, es decir, en ambos ciclo se verifica que el estado sea 'TRUE' o 'FALSE'.

# Ejemplo Do while



```
do{  
    apagaBocina();  
}while(digitalRead(boton)==LOW);
```

En este ciclo se evalúa el estado donde se encuentre conectado un botón, si el botón se encuentra en un estado "bajo" el ciclo continuará habilitado llamando a la función apagaBocina.

# Ejemplos While



```
while(temperatura>30){  
    enciendeVentilador();  
}
```

En este ciclo while PRIMERO se evalúa la comparación, si la variable temperatura es mayor a 30 el ciclo entrará para llamar a la función enciendeVentilador.

# Precauciones



Es importante cuidar la lógica al hacer ciclos de control, ya que es un error muy común caer en contradicciones lógicas y estar evaluando de manera errónea las condiciones de cada ciclo. Esto puede llevar a que el programa se 'bugee' o se 'trabe' al caer en un bucle infinito.

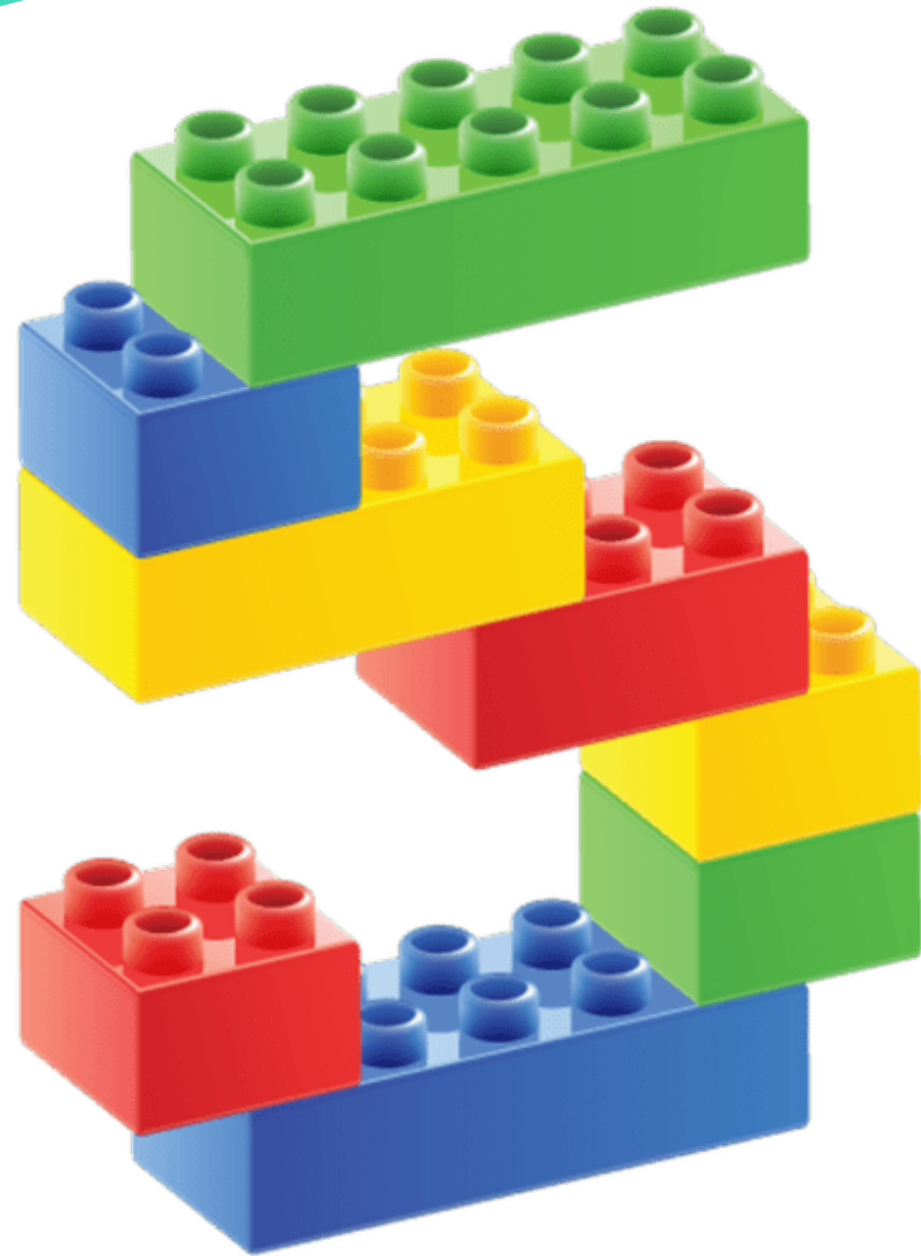


```
while(1==1){  
    //Este ciclo es infinito.  
}
```



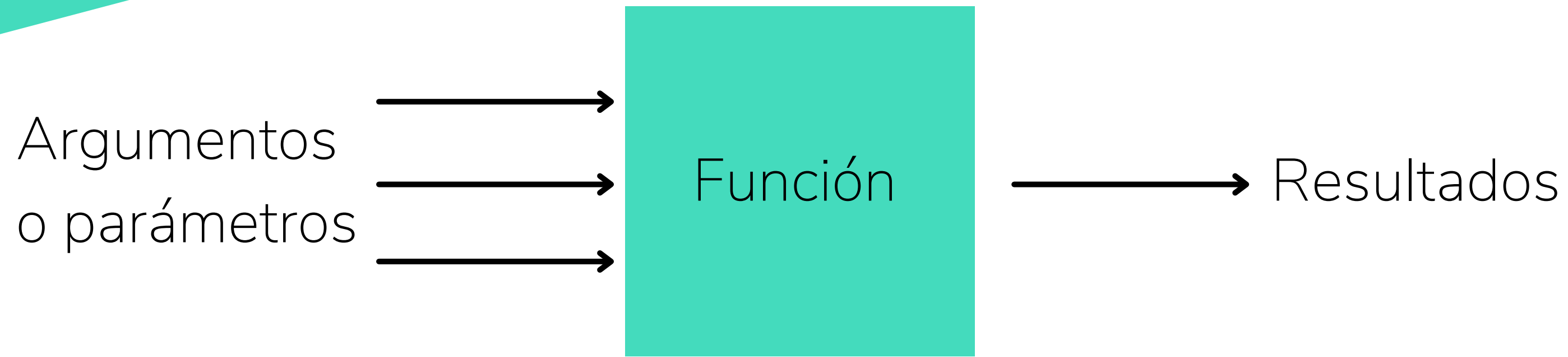
Tiempo de descanso y de restablecer humanidad.(:

# Funciones



Una función es un fragmento, bloque o parte de un código, que ejecutan una rutina o un conjunto de instrucciones en específico. Tienen la ventaja que ahorran espacio y tiempo a la hora de programar ya que podemos usar la funciones cuantas veces queramos.

# Funciones



Una función también puede ser entendida como un "subsistema" del código que uno diseña, este "subsistema" puede adquirir roles o tareas en específico.



# ¿Cómo se hace una función?



Tipo de dato que  
va a regresar mi  
función.

Tipo de dato que  
va a recibir mi  
función.

```
float nombreDeMiFuncion(float argumento){  
    float resultado;  
    return resultado;  
}
```

↑  
return es la palabra  
reservada que le indica a  
la función qué variable o  
dato debe de dar como  
salida.

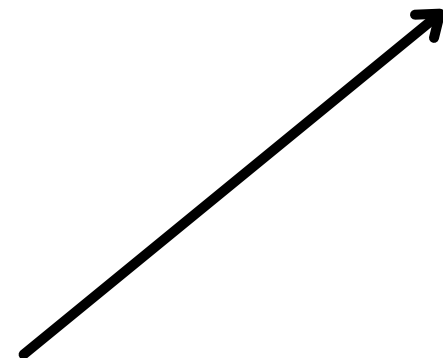
Toda variable que  
se declara dentro de  
una función es una  
variable local.


# ¿Cómo se usa una función?



Si la variable ya está declarada, sólo colocamos el nombre de la variable.

 `float salida = nombreDeMiFuncion(variable);`

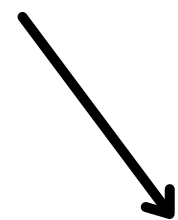
 Nombre de la  
función que  
declaramos  
anteriormente.

 Como argumento puede recibir cualquier variable o constante, siempre y cuando sea del mismo tipo de dato solicitado.

# ¿Cómo se usa una función?

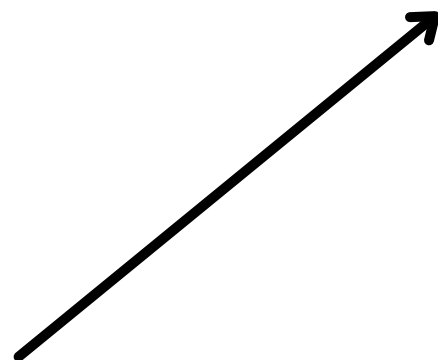


Puede ir dentro de otra función



```
Serial.println(nombreDeMiFuncion(variable));
```

Nombre de la  
función que  
declaramos  
anteriormente.



Como argumento puede recibir cualquier variable o constante, siempre y cuando sea del mismo tipo de dato solicitado.



# Ejemplo de funciones



Si quisiera hacer una función que calcule el IMC dado la altura y peso de un paciente.



## Prototipado

```
float calculaIMC(float altura; float peso){  
    float resultado;  
    resultado = peso/(altura*altura);  
    return resultado;  
}
```

## Implementación

```
IMCpaciente=calculaIMC(sensorAltura,sensorPeso);
```

# Ejemplo de funciones



Si quisiera calcular el área de un rectángulo.



## Prototipado

```
int areaRectangulo(int altura; int base){  
    return base*altura;  
}
```

## Implementación

```
area1=areaRectangulo(5,8);
```

# Ejemplo de funciones "vacías"



Son funciones que no reciben nada y no regresan nada, pero que dentro de las llaves ejecutan una serie de instrucciones

## Prototipado

```
void rutinaDelInicio(){  
    enciendeMotor();  
    apagaVentilador();  
    imprimePantalla();  
}
```

## Implementación

```
rutinaDelInicio();
```

# Bibliotecas



Una biblioteca o 'library' es un conjunto de funciones hechas por terceros que podemos descargar e implementar en nuestro propio código para ahorrar tiempo a la hora de desarrollar un nuevo proyecto.

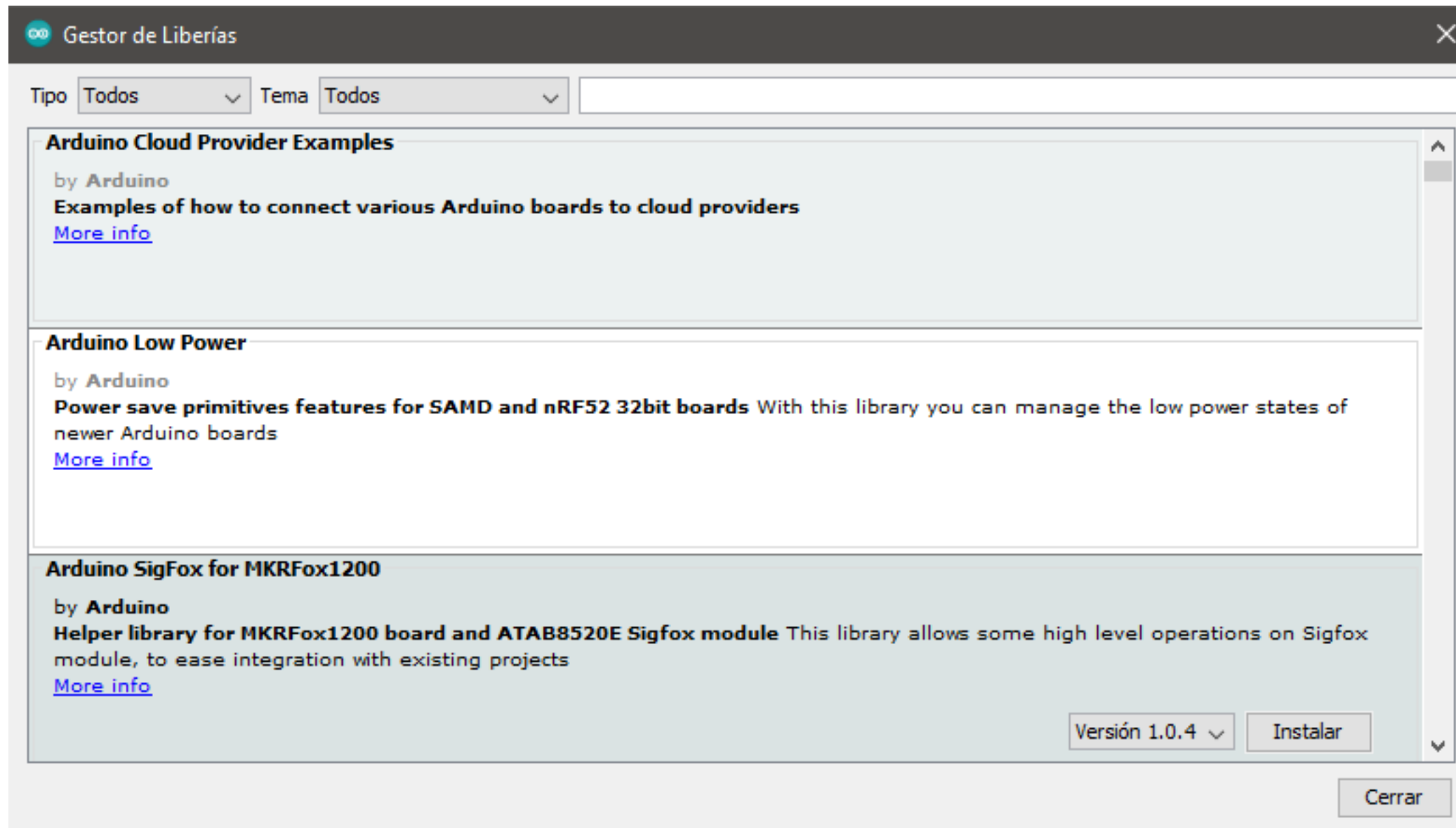




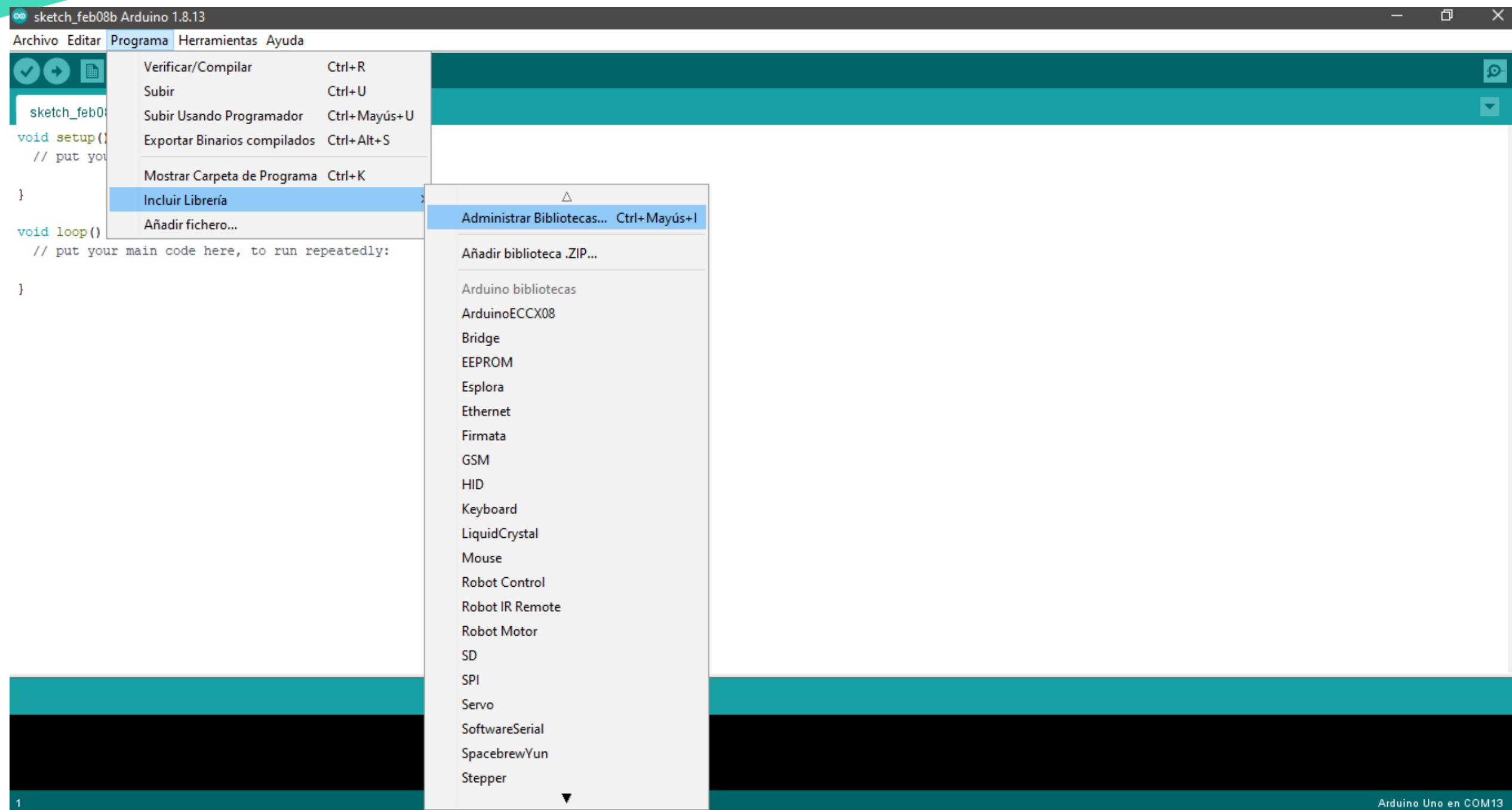
# Bibliotecas



La manera más sencilla de agregar alguna biblioteca a nuestro nuestro Arduino IDE es mediante el 'Gestor de Librerías'.



# Bibliotecas



*Programa > Incluir Librería > Administrar Bibliotecas*

# Bibliotecas



La gran mayoría de bibliotecas que se encuentran en Arduino cuentan con una documentación oficial de la persona o del equipo que se encargó de realizarlas, en esta documentación podemos encontrar información detallada de cada función y ejemplos sobre cómo implementarlas.

# Bibliotecas



Existe una gran aportación por parte de la comunidad de Arduino en cuestión de proyectos y bibliotecas como ninguna otra plataforma de desarrollo en microcontroladores, todo es cuestión de saber buscar la que mejor se adapte a tu proyecto y aprender cómo implementarla.



# Fuentes



1. <https://www.arduino.cc/reference/en/language/structure/control-structure/for/>
2. <https://www.arduino.cc/reference/en/language/structure/control-structure/while/>
3. <https://www.arduino.cc/reference/en/language/structure/control-structure/dowhile/>