

# EMPRESA 4.0

## MANUAL DEL ALUMNO

El concepto de “Empresa 4.0”, fue acuñado por el gobierno alemán para describir la fábrica inteligente, una visión de la fabricación informatizada con todos los procesos interconectados por Internet de las Cosas (IoT). Es lo que conocemos como Internet Industrial de las cosas o I2oT. Se espera que el nuevo concepto sea capaz de impulsar cambios fundamentales al mismo nivel de la revolución industrial a vapor, la producción en masa de la segunda y la electrónica y la proliferación de la tecnología de información que ha caracterizado a la tercera.

Según Mark Watson, director asociado para la automatización industrial de IHS, “El desafío para la cuarta revolución industrial es el desarrollo de software y sistemas de análisis que convierten el diluvio de datos producidos por las fábricas inteligentes en información útil y valiosa.”

Con los sistemas de automatización industrial integrando cada vez más sensores y capacidades de comunicaciones inalámbricas, las fábricas deben ganando en capacidad de reunir datos suficientes e interoperabilidad entre sus procesos. Para poder lograr mejoras reales en la eficiencia de fabricación y flexibilidad, los fabricantes deben ser capaces de gestionar y analizar grandes cantidades de datos, para lo cual el mayor desafío se encuentra en el lado del software.

En esta ocasión nosotros vamos a utilizar una herramienta llamada Briko que nos permite iniciar en el aprendizaje de la integración de sensores, programación, electrónica y adquisición de datos, con la intención de que puedas imaginar un poco de lo que sería posible realizar con el conocimiento de hardware y software dentro de unos años cuando puedas integrarte al mundo laboral.

### INICIANDO EL TRABAJO CON BRIKO

Briko es una herramienta desarrollada para apoyar en la enseñanza de la programación y entendimiento de la tecnología desde niveles básicos. Una realidad es que los trabajos actuales en muy poco tiempo van a evolucionar y depender cada vez más del entendimiento y dominio que podamos tener sobre tecnología.

Briko es un kit que busca integrar la electrónica, la robótica y la programación de forma que la creación de proyectos y nuevas propuestas tengan como único límite tu creatividad. Se busca que con las bases proporcionadas sea posible desarrollar generadores de creadores y no consumidores de nuevas tecnologías.

Los 3 elementos que forman el sistema con el que vamos a trabajar son:

- Los brikos que contienen la electrónica.
- Las piezas de construcción que equivalen a la parte mecánica.
- Y la programación por texto o bloques conocida como brikode o brikoblocks.

Los módulos de electrónica se conectan a un cerebro central que los controla llamado Bk7. La conexión entre ambos se realiza por cables de rápida conexión. Contamos con los siguientes módulos para el desarrollo de proyectos:

- Sensor de temperatura
- Botones
- Sensor de distancia
- Bocina
- Sensor de luz
- Display
- LEDs
- Sensor de distancia
- Relevador
- Perilla
- Motor
- Servo
- Cables planos
- Cable USB

El cerebro Bk7 tiene forma octagonal, es el elemento más importante ya que nos va a permitir la integración de los brikos, les va a suministrar la energía y es el que debemos conectar a nuestra computadora.

Cuenta con 7 puertos que nos van a permitir establecer diferentes funciones dentro de nuestros proyectos y laboratorios a desarrollar, toda la programación será desarrollada desde la computadora y comunicaremos a través de un cable micro USB, el cerebro Bk7 en donde vamos a guardar el programa que permita que cada uno de los brikos conectados funcionen.

La conexión entre los brikos y el cerebro Bk7, al cual se pueden conectar un máximo de 7 sensores de forma simultánea de ahí su nombre, se realiza mediante cables planos, los cuales cuentan con una muesca que nos indica la colocación correcta del cable.

El cerebro Bk7 cuenta con los siguientes indicadores

- Botón de reset/inicio: Al presionarlo una sola vez permite pausar el proyecto haciendo que quede inactivo y permite que al volverlo a presionar se reinicie, si se deja presionado por más de 4 segundos el Bk7 entra en modo de programación.
- Led indicador superior: Se encuentra encendido todo el tiempo para indicar que el Bk7 se encuentra recibiendo energía.
- Led indicador inferior: Tiene dos modos de parpadeo uno lento y constante que significa que el proyecto se encuentra en pausa y un parpadeo irregular y rápido que indica que el cerebro se encuentra recibiendo un programa de la computadora.

Las brikos que permiten visualizar las salidas de tus proyectos son

- Leds: Cuenta con 5 focos led que pueden prender con varios colores de acuerdo a tu proyecto.
- Bocina: Tiene la posibilidad de generar diversos sonidos y notas musicales dependiendo de su programación.
- Motor: Genera un movimiento rotatorio en el cual puedes controlar la velocidad y dirección de giro.
- Servo: Es un motor inteligente al que le puedes programar la posición exacta que requieres para tu proyecto.
- Relevador: Es el briko que te permite generar proyectos en donde puedas contar con corriente alterna.
- Display numérico: Tiene la capacidad de mostrar hasta números y letras de manera simultánea.

Los brikos de entrada van a permitir que alimentes con datos a tus proyectos para medir lo que sucede en el ambiente o bien que alguien los controle, se clasifican en entradas humanas y entradas de interacción.

Los sensores nos permiten medir variables del ambiente y existen 3 brikos que nos ayudan a ello.

- Sensor de distancia: Mide la distancia de un objeto y tiene un rango de función de 7 a 80 centímetros.
- Sensor de temperatura: Es un sensor que nos ayuda a medir la temperatura ambiente y funciona en un rango de -30 a 110 grados centígrados.
- Sensor de luz: Mide la cantidad de luz en el ambiente, proporciona valores entre 0 y 255, siendo 0 completa oscuridad y 255 el máximo valor de luz detectada.

Las entradas en donde se puede presentar interacción humana se dividen en:

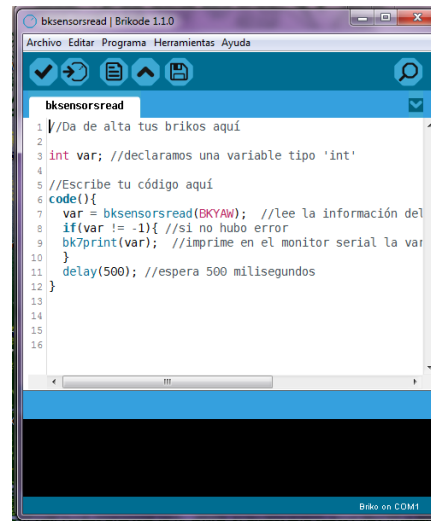
- Botones: Nos ayuda a que las personas puedan controlar acciones en los proyectos.
- Perilla: Nos sirve para poder incrementar valores de forma gradual, por ejemplo la intensidad de luz de un led o la velocidad de un motor.

La programación en briko utiliza su propio lenguaje de programación basada en Arduino, de una forma muy simplificada.

Existen dos forma de programación puede ser escrita en código utilizando un programa llamado Brikode o bien, una programación gráfica con bloques utilizando un programa propietario llamado Brikoblocks.

En ambos casos el software lo puedes obtener como descarga gratuita desde la página web de la empresa <http://briko.cc/instalacion>.

La apariencia de la versión de texto es la siguiente:

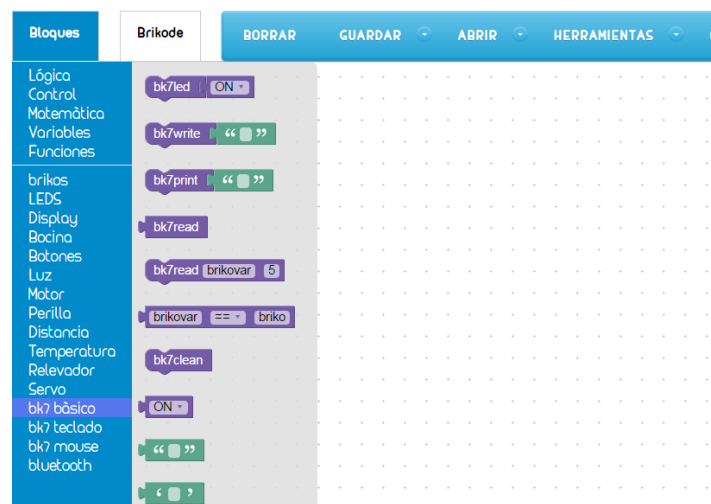


The screenshot shows the Brikode 1.1.0 text editor window. The title bar reads "bksensorsread | Brikode 1.1.0". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". The toolbar contains icons for file operations and execution. The main text area displays the following code:

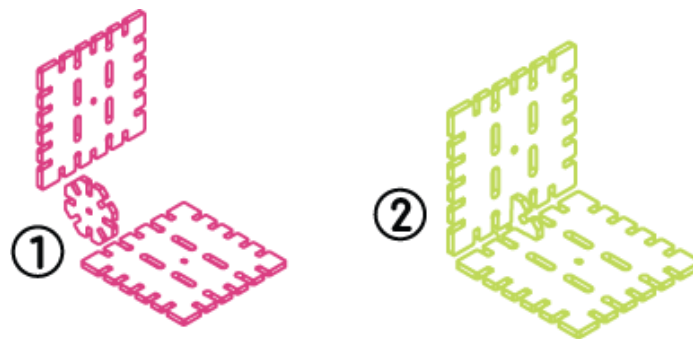
```
bksensorsread
1 //Da de alta tus brikos aquí
2
3 int var; //declaramos una variable tipo 'int'
4
5 //Escribe tu código aquí
6 code(){
7   var = bksensorsread(BKYAW); //lee la información del
8   if(var != -1){ //si no hubo error
9     bk7print(var); //imprime en el monitor serial la var
10  }
11  delay(500); //espera 500 milisegundos
12 }
13
14
15
16
```

The status bar at the bottom indicates "Briko on COM1".

La versión por bloques se puede visualizar de la siguiente forma:



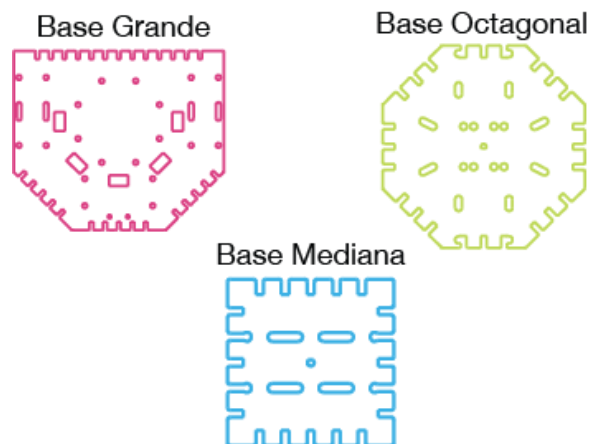
En la parte mecánica el briko incluye diferentes piezas de construcción para crear proyectos y darle estructura a la propuesta de ideas, las cuales vamos a ensamblar utilizando muescas.



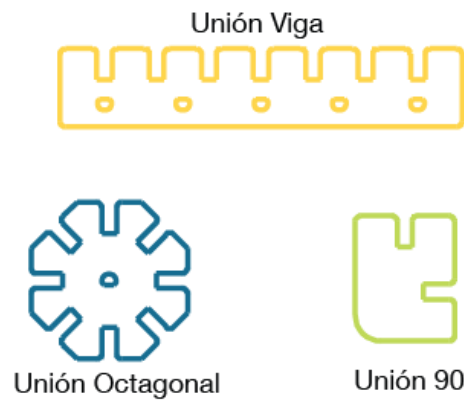
La ventaja de los brikos es que se pueden unir con las piezas del kit y prácticamente con cualquier material de 2 formas diferentes:

- Utilizando tornillos y tuercas
- Utilizando remaches plásticos

Las bases son donde se van a montar los brikos



Las uniones las vamos a utilizar para unir las bases entre sí



Es importante que conozcas la intención que tuvieron los creadores de Briko, en nuestro país normalmente nos hemos convertido en consumidores de tecnología pero muy pocas personas se dedican a desarrollo de nuevas propuestas o productos, esta situación nos coloca en un gran atraso tecnológico comparados con otros países.

El sistema educativo actual en nuestro país se ha encargado que la programación, electrónica y robótica se vean como áreas complejas y difíciles de aprender, eso ha inspirado a un grupo de emprendedores para que generen Briko y permitan a las nuevas generaciones un aprendizaje divertido de las áreas mencionadas.

Si recuerdas el concepto de Empresa 4.0 podemos predecir que dentro de pocos años muchos de los trabajos actuales tienden a desaparecer o ser sustituidos gracias a la inteligencia artificial, sin embargo, no significa que ya no tendremos empleo, sino que debemos desarrollar nuevos conocimientos y habilidades posiblemente incluso en áreas hasta el día de hoy desconocidas, pero si algo podemos afirmar sin temor a equivocarnos que estas áreas incluyen robótica y programación. Es por ello que debemos iniciar desde niveles básicos para el desarrollo de habilidades es por ello que nace Briko para contribuir a la educación en este tipo de áreas.

Vamos a utilizar el Kit de Briko durante nuestro curso para poder aprender de forma divertida a integrar el hardware y software, desarrollar la creatividad y el pensamiento lógico. Con lo anterior queremos contribuir a una nueva generación de profesionistas que puedan ser creadores y no consumidores de la tecnología.

## LABORATORIO 1

Vamos a iniciar con brikode, como mencionamos antes es una aplicación que se descarga de forma gratuita.

La sintaxis de un programa se refiere al orden y estructura que tiene que llevar el código para que funcione de forma correcta. Existen 3 reglas principales:

- Toda instrucción debe terminar con “;” con esto logramos que la computadora y el BK7 se enteren que se ha terminado una línea de código, existen algunas excepciones que mencionamos más adelante.
  - Incorrecto: `luces.color(RED)`.
  - Correcto: `luces.color(RED);`
- El código es sensible a mayúsculas y minúsculas, es decir son tres cosas distintas: Hola, Hola, hola.
- Los paréntesis ( ), llaves { } y corchetes [ ], son muy utilizados en programación cada uno con un propósito diferente, como referencia siempre van por pares y debes asegurarte de que así suceda.

Lo primero que hay que hacer antes de utilizar cualquier briko es darlo de alta o bien, declararlo. Es decir, decirle al BK7 que brikos se van a utilizar con los siguientes pasos:

1. Seleccionar el briko que se desea utilizar y buscar el nombre con el que se reconoce en la siguiente tabla

| Briko                 | Nombre                     |
|-----------------------|----------------------------|
| Perilla               | <code>knobbk</code>        |
| Sensor de distancia   | <code>distancebk</code>    |
| Botones               | <code>buttonsbk</code>     |
| Sensor de temperatura | <code>temperaturebk</code> |
| Bocina                | <code>buzzerbk</code>      |
| Sensor de luz         | <code>lightbk</code>       |
| Display               | <code>displaybk</code>     |
| Leds                  | <code>ledsbk</code>        |
| Motor                 | <code>motorbk</code>       |

2. Asignar un nombre con el cual lo vas a poder manejar durante el programa, la regla para asignación de nombres es no poner acentos, iniciar con números o caracteres especiales.
3. Finalmente, hay que indicar el puerto en el que se encuentra conectado el briko al Bk7 utilizando la palabra `PORT`, seguida por el número particular que se está utilizando. La sintaxis completa sería la siguiente:
  - a. `ledsbk luces (PORT1);`

Seleccionas el briko que quieres usar y buscamos el nombre de la tabla anterior, se le asigna un nombre con el cuál se va a referir al briko durante el programa y finalmente se le indica al Bk7 el puerto en el que se asigna el briko.

El ejemplo anterior nos indica que se da de alta un briko de leds (`ledsbk`) con el nombre de “`luces2`” conectado en el puerto 1.

### RETO 1:

Escribe la línea que debería dar de alta un briko de leds con el nombre “focos” en el puerto 4.

### RETO 2:

Da de alta un briko de bocina con el nombre “bocina” en el puerto 6.

En ambos casos no podemos verificar el código en brikode porque no cumple con la estructura básica de un programa.

### ESTRUCTURA BÁSICA DE UN PROGRAMA

Los códigos de brikode se dividen en dos partes:

1. Área de inicialización: Es la parte del programa donde se dan de alta los brikos y variables, se inicializa todo aquello que se va a utilizar en el código.
2. Bloque de code: En esta parte vamos a escribir la secuencia de instrucciones de nuestro programa, todas las instrucciones que incluye el code van dentro de llaves { }

Las instrucciones contenidas dentro del code es la que se repite en un loop infinito. Es importante considerar que la programación de los brikos se realiza en forma secuencial, es decir primero la línea 1 posteriormente la 2 por lo que es de gran importancia que previo a la programación desarrollemos la lógica de nuestro programa.

```
code(){  
  
}
```

### RETO 3

Dar de alta un briko de bocina con el nombre “bocina” en el puerto 2 y escribir la estructura básica de un programa.

---

---

---



Una vez que hemos dado de alta el briko ya es posible utilizarlo dentro del programa, es importante conocer que cada briko cuenta con funciones de las cuales ya podemos hacer uso siempre y cuando se respete la siguiente estructura:

Nombre.funcion(parámetros);

Nombre: La asignación de nombre que se la ha dado al objeto.

Función: Las funciones que existen disponibles es posible verificarlas en el siguiente link <http://briko.cc/referencia>.

Parámetros: Los datos que vamos a controlar del briko siempre van entre paréntesis ( ).

Ejemplo:

Vamos a iniciar utilizando el briko de LEDs, el cual cuenta con una función llamada color, al aplicarla nos permite encender los cinco leds del briko en el color seleccionado, el formato de la función es ".color(COLOR)", el parámetro COLOR debe estar en inglés y en mayúsculas para que pueda ser interpretado.

Para poder dar de alta un briko de LEDs con el nombre de "luces" para que puedan prender en rojo nuestro código queda de la siguiente forma:

```
ledsbk luces(PORT1);

code(){
  luces.color(RED);
}
```

Es importante que siempre recuerdes lo siguiente: Terminar las instrucciones con ";". Se debe separar el nombre del briko del nombre de la función con ".", además cerrar los paréntesis.

En el siguiente ejemplo se va a encender el led 1 en color azul:

```
ledsbk luces(PORT1);

code(){
  luces.color(1,BLUE);
}
```

La función puede ser ejecutada con RGB en donde R, G y B es la intensidad de rojo, verde y azul que debe contener el color y los valores van de 0 a 255.

En el siguiente ejemplo podemos visualizar que se prenden en un color verde aqua por la forma en la que están combinadas las intensidades de cada color (verde y azul) en el puerto 1.

```
ledsbk luces(PORT1);

code(){
  luces.color(0,255,255);
}
```

Reto: Dar de alta un briko de leds con el nombre de “luces” en el puerto 1 y después hacer que los 5 LEDs prendan en color verde utilizando la función normal de color sin RGB.

---

---

---

En el siguiente ejemplo solo prende el led 1 en color azul.

```
ledsbk luces(PORT1);
code(){
  luces.color(1,BLUE);
}
```

Reto: Dar de alta un briko con el nombre de “luces” en el puerto 1 y posteriormente prender el led 1 en verde, el led 2 en rojo y por último el led 3 en azul, utilizando la función color y sin utilizar la opción RGB.

---

---

---

---

---

---

Ahora utilizaremos el briko de bocina, para utilizar el briko de bocina utilizaremos la función “set(ESTADO)” en la cual los estados son prendida “ON” o apagada “OFF”. En el siguiente ejemplo vamos a dar de alta el briko de bocina en el puerto 2 con el nombre “bocina” y lo vamos a prender

```
buzzerbk bocina(PORT2);

code(){
  bocina.set(ON);
}
```

Si quieres que la bocina deje de sonar puedes utilizar el botón de pausa del Bk7 y seguir programando sin ningún problema.

*Para conocer el resto de funciones que puedes utilizar con la bocina puedes consultar <http://briko.cc/briko-bocina>.*

#### RETO 4:

Dar de alta un briko bocina con el nombre “bocina” en el puerto 2, después utilizando únicamente la función “play” haz que la bocina toque la nota “D2” por 1000 milisegundos y se apague por otros 500 milisegundos; para mayor información de las funciones puedes verificar <http://briko.cc/referencia>.

---

---

---

---

---

---

#### PAUSAS EN EL CÓDIGO: DELAY

En algunas ocasiones vamos a necesitar que el programa lleve a cabo pausas antes de realizar la siguiente acción, por ejemplo cuando queremos que prenda un led necesitamos que permanezca prendido cierto tiempo y posteriormente otro tiempo apagado, para llevar a cabo esta acción utilizamos la función “delay” la cual se escribe de la siguiente forma:

```
delay(tiempo);
```

La función hace que el programa se detenga una cantidad específica de tiempo en milisegundos, es decir si escribimos 1000 estamos esperando 1 segundo para que lo consideres en tus desarrollos.

La función se debe escribir posterior a la línea o líneas que deseamos pausar, por ejemplo en el siguiente código vamos a hacer que los leds parpadeen de color “azul”.

```
ledsbk luces(PORT1);
```

```
code(){  
  luces.color(BLUE);  
  delay(1000);  
  luces.color(BLACK);  
  delay(1000);  
}
```

Recordando que en nuestro briko nuestro código se ejecuta en forma secuencial lo que va a realizar el programa es lo siguiente:

1. Prende los leds en color azul
2. Se espera un segundo
3. Apaga los leds
4. Se espera otro segundo

Al terminar la última línea el Bk7 vuelve a comenzar con la primera línea que se encuentra dentro del “code”.

#### RETO 5:

Da de alta un briko de leds con el nombre “luces” en el puerto 1 y después haz que parpadeen los 5 leds en color “naranja” utilizando la función “color” (no utilices el color en sus funciones R, G, B). Utiliza delays de 1000 milisegundos.

---

---

---

---

---

---

#### RETO 6:

Dar de alta un briko de leds con el nombre de “luces” en el puerto 1 y un briko de bocina con el nombre de “bocina” en el puerto 2, después utilizando la función “color” para prender los leds en verde y luego que prenda la bocina con la función “set”, que dure así por 1500 milisegundos y posteriormente apagar los leds por 500 milisegundos finalmente que apague la bocina y que espere 1 segundo antes de repetir todo el código.

---

---

---

---

---

---

## COMENTARIOS

Normalmente en el desarrollo de todos los proyectos de tecnología se busca documentar la lógica con la que fueron creados, el comentario es una herramienta que va a servir para explicar en palabras simples lo que sucede con el código, nos ayuda a que cualquier persona pueda dar seguimiento o continuidad a nuestros proyectos y sobre todo a trabajar de forma colaborativa en algunos de ellos que implican un grado de complejidad mayor. El Bk7 ignora las líneas de comentarios solo ejecuta las instrucciones.

En bkickcode existen dos formas de poner comentarios:

Utilizando "//" antes de cada línea de comentario, o bien, "/\* \*/" en este caso se abre y se cierra el comentario y puedo escribir varias líneas. Ejemplo:

```
code(){
  //comentario de un renglon.
  /*
  Todo lo que
  esta entre los
  '*/ es un
  comentario
  */
}
```

## RETO 7:

Da de alta un briko de leds en el puerto 1 con el nombre "**luces**", en la misma línea donde dimos de alta el briko, escribiremos un comentario utilizando "//" que diga: "Declaracion de briko", después haz que el briko de leds prenda el led número 1 de color morado utilizando la función "color" ( no utilices los valores en R,G,B), y en las siguientes 2 líneas escribe un comentario utilizando "/\* \*/" que diga: "Aqui prendemos (en la primera linea) el led 1 en morado (en la segunda línea)", **sin acentos**.

---

---

---

---

---

---

---

---

## VARIABLES:

Una variable es un valor que puede cambiar a lo largo de la ejecución del programa, funciona como la memoria de nuestro programa.

Existen diferentes tipos de variables y las más comunes son las siguientes

| Tipo de variable | Valor que se puede almacenar             |
|------------------|--|
| <b>int</b>       | Números enteros (negativo o positivo)    |
| <b>float</b>     | Números decimales (negativo o positivo)  |
| <b>char</b>      | Caracteres                               |
| <b>bool</b>      | Un estado (verdadero/true o falso/false) |

Para declarar una variable debemos escribir primero el tipo y posteriormente el nombre de la variable, ejemplo:

```
char caracter;  
int entero;  
bool estado;  
float decimal;  
  
code(){  
}
```

Para asignarle un valor a una variable utilizamos el signo "=", las variables pueden ser utilizadas en cualquier parte del programa y hacen que el código trabaje de forma más eficiente, por ejemplo si queremos hacer parpadear los leds en azul y controlar el tiempo de la función delay con una variable, podemos utilizar el siguiente código:

```
ledsbk luces(PORT1);  
int tiempo = 1000;  
  
code(){  
  luces.color(BLUE);  
  delay(tiempo);  
  luces.color(BLACK);  
  delay(tiempo);  
}
```

La ventaja de trabajar de esta forma es que si deseas realizar un tiempo en la espera solamente cambias el valor de la variable, algunos ejemplos son:

```
//caracter inicia con el valor d
char caracter = 'd';
//entero inicia con el valor 3
int entero = 3;
//estado inicia con el valor true
bool estado = true;
//decimal inicia con el valor 4.1
float decimal = 4.1;
code(){
}
```

En briko tenemos palabras reservadas que son utilizadas para usar las funciones de forma más fácil las principales las puedes encontrar en <http://briko.cc/palabras-reservadas>, las palabras también son conocidas como "define" nos va a servir para crear objetos con un **valor específico el cual no puede cambiar** durante todo el programa, a este tipo de valores se le conoce como **constantes**.

Debemos recordar que NUNCA debemos llamar a una variable o briko con el mismo nombre de una palabra reservada.

Para declarar un "define":

1. Escribir #define
2. Separar con un espacio
3. Darle nombre
4. Separar con un espacio
5. Asignarle un valor.

Es importante que al declarar un "define" recuerdes que NO debe utilizarse el ";" al final.

Ejemplo:

```
#define tiempo 1000
#define Nuevocolor 27,55,200
code(){
    luces.color(Nuevocolor);
    delay(tiempo);
}
```

## RETO 8:

Da de alta un briko de leds en el puerto 1 con el nombre "**luces**", después declara una variable tipo "**int**" con el nombre "**rojo**" y que inicie con el valor de "**255**", otra con el nombre "**verde**" y que inicie con el valor de "**0**" y una última con el nombre "**azul**" y que inicie con el valor de "**0**", después utilizando la función "color( );" y haz que prendan los 5 leds de color "**rojo**" (no utilices la palabra RED, utiliza las variables que declaraste arriba para lograrlo)"

---

---

---

---

---

## ESTRUCTURAS DE PROGRAMACIÓN

En brikode existen estructuras de programación que nos permiten realizar acciones más complejas, los más comunes son if, for, while y switch. Con ellas va a ser posible tomar decisiones y realizar acciones de acuerdo a ciertas condiciones, además de simplificar el código nos ayudan a automatizar los procesos.

La estructura "if" permite que una sección del código funcione **únicamente cuando la condición se cumpla**. Ejemplo:

```
if(Condición a cumplir){  
  
    /*Si la condición se cumple corre el código  
    adentro de los '{ }'*/  
  
}
```

Se utilizan los paréntesis para definir la condición y las llaves para definir la acción.

Para realizar las comparaciones es importante identificar los operadores de comparación

| Símbolo | Significado   |
|---------|---------------|
| ==      | Igual         |
| !=      | Diferente     |
| <       | Menor         |
| <=      | Menor o igual |
| >       | Mayor         |
| >=      | Mayor o igual |

Al utilizar cualquiera de estos operadores el resultado siempre es verdadero o falso.



Los brikos de entrada son excelentes para utilizar como condicionantes a la hora de usar estructuras, ya sea porque quieras que se enciendan los leds al presionar un botón o que el brillo de los LEDs cambie de acuerdo a un sensor de distancia.

Uno de los brikos más utilizados es el de botones porque nos permite interactuar con el usuario y controlar acciones. Una de las funciones que vamos a utilizar es la función `read()` con la cual nos va a **regresar** el número de botón que se encuentra presionado.

En el siguiente ejemplo cuando se presione el botón 1 los LEDs se van a encender en color azul:

```
ledsbk luces(PORT1);
buttonsbk botones(PORT2);

code(){
  if(botones.read() == 1){
    luces.color(BLUE);
  }
}
```

#### RETO 9:

Da de alta un briko de leds con el nombre "**luces**" en el puerto 1 y después un briko de botones con el nombre "**botones**" en el puerto 2, después con una estructura "if" haz que si presionamos el botón 3 prendan los leds utilizando la función "color" de color "**rojo**" (no utilices los valores en R,G,B)

---

---

---

---

---

Podemos tomar más de una decisión por ejemplo que al presionar cualquier botón menor a 3, se encienda la bocina y al presionar cualquier valor mayor o igual a 3, se apague:

```
buzzerbk bocina(PORT1);
buttonsbk botones(PORT2);

code(){
  if(botones.read() < 3){
    bocina.set(ON);
  }
  if(botones.read() >= 3){
    bocina.set(OFF);
  }
}
```

La estructura if...else permite tener bloques de instrucciones para cuando la condición es verdadera y también para cuando la condición es falsa. La estructura cumple con el siguiente formato

```
if(Condición a cumplir){  
  
    /*Si la condición se cumple corre  
    el código adentro de los '{ }' del if*/  
  
}  
else{  
  
    /*Si no cumplió la condición  
    corre el código adentro de los '{ }' del else*/  
  
}
```

Como ejemplo vamos a realizar un programa para que cuando se presione el botón 1 los leds se prendan en azul y si la condición no se cumple los leds se apaguen.

```
ledsbk luces(PORT1);  
buttonsbk botones(PORT2);  
  
code(){  
    if(botones.read() == 1){  
        luces.color(BLUE);  
    }else{  
        luces.color(BLACK);  
    }  
}
```

Existen operadores lógicos que nos sirven para invertir un estado o juntar dos o más condiciones en una sola estructura, en brikode se cuenta con 3 operadores lógicos:

1. Not ( ! ): Se utiliza cuando queremos que se tome el valor contrario del estado que tenemos !true para obtener un false.
2. And ( && ): Se utiliza cuando queremos que una acción se realice si las condiciones se cumplen
3. Or ( || ) Se utiliza cuando queremos que una acción sea realiza si cualquiera de las condiciones se cumple.

Un ejemplo del uso de los operadores lógicos es el siguiente:

```
buzzerbk bocina(PORT1);  
buttonsbk botones(PORT2);  
ledsbk luces(PORT3);  
  
code(){  
    if(!(botones.read() == 0)){  
        luces.color(BLACK);  
    }else{  
        luces.color(RED);  
    }  
}
```

Analiza con tu equipo qué acciones lleva a cabo el código y anótalo

---

---

---

#### RETO 10:

Da de alta un briko de bocina con el nombre "**bocina**" en el puerto 1 y después un briko de botones con el nombre "**botones**" en el puerto 2, también hay que declarar una variable llamada "**boton**" sin inicializarla en nada, después hay que hacer que el botón que presionemos se guarde en la variable tipo "int" "**boton**" (utilizando la función read() ) y si la variable esta entre 1 y 3 (incluyéndolos) que prendan la bocina y que si no presionamos nada o cualquier botón fuera de ese rango que apague la bocina (utiliza la función "set" para prender y apagar la bocina).

---

---

---

Es posible utilizar la versión de if anidado el cual debe cumplir con la siguiente estructura

```
if(Condición a cumplir){

    /*Si la condición se cumple corre
    el código adentro de los '{ }' del if*/

}else if(Segunda condición){

    /*Si la primera condición no se cumplió
    y la segunda condición se cumple corre
    el código adentro de los '{ }' del else if */

}else{

    /*Si no cumplió ninguna de las condiciones
    anteriores corre el código*/

}
```

La estructura “**for**” permite que un bloque de instrucciones se repitan un número de veces, previamente definido. Al terminar de ejecutarse se continúa con la ejecución del resto de instrucciones fuera del “for”. Para hacer uso de la instrucción es necesario hacer uso de variables, operadores de comparación y decrementos/incrementos, la estructura del “for” es la siguiente:

```
for(variable;condición;incremento/decremento){  
  
    //bloque de código que se repite  
  
}
```

La variable la vamos a utilizar para guardar el valor que se usará en la condición, esa condición determina el número de veces que se va a ejecutar el código, el incremento o decremento nos va a permitir afectar la variable para que en algún momento la condición permita que termine el ciclo.

Las operaciones aritméticas que se pueden llevar a cabo son las siguientes

| Carácter | Función         |
|----------|-----------------|
| +        | Suma            |
| -        | Resta           |
| *        | Multiplicación  |
| /        | División        |
| =        | Igualdad        |
| %        | División módulo |

Algunos ejemplos del uso de operadores

```
int var;  
  
code(){  
var= 2+4;    //aqui var tomara el valor de 6  
var= 3-2;    //aqui var tomara el valor de 1  
var= 4/2;    //aqui var tomara el valor de 2  
var= 4*3;    //aqui var tomara el valor de 12  
var= 5%2;    //aqui var tomara el valor de 1  
}
```

Es posible también utilizar operadores compuestos que nos van a permitir simplificar el código

| Operadores compuestos | Equivalencia |
|-----------------------|--------------|
| var+=10               | var=var+10   |
| var-=10               | var=var-10   |
| var*=10               | var=var*10   |
| var/=10               | var=var/10   |
| var%=10               | var=var%10   |

En los incrementos y decrementos podemos considerar que normalmente afectan a la variable de uno en uno aunque puede variar de acuerdo a la lógica del programa, pero para el primer caso también podemos simplificar la acción de la siguiente forma:

| Incremento/decremento | Equivalencia |
|-----------------------|--------------|
| <b>var++</b>          | var=var+1    |
| <b>var--</b>          | var=var-1    |

Como ejemplo vamos a tener un programa que utiliza un contador para controlar la cantidad de veces que se encienden los leds y cuando el ciclo termine se van a apagar.

```
ledsbk luces(PORT1);
int contador;

code(){
    for(contador=1;contador<6;contador++){
        luces.color(contador,BLUE);
        delay(1000);
    }
    luces.color(BLACK);
}
```

## RETO 11:

Da de alta un briko de leds con el nombre "**luces**" en el puerto 1, también hay que declarar una variable llamada "**contador**" sin inicializarla en nada, después utilizando la estructura "**for**" y la variable que declaramos, haz que prenda los leds en secuencia de color "**morado**", desde el led 1 al led 5 con una pausa entre led de 500 milisegundos, esto debe hacerlo sumando de 1 en 1 y mientras la variable sea menor a 6, terminando esa estructura "**for**" escribiremos una nueva utilizando la misma variable que prenda los leds en secuencia de color "**rojo**", desde el led 5 al led 1 con una pausa entre led de 500 milisegundos, esto debe hacerlo restando de 1 en 1 y mientras la variable sea mayor a 0 (utiliza la función "**color**" para prender y apagar la leds, no utilices los valores en R,G,B).

[illegible]

## SENSORES Y SU MONITOREO EN EL CÓDIGO

Un sensor es un dispositivo que nos ayuda a adquirir datos del medio ambiente, como temperatura, luz, sonido, etc. Un ejemplo de sensor en briko es el de distancia que nos ayuda a medir la distancia entre un sensor y un objeto, trabaja en un rango de 7 a 80 centímetros; para utilizar un briko de este tipo vamos a hacer uso de la función `read()`, con la cual vamos a poder leer la distancia y utilizar el valor para tomar decisiones dentro de nuestro programa.

Por ejemplo un programa en el que si el sensor detecta un objeto a menos de 20 centímetros prenda el led en rojo, si detecta un objeto entre 20 y 40 centímetros se prendan los leds en azul y si no detecta objetos a 40 centímetros o más se apaguen los leds.

```
ledsbk luces(PORT1);
distancebk sensor(PORT2);

code(){
  if(sensor.read() <= 20){
    luces.color(RED);
  }else if((sensor.read() > 20) && (sensor.read() <= 40) ){
    luces.color(BLUE);
  }else{
    luces.color(BLACK);
  }
}
```

El monitor serial es una herramienta que maneja dos funciones con las cuales se puede imprimir información o mandar información, hay que activarla en la opción de herramientas y sería útil si requerimos saber el valor que está leyendo el sensor de proximidad.

El plotter serial es otra herramienta más que nos va a servir con la función `print` a diferencia del monitor serial esta herramienta solo nos sirve para desplegar información numérica, sirve para graficar en tiempo real las variables.

## RETO 12:

Da de alta un briko de distancia con el nombre "**sensor**" en el puerto 1, después haz que se imprima en el plotter serial la distancia medida por el sensor en pulgadas. Después de mandar cada mensaje haz que se espere 250 milisegundos. Modifica el programa para que también trabaje con monitor serial.

---

---

---

---

---

## WHILE

Es una estructura que nos permite que se ejecute un bloque de código mientras se cumpla una condición, si la condición no se cumple o deja de cumplirse se deja de repetir el bloque de instrucciones. Al terminar de ejecutar el ciclo se ejecuta el código que se encuentra fuera de él. Para poder utilizarlo es necesario hacer uso de variables y operadores de comparación.

```
while(condición){  
  /*Si la condición se cumple corre  
  el código adentro de los { } una y otra vez  
  hasta que la condición se deje de cumplir*/  
}
```

Para dar un ejemplo de uso de la estructura vamos a hacer que los leds parpadeen en rojo cada segundo si no presionamos ningún botón, pero mientras el botón 1 se encuentre presionado los leds van a parpadear en azul:

```
ledsbk luces(PORT1);  
buttonsbk botones(PORT2);  
  
code(){  
  luces.color(RED);  
  delay(1000);  
  luces.color(BLACK);  
  delay(1000);  
  
  while(botones.read() == 1){  
    luces.color(BLUE);  
    delay(1000);  
  }  
}
```

## RETO 13:

Da de alta un briko de leds con el nombre "**luces**" en el puerto 1 y después un briko de botones con el nombre "**botones**" en el puerto 2, también hay que declarar una variable llamada "**botón**" sin inicializarla en nada, después hay que hacer que el botón que presionemos se guarde en la variable "**botón**" (utilizando la función read() ) y después escribe una estructura "while" y haz que si la variable está fuera del rango de 2 al 4 que entre a la estructura y haga parpadear los 5 leds en morado(solo prende y paga los leds una vez y utiliza un delay de 500 milisegundos entre instrucciones y otro después de apagar los leds), y que si la variable está en el rango de 2 al 4 salga de la estructura "while" y apague los leds(utiliza la función "color" para prender y apagar la leds, no utilices los valores en R,G,B).

---

---

---

---

---

## SWITCH CASE BREAK

En algunas ocasiones los if's anidados se pueden volver bastante complejos por lo que existe una estructura que permite simplificar su implementación, el "switch" analiza una variable y según su valor realiza cierta acción. La estructura se encuentra formada por diferentes casos (case) que se ejecutan dependiendo del valor de la variable que seleccionamos, cada caso debe terminar con una instrucción break, que permite que el programa salga de la estructura switch y siga ejecutando el resto del código. Si la variable no cumple con ningún caso se ejecuta el bloque de "default".

```
switch(variable){  
  
    case valor1: //caso 1  
        accion1;  
        break;  
    case valor2: //caso 2  
        accion2;  
        break;  
    case valor3: //caso 3  
        accion3;  
        break;  
  
    //...N número de casos  
  
    default: //si NO se cumple ningun caso realiza este bloque  
        accion_default;  
        break;  
}
```

En el primer ejemplo se van a encender los leds azules si se presiona el botón 1, verdes si es el botón 2 y si no se presiona ninguno o cualquier otro botón que se apaguen.

```
ledsbk luces(PORT1);  
buttonsbk botones(PORT2);  
int boton;  
  
code(){  
    boton = botones.read();  
    switch (boton){  
        case 1:  
            luces.color(BLUE);  
            break;  
        case 2:  
            luces.color(GREEN);  
            break;  
        default:  
            luces.color(BLACK);  
            break;  
    }  
}
```



## RETO 14:

Da de alta un briko de leds con el nombre "**luces**" en el puerto 1 y después un briko de botones con el nombre "**botones**" en el puerto 2, después utilizando la estructura "switch" (utilizando directamente la lectura de los botones en el parámetro de la estructura sin crear ninguna variable) haz que si presionamos el botón 1 o el botón 3, prenda los leds en azul, si presionamos el botón 2 o el 5 que prendan los leds en rojo y que si no presionamos nada o cualquier botón que no sean esos que apague los leds (utiliza la función "color" para prender y apagar la leds, no utilices los valores en R,G,B).

---

---

---

---

---

## ARREGLOS

En algunas ocasiones en los programas es necesario utilizar más de una variable al mismo tiempo, para solucionar este tipo de problemas existe una estructura que nos permite almacenar varios valores del mismo tipo. La forma de declarar un arreglo es la siguiente

```
int variable[tamaño] = {dato0,dato1,...};
```

Donde tamaño es la cantidad de datos del arreglo y dato0, dato1 son los valores que vamos a utilizar para inicializarlo. Es importante recordar que las localidades del arreglo van de 0 a tamaño-1. Ejemplo:

```
int variable[5] = {4,9,3,6,1};
```

```
//variable[0] tiene el valor de 4  
//variable[1] tiene el valor de 9  
//variable[2] tiene el valor de 3  
//variable[3] tiene el valor de 6  
//variable[4] tiene el valor de 1
```

En el primer ejemplo vamos a utilizar la estructura for para hacer que la bocina toque diferentes notas, el arreglo las va a tener almacenadas.

```
int notas[5] = {NOTE_B4,NOTE_C4,NOTE_D4,NOTE_E4,NOTE_F4};  
int contador;  
buzzerbk bocina(PORT1);  
  
code(){  
  for(contador=0;contador<5;contador++){  
    bocina.play(notas[contador],500,500);  
  }  
}
```

## RETO 15:

Declara un arreglo tipo int de "**tamaño 7**" con el nombre "**arreglo**" e inicialízalo con los valores 4,9,5,3,0,7,6 y declara una variable tipo int que se llame "**contador**" sin inicializarla en nada, después utilizando una estructura "for" y la función `bk7print(NUMERO)` haz que se imprima todos los valores del arreglo en orden y con una separación de 1 segundo cada vez que envía.

---

---

---

---

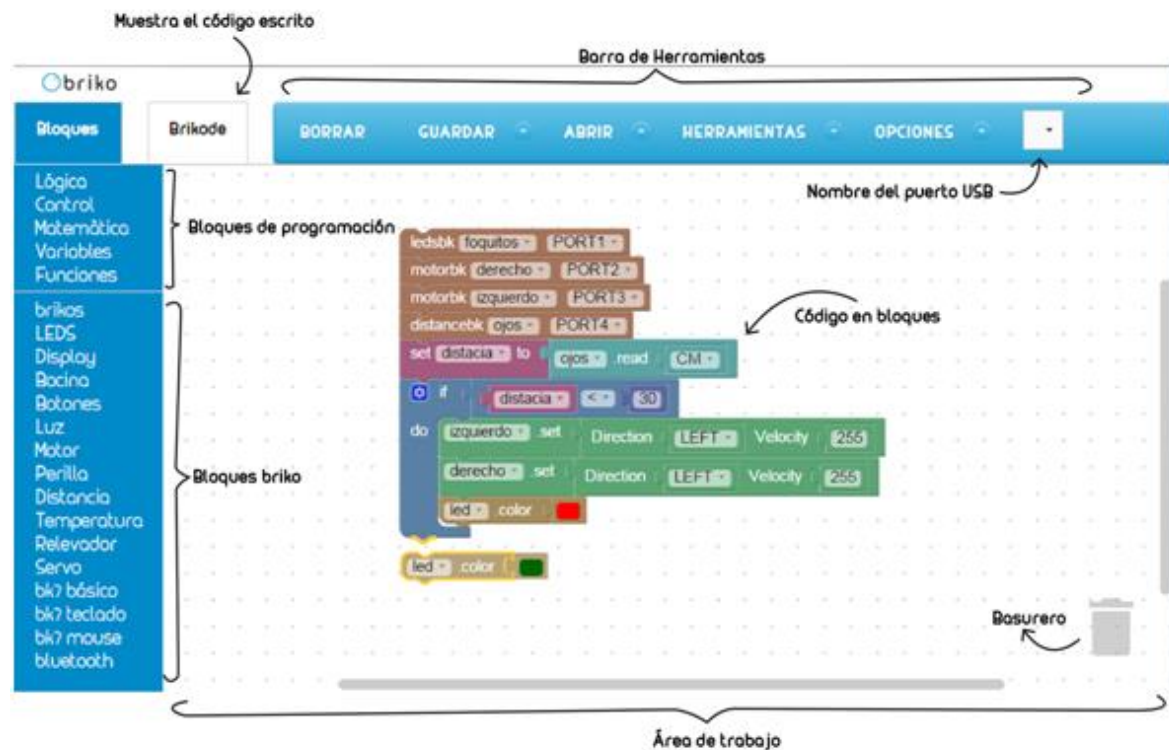
---

---

## PROGRAMACIÓN POR BLOQUES

Toda la información revisada previamente es posible realizarla en programación por bloques la herramienta se llama Brikoblocks. Este tipo de programación es muy sencilla y vamos a trabajarla en base a analogías de lo que ya hemos revisado en texto.

La interfaz de programación es la siguiente



Cada uno de los brikos revisados en la sección anterior tiene un bloque que les corresponde

| Briko                 | Bloque  |
|-----------------------|---|
| Perilla               |  |
| Sensor de distancia   |  |
| Botones               |  |
| Sensor de temperatura |  |
| Bocina                |  |
| Sensor de luz         |  |
| Display               |  |
| Leds                  |  |
| Motor                 |  |

Para dar de alta un briko debemos seguir 3 pasos:

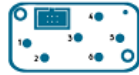
1. Arrastrar el bloque del briko que se va a utilizar al área de trabajo
2. Dale un nombre a tu briko, por ejemplo si estas utilizando un briko de leds para iluminar algún proyecto ponerle el nombre de “luces”.
3. Por último debes seleccionar el puerto en el que se encuentra conectado el briko.

Te pido que con tu equipo puedas explorar la herramienta seleccionar 3 de los retos de la sección anterior y poder convertirlos ahora en el formato de bloques.

Como cierre vamos a realizar un piano que nos permita integrar los conocimientos revisados al momento



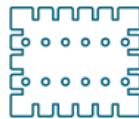
- 1 bk7 Controlador principal de briko



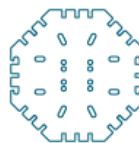
- 1 Botones Módulo de botones de briko



- 1 Bocina Módulo de bocina de briko



- 2 bases mediana base de briko



- 1 base octagonal base de briko



- 3 uniones 90  
unión de briko



- 3 uniones octagonal  
unión de briko



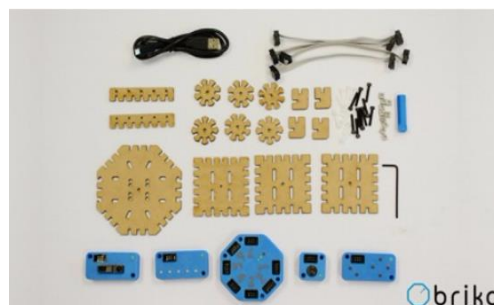
- 6 Tornillos chico  
Tornillo de briko



- 6 Tuercas  
Tuerca de briko



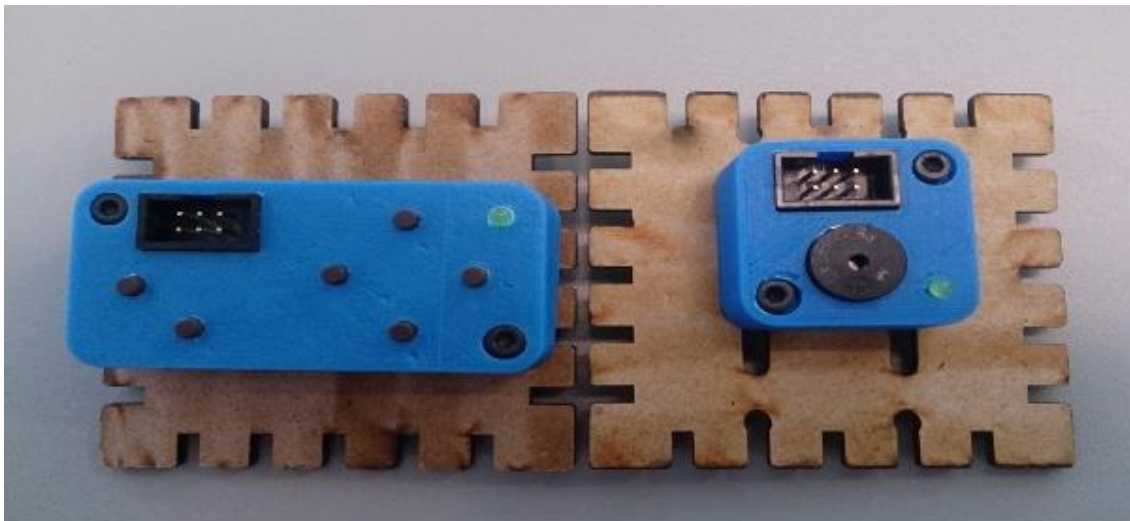
- 1 Cable Micro USB  
Cable para programar



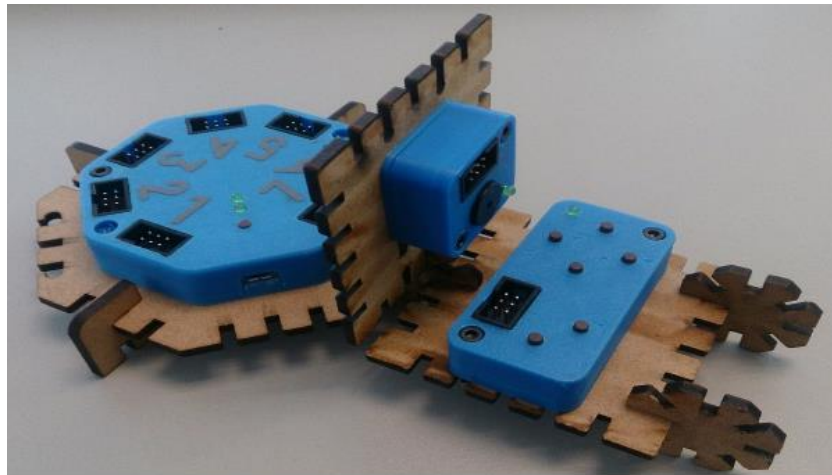
Paso 1:



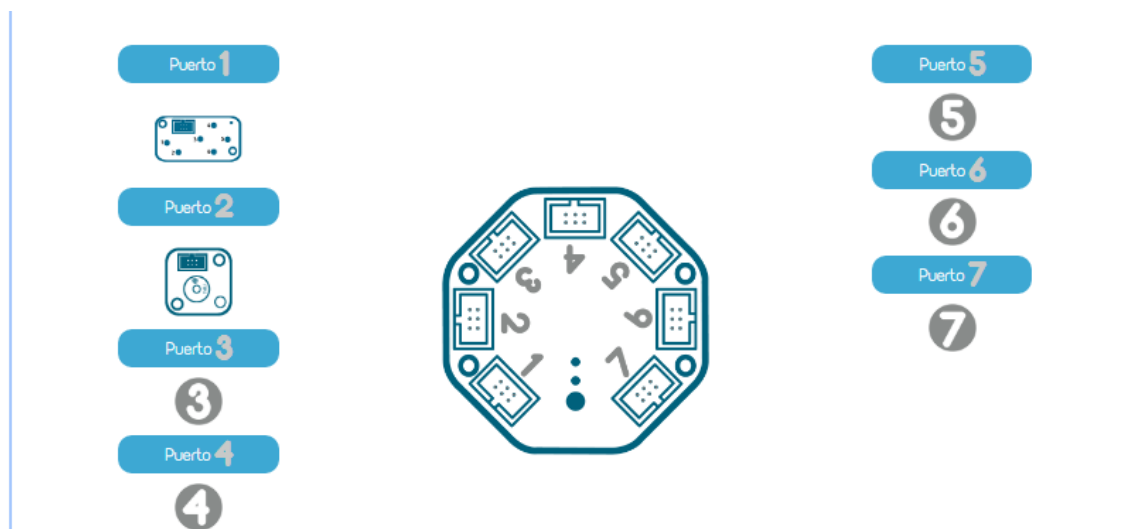
Paso 2:



Paso 3:



Paso 4: Conecta los brikos como se muestra a continuación



Paso 5:

Programa el siguiente código, o si lo prefieres utiliza el formato en bloques

```
// Aqui se declaran los modulos que vas a utilizar

buttonsbk botones(PORT1);      //Se declara un briko de botones en el puerto 1
buzzerbk bocina(PORT2);        //Se declara un briko de bocina en el puerto 2

int numero;                    //Se declara una variable tipo int llamada numero

code() {                       // Aqui se escribe tu programa

    numero = botones.read();    /*Se le asigna a la variable numero
                                el valor del boton presionado
                                que te manda el briko de botones*/

    if (numero == 0) {          //Si el valor guardado en la variable numero es cero
        bocina.set(OFF);        //La bocina se apaga
    }

    if (numero == 1) {          //Si el valor guardado en la variable numero es uno
        bocina.playtone(NOTE_DO); //La bocina toca la nota DO
    }

    if (numero == 2) {          //Si el valor guardado en la variable numero es dos
        bocina.playtone(NOTE_RE); //La bocina toca la nota RE
    }

    /*Despues de esto el programa vuelve a repetirse. Recuerda que el programa
    dentro de "code" siempre se vuelve a repetir hasta que apagues el controlador.+/>
}
```

Te pido que hagas 3 modificaciones al programa y las documentes en los siguientes renglones

---

---

---

---

---

---

---

---

---

---

## RETO FINAL

Vamos a desarrollar un carrito que pueda seguir una línea y como reto a modificar su programación para poder correr una carrera y que gane el mejor construido y programado.

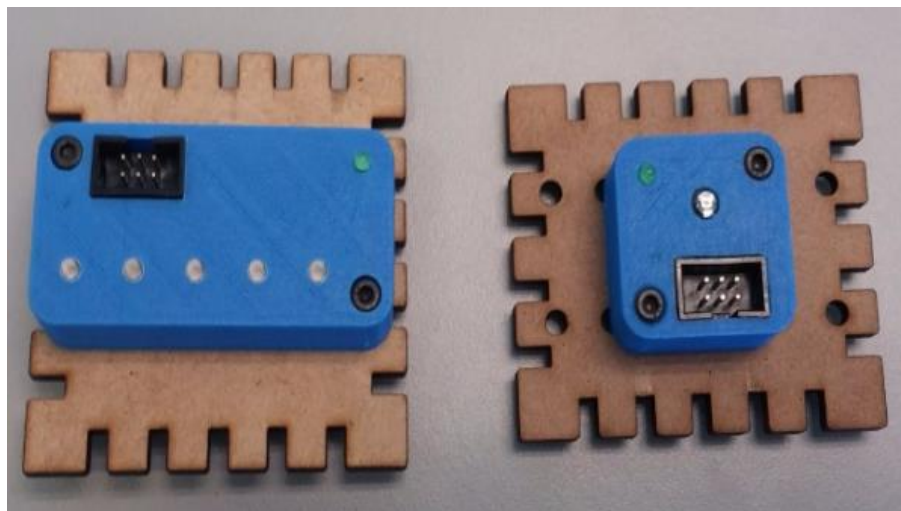
### Materiales

- 1 controlador principal de briko
- 1 módulo de leds
- 1 sensor de luz
- 2 módulos de motor
- 3 bases medianas
- 4 uniones de 90
- 8 uniones octagonales
- 1 base grande
- 2 llantas
- 6 tornillos chicos
- 4 tornillos grandes
- 4 cables planos
- 1 cable de Micro USB
- 1 batería

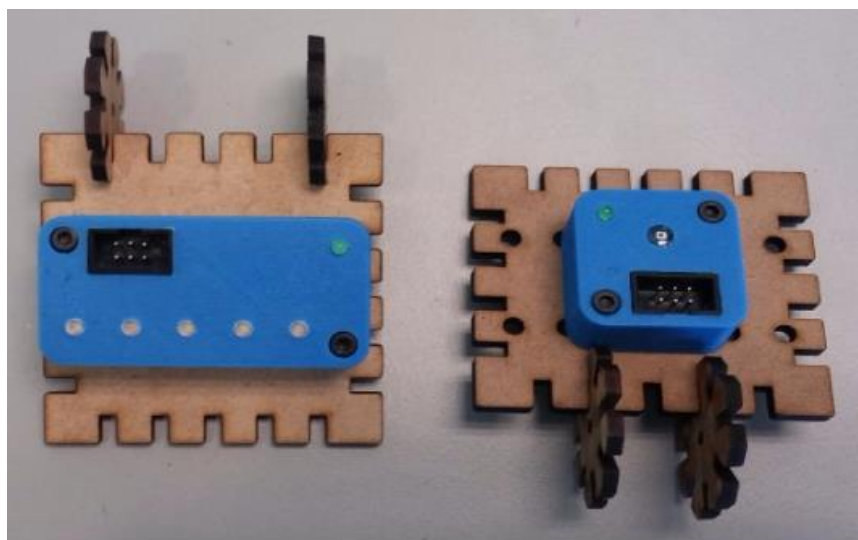


Se deben fijar el briko de leds y el de luz a las bases cuadradas

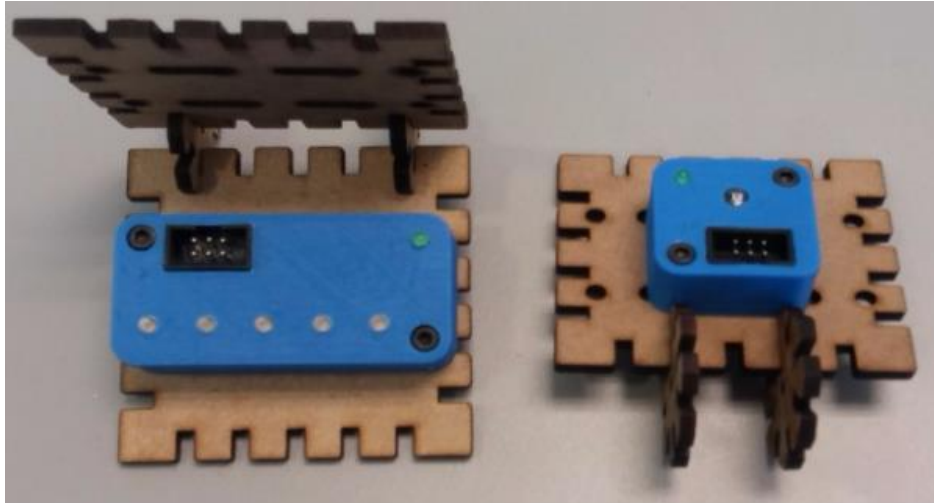




Posteriormente añadir las uniones como se muestra en la figura



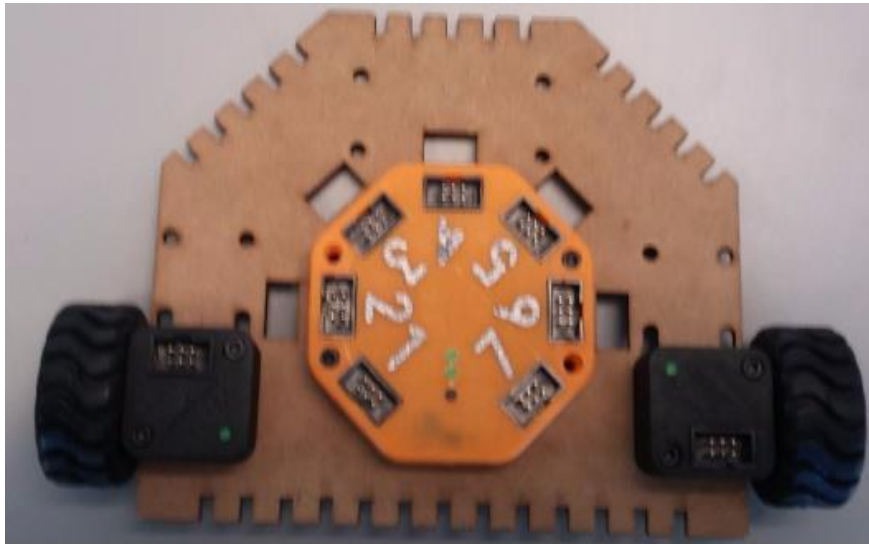
Se debe unir la siguiente base para obtener el siguiente resultado



Debes armar una especie de triángulo con las dos estructuras el resultado debe ser como el siguiente ejemplo



Vamos a ensamblar ahora la base grande y fijar cada motor con su respectiva llanta, para después añadir el bk7 al centro como se muestra



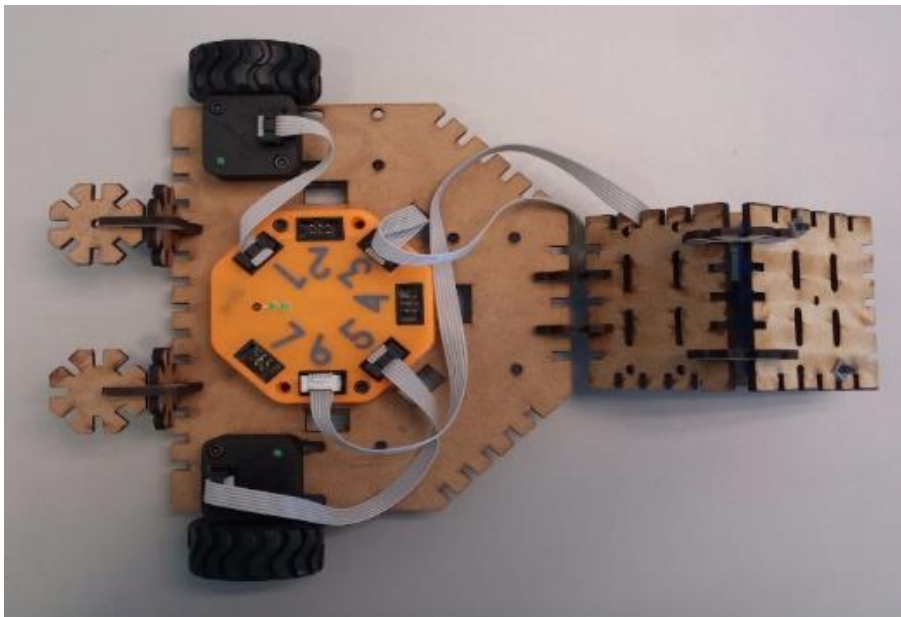
Ahora se debe realizar un pequeño soporte para la batería para lo que utilizaremos las bases octagonales y las uniones



Se debe unir la base con el bk7 y el soporte para la batería para obtener lo siguiente



Uniendo el total de ensambles debemos poder obtener el siguiente esquema



Las conexiones se deben establecer en el siguiente formato

- Motores en el puerto 1 y 5
- Puerto 3 sensor de luz
- Puerto 6 sensor de leds

Finalmente debes programar el briko con el siguiente código

```
motorbk motorizq(PORT1);
```

```
motorbk motorder(PORT5);
```

```
ledsbk leds(PORT6);
```

```
lightbk luz(PORT3);
```

```
int flag = 0; //declara una variable y la inicializa en 0
```

```
int sensor; //declara una variable
```

```
int rango; //declara una variable
```

```
code() {
```

```
// Aquí se escribe tu programa
```

```
if(flag == 0){ //este ciclo solo lo va hacer una vez
```

```
leds.brightness(255); //pone el brillo a máximo
```

```
flag=1; //hace la variable 1
```

```
leds.color(WHITE); //prende los leds en blanco
```

```
rango = luz.read(); //mide la intensidad de la luz para calibrar
```

```
rango = rango*1.5; //le da un tolerancia a la medición
```

```
delay(1000); //espera un segundo
```

```
}
```

```
sensor = luz.read(); //mide la intensidad de la luz en el ambiente
```

```
if(sensor>=rango){ //si lo que mide el sensor es mayor al rango que calibramos el robot  
gira
```

```
motorizq.set(LEFT,255); //hace que el motor izquierdo gire a izquierda
```

```
motorder.set(LEFT,0); //hace que el motor derecho no gire
```

```
}
```

```
else{ //sino el robot va al otro lado

    motorizq.set(RIGHT,0); //hace que el motor izquierdo gire a la derecha
    motorder.set(RIGHT,255); //hace que el motor derecho no gire
}

delay(50);          //Espera 50 milisegundos
}
```