RECOPILACIÓN DE BLOQUES BÁSICOS

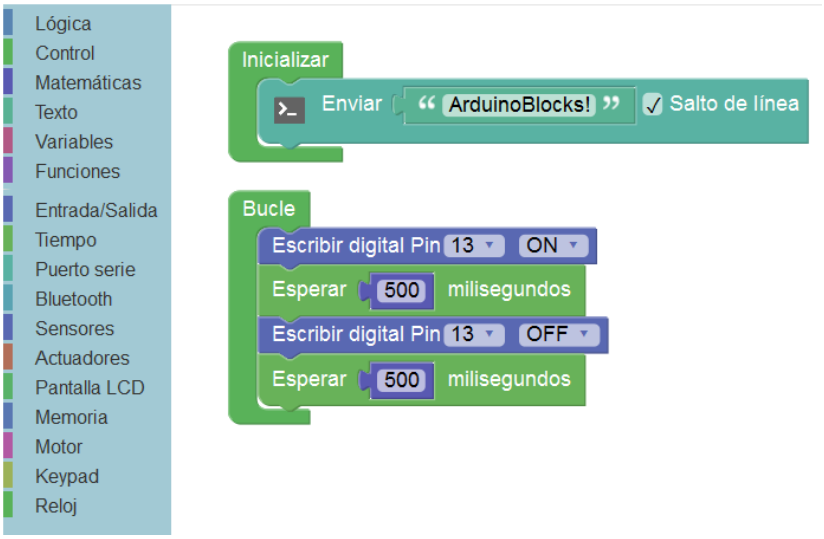
3 SOFTWARE

Una vez tenemos definido el hardware necesario para un proyecto el siguiente paso es programar el microcontrolador de la placa Arduino para que realice las tares necesarias para el funcionamiento deseado.

La programación de la placa Arduino se realiza normalmente en lenguaje C++ desde el entorno Arduino IDE. Para programar debemos conocer primero este lenguaje, lo cual supone mucho tiempo del que muchas veces no disponemos.

ArduinoBlocks, al igual, **es un entorno online que nos permite programar Arduino** (sin necesidad de conocer el lenguaje de programación C++) **de forma visual al estilo de programación de bloques.** ArduinoBlocks implementa bloques generales comunes a cualquier entorno de programación y por otro lado bloques específicos para Arduino donde podemos acceder a leer/escribir datos de los pines de entrada/salida, acceder a información de sensores conectados, manejar actuadores, periféricos como la pantalla LCD y muchas funcionalidades más.

Programa de ejemplo generado automáticamente en modo "prueba":



3.1 BLOQUES DE USO GENERAL

Los bloques de uso general nos permiten implementar funciones comunes en cualquier entorno o sistema programable. Esto incluye funciones lógicas, matemáticas, condiciones, bucles, funciones de texto, etc.

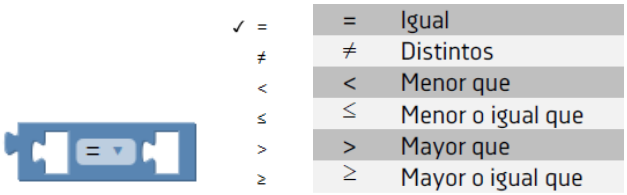
3.1.1 LÓGICA

Con estos bloques tenemos acceso a las funciones lógicas necesarias para implementar en nuestro programa de Arduino.

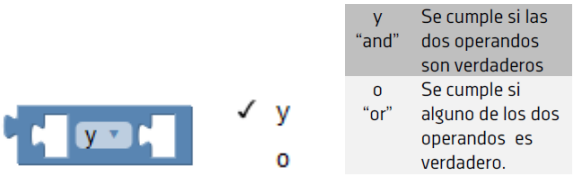
Las funciones lógicas trabajan con valores o expresiones de "verdadero" o "falso"

- **Condición / decisión:** Evalúa una condición lógica, si se cumple realiza el bloque “hacer” si no se cumple realiza el bloque “sino” (opcional)

- **Evaluar condición:** Devuelve verdadero o falso según si la condición indicada se cumple entre los dos operandos.

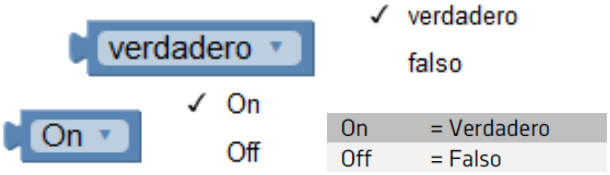


- **Conjunción/Disyunción:** Evalúa dos expresiones lógicas y devuelve verdadero o falso según la función lógica seleccionada.



- **Negación:** Permite negar (invertir) un valor lógico de verdadero o falso.

- **Constantes lógicas:** son valores booleanos indicando uno de los dos estados posibles



3.1.2 CONTROL

Las estructuras de control nos permiten realizar bucles e iteraciones.

- **Repetir:** Repite (n) veces los bloques de su interior.

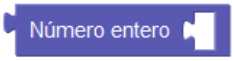
- **Repetir según condición:** Repite mientras o hasta que se cumpla una condición.

- **Contar:** Realiza un bucle contando con un variable índice. Se define un valor de inicio, un valor de fin y los incrementos que se realizarán en cada iteración del bucle. Dentro del bucle podremos usar esta variable.

3.1.3 MATEMÁTICAS

- **Constante numérica:** Permite especificar un valor numérico entero o decimal

- **Número entero / sin signo:** Trata el valor como un entero. Si especificamos sin signo, trata el valor como una variable sin signo internamente.
  Para las variables ArduinoBlocks utiliza el tipo de dato “double” cuando traduce el programa a lenguaje C++. En caso de hacer la conversión se trata como un “cast” a un tipo de datos “long” o “unsigned long”

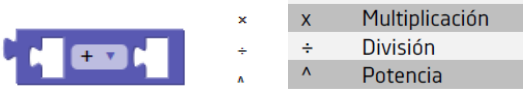


Trata el valor como tipo entero

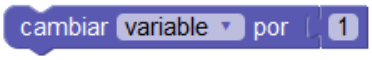
Trata el valor como tipo entero sin signo

- **Ángulo:** Permite definir un valor de ángulo en grados. Es un valor numérico tal cual, pero con la ventaja que permite definir el valor de una forma visual viendo el ángulo gráficamente.

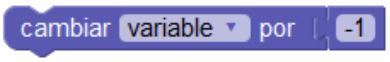
- **Operaciones básicas:**



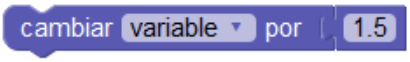
- **Cambiar variable:** Aumenta o disminuye el valor de una variable por el valor indicado (si es un valor positivo aumenta si es negativo disminuye)



Aumenta la variable en +1
variable = variable + 1

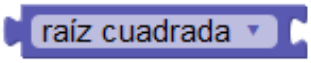


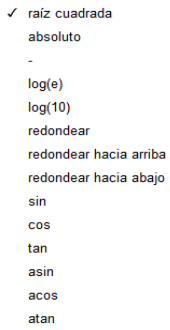
Disminuye la variable en -1
variable = variable – 1



Aumenta la variable en +1.5
variable = variable + 1.5

- **Funciones matemáticas:**





- **Mapear:** Permite modificar el rango de un valor o variable desde un rango origen a un rango destino. Esta función es especialmente útil para adaptar los valores leídos de sensores o para adaptar valores a aplicar en un actuador.

- **Limitar:** Permite acotar el valor mínimo y máximo.

- **Número aleatorio:** Genera un valor aleatorio entre los valores especificados.

- **Resto:** Obtiene el resto de la división de los dos operandos.

3.1.4 TEXTO

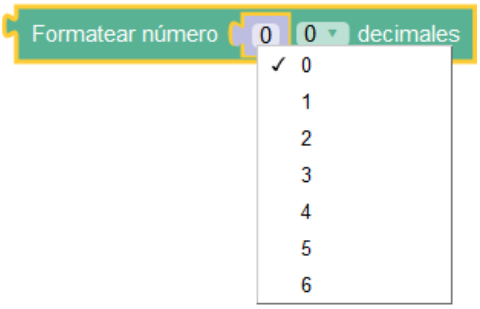
Las funciones de texto son especialmente útiles con la utilización en el puerto serie (consola), y otros periféricos como pantallas LCD. Permiten trabajar con variables de tipo texto o con textos prefijados.

- **Constante de texto:** Define un texto de forma estática.

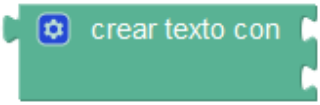
- **Formatear número:** Obtiene en forma de texto el valor de una variable o constante numérica en el formato especificado.

HEX	Genera el texto con la representación hexadecimal del valor.
DEC	Genera el texto con la representación decimal del valor.
BIN	Genera el texto con la representación binaria del valor.

- **Formatear número con decimales:** Realiza la conversión de una variable o constante numérica a texto igual que el bloque anterior pero pudiendo indicar el número de decimales a mostrar.

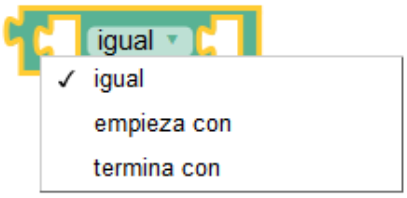


- **Crear texto con:** Crea un texto a partir de la unión de otros textos o variables. Las variables especificadas se convertirán a texto con formato decimal.

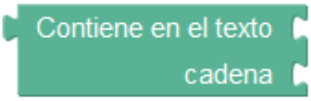


- **Longitud:** Obtiene el número de caracteres del texto.

- **Comparación de textos:** Permite comparar dos cadenas de texto. El resultado es un valor lógico de verdadero o falso.



- **Contiene el texto:** Comprueba si existe un texto dentro del texto indicado. Devuelve verdadero si existe y falso en caso contrario.

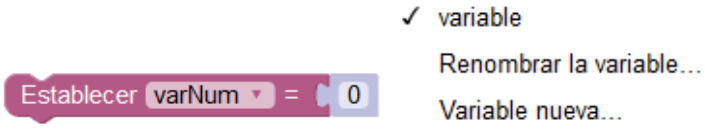


3.1.5 VARIABLES

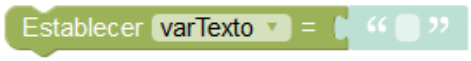
Una variable es un hueco en la memoria donde el programa puede almacenar valores numéricos. El sistema nos permiten asignarles un nombre simbólico como por ejemplo “temperatura exterior”, “velocidad”, “posición servo 1”,”estado”,… para facilitar su uso.

Hay tres tipos de variables en ArduinoBlocks: numéricas, booleanas y de texto.

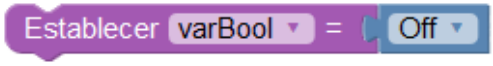
• **Variables numéricas:** permite valores numéricos enteros o con decimales, internamente se representan con el tipo de datos “doublé” a la hora de generar el código para Arduino. Este tipo utiliza 4 bytes y permite almacenar valores en el rango: -3.4028235E+38 a 3.4028235E+38

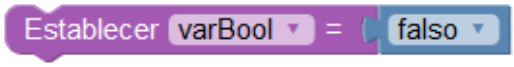


• **Variables de texto:** permite almacenar valores de texto. Internamente utiliza el tipo de dato “String” a la hora de generar el código para Arduino.



• **Variables booleanas:** permite almacenar valores lógicos booleanos de dos estados (verdadero/falso, ON/OFF, HIGH/LOW, …)



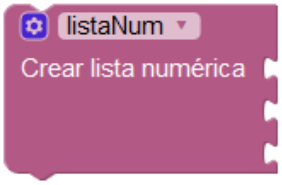


3.1.6 LISTAS

Las listas de datos nos permiten almacenar un listado de valores y acceder a ellos por su posición en la lista. Las listas pueden ser de tipo numéricas o de texto.

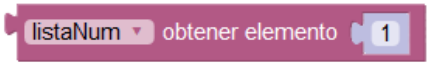
• **Listas numéricas:**

Podemos crear una lista asignándole un nombre a la lista y asignándole valores iniciales.

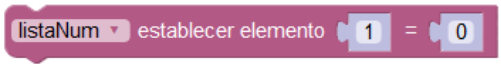


Para saber el número de elementos que tenemos en una lista podemos usar el bloque:

En una lista podemos obtener el valor de una posición (desde la 1 hasta el número de elementos en la lista) con el bloque:

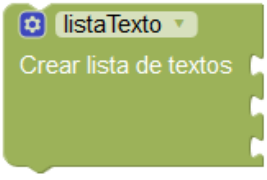


O cambiar el valor de un elemento indicando su posición y el nuevo valor:



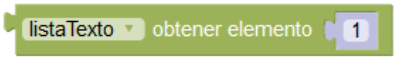
• **Listas de textos:**

Podemos crear una lista asignándole un nombre a la lista y asignándole valores iniciales.

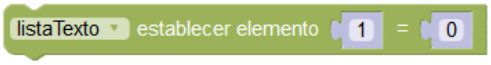


Para saber el número de elementos que tenemos en una lista podemos usar el bloque:

En una lista podemos obtener el valor de una posición (desde la 1 hasta el número de elementos en la lista) con el bloque:



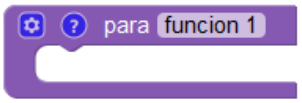
O cambiar el valor de un elemento indicando su posición y el nuevo valor:



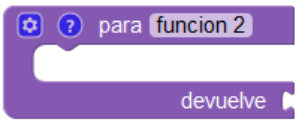
3.1.7 FUNCIONES

Las funciones permiten agrupar bloques de código. Esto es útil cuando un bloque de código se repite en varias partes del programa y así evitamos escribirlo varias veces o cuando queremos dividir el código de nuestro programa en bloques funcionales para realizar un programa más entendible.

- **Definición de una función:** La definición consiste en crear el grupo donde podremos insertar el código de bloques que forma la función. Debemos darle un nombre representativo que utilizaremos para llamar a esa función y ejecutarla.

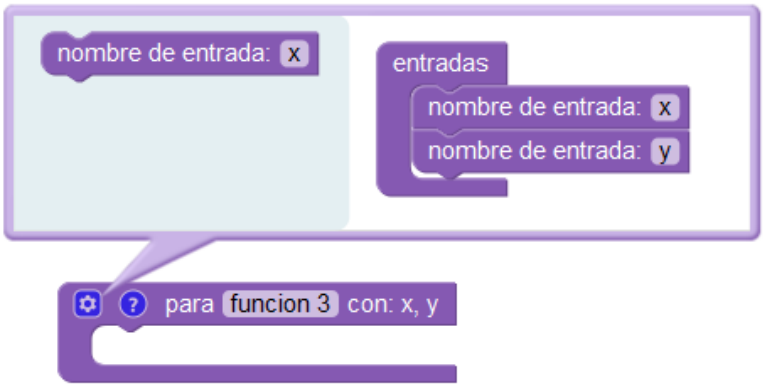


Función sin valor de retorno.
La función ejecuta los bloques de su interior y vuelve al punto de llamada.

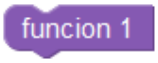


Función con valor de retorno.
La función ejecuta los bloques de su interior y devuelve un resultado.

- **Parámetros:** A las funciones se les pueden añadir parámetros para especificar en la llamada.

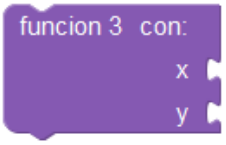


- **Llamada a una función:** Permite llamar a la ejecución de la función, se ejecutarán los bloques internos de la función y al terminar se seguirá la ejecución por donde se había realizado la llamada a la función.



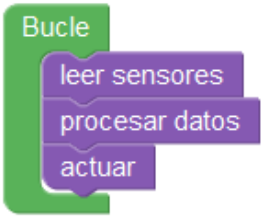
Llamada a una función sin valor de retorno.

Llamada a una función con valor de retorno.



Llamada a una función sin valor de retorno y con 2 parámetros

Llamada desde el bucle principal del programa:



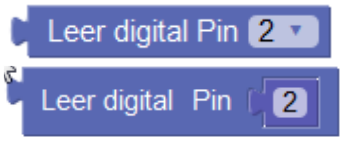
3.2 BLOQUES ARDUINO

En el siguiente apartado veremos los bloques relacionados con funciones propias de la placa Arduino. Estos bloques nos permitirán acceder a funcionalidades del propio microcontrolador y otros estarán orientados a sensores, actuadores o periféricos que podemos conectar a la placa Arduino para desarrollar nuestros proyectos.

3.2.1 ENTRADA/SALIDA

Las funciones de entrada/salida genéricas nos permiten leer o escribir en los pines digitales y analógicos de la placa Arduino.

- **Leer pin digital:** Obtiene el valor digital del pin (0/1, ON/OFF, verdadero/falso). (Recuerda para leer un ON/1 debemos aplicar 5v en la entrada digital y 0v para leer un OFF/0)

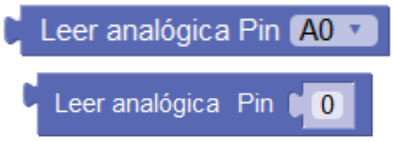


- **Escribir pin digital:** Escribe el valor en un pin digital pin (0/1, ON/OFF, verdadero/falso). (Si se activa, la salida suministrará 5v en caso contrario 0v)

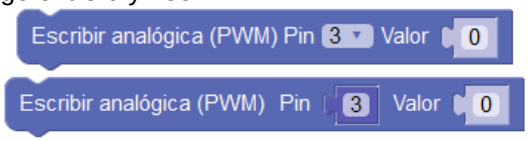
- **Leer pin analógico:** Lee el valor de una entrada analógica.
  El conversor interno DAC (Digital Analog Converter) es de 10 bits por lo que los valores leídos de una entrada analógica van de 0 a 1023

10 bits = 2 ^ 10 = 1024 posibles valores

Voltaje en la entrada analógica	*Valor leído*
0 voltios	0
2.5 voltios	512
5 voltios	1023



- **Escribir pin analógico:** Establece el valor del ciclo de pulsos activo/inactivo de una salida digital PWM. El valor debe estar en el rango entre 0 y 255.

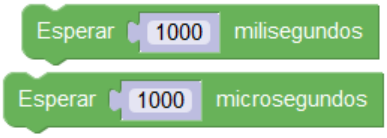


- **Leer pulso:** Lee un pulso en un pin hasta que el valor de la entrada cambie a estado alto (ON) o bajo (OFF). Mide la duración del pulso en microsegundos. Si se supera el tiempo de espera indicado sin cambiar de estado devolverá el valor 0.

3.2.2 TIEMPO

Las funciones de tiempo o retardo nos permiten realizar pausas y obtener información sobre el tiempo transcurrido dentro del microcontrolador.

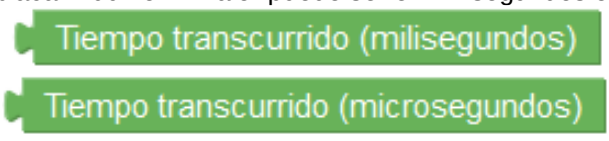
- **Esperar:** Realiza una pausa (bloquea la ejecución del programa) hasta seguir con la ejecución del siguiente bloque.



Milisegundos

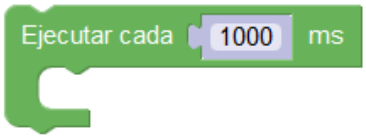
Microsegundos

- **Tiempo transcurrido:** Obtiene un valor con el tiempo transcurrido desde el inicio o reset del microcontrolador de la placa Arduino. El valor puede ser en milisegundos o microsegundos.



- **Esperar por siempre:** Bloquea indefinidamente la ejecución finalizando por tanto el programa.

- **Ejecutar cada:** Bloque que implementa automáticamente la función de tareas explicada anteriormente.
  IMPORTANTE: Este bloque no bloquea la ejecución del programa

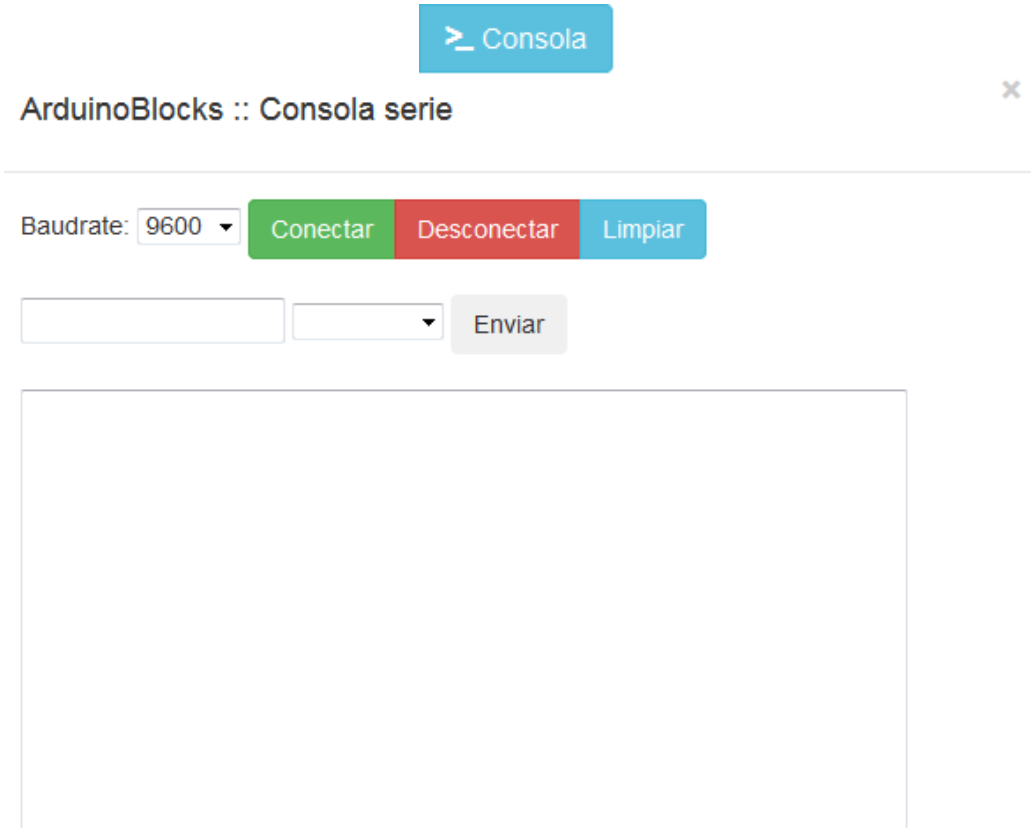


3.2.3 PUERTO SERIE

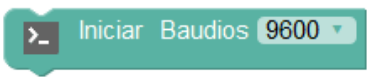
La comunicación vía puerto serie es muy utilizada. Es una vía de comunicación bidireccional sencilla que nos permite enviar información desde Arduino que visualizaremos en la consola o al contrario, enviar información desde la consola que recibiremos en el Arduino.

En muchas ocasiones simplemente se utiliza como una forma de depurar o mostrar información para saber si nuestro programa dentro del microcontrolador de Arduino está funcionando bien, en otros casos se puede utilizar de una forma más compleja sirviendo de vía de comunicación con aplicaciones en un PC, con periféricos como un GPS o comunicando con otros sistemas o por qué no, con otra placa Arduino.

En ArduinoBlocks tenemos acceso a la consola vía web (con ArduinoBlocks-Connector instalado) aunque podemos utilizar si lo preferimos cualquier aplicación de consola o terminal serie compatible con nuestro sistema.

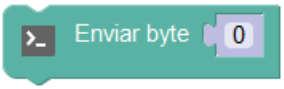


- **Iniciar:** Configura la velocidad de la comunicación serie. Este valor debe ser igual en la consola y en el programa Arduino para establecer una comunicación correcta. Por defecto, y si no se pone nada, la velocidad es 9600bps.



- **Enviar:** Escribe un valor de texto o el valor de una variable en el puerto serie. La opción “Salto de línea” permite añadir o no un retorno de carro al final del envío para bajar de línea.

- **Enviar byte:** Envía un valor numérico como un byte (8 bits). Por tanto el valor debe estar comprendido entre 0 y 255.



- **¿Datos recibidos?:** Obtiene un valor de verdadero si hay datos recibidos pendientes de procesar o falso si no se ha recibido nada por la conexión serie.

- **Recibir texto:** Lee una cadena de texto recibida por el puerto serie. Si se indica la opción “hasta salto de línea” en cuanto se encuentra un salto de línea devuelve el texto recibido. Si no, hasta que se dejen de recibir datos.

- 3.- SOFTWARE (Bloques)
  - 3.1 BLOQUES DE USO GENERAL
    - 3.1.1 LÓGICA
      - Evaluar condición
      - Conjunción/Disyunción
      - Negación
      - Constantes lógicas
    - 3.1.2 CONTROL
      - Repetir
      - Repetir según condición
      - Contar
    - 3.1.*3 MATEMÁTICAS*
      - Constante numérica
      - Número entero / sin signo
      - Ángulo
      - Operaciones básicas
      - Cambiar variable
      - Funciones matemáticas
      - Mapear
      - Limitar
      - Número aleatorio
      - Resto
    - *3.1.4 TEXTO*
      - Constante de texto
      - Formatear número
      - Formatear número con decimales
      - Crear texto con
      - Longitud
      - Comparación de textos
      - Contiene el texto
    - *3.1.5 VARIABLES*
      - Variables numéricas
      - Variables de texto
      - Variables booleanas
    - *3.1.6 LISTAS*
      - Listas numéricas
      - Listas de textos
    - *3.1.7 FUNCIONES*
      - Definición de una función
      - Parámetros
      - Llamada a una función
  - 3.2 BLOQUES ARDUINO
    - *3.2.1 ENTRADA/SALIDA*
      - Leer pin digital
      - Escribir pin digital
      - Leer pin analógico
      - Escribir pin analógico
      - Leer pulso
    - *3.2.2 TIEMPO*
      - Esperar
      - Tiempo transcurrido
      - Esperar por siempre
      - Ejecutar cada
    - *3.2.3 PUERTO SERIE*
      - Iniciar
      - Enviar
      - Enviar byte
      - ¿Datos recibidos
      - Recibir texto