ROBOT BB-8

Sistemas basados en microprocesadores

PAULA GÁRATE => INFORME
MARÍA GUTIÉRREZ=> INFORME
ALBERTO HERNANDO=> MECÁNICA
LAURA INÉS=>DISEÑO
BEATRIZ MARTÍNEZ=>PROGRAMACIÓN
MARIO PEÑALBA=>PROGRAMACIÓN
ADRIÁN TELLO=>GITHUB
CECILIA VECINO=> INFORME Y BLUETOOTH

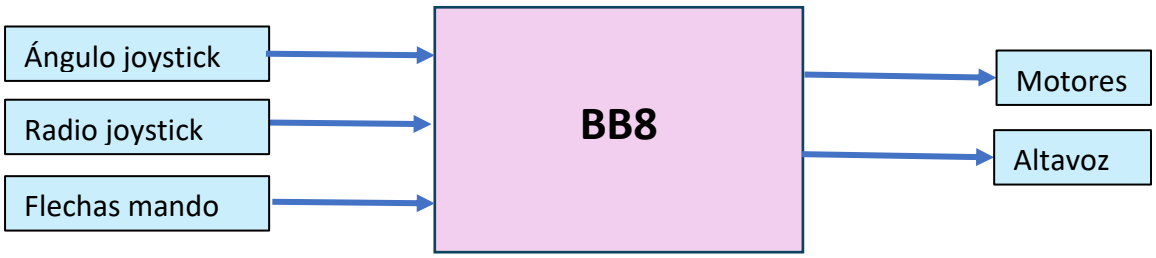
ÍNDICE

1. INTRODUCCIÓN

Durante este informe describiremos la programación y el montaje de un robot BB8 para conseguir el movimiento de este robot, hacia delante, hacia atrás, derecha e izquierda y diagonales con un control remoto joystick conectado a través de bluetooth.

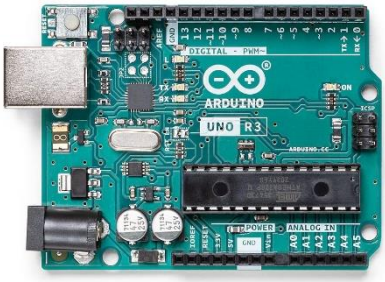
También hemos implementado junto con el código de programación el sonido característico de este robot.

2. DIAGRAMA DE ENTRADAS Y SALIDAS

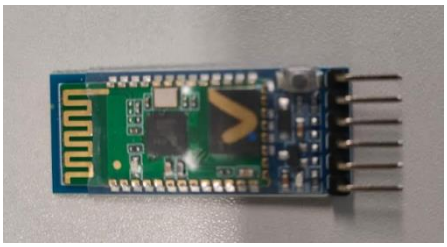


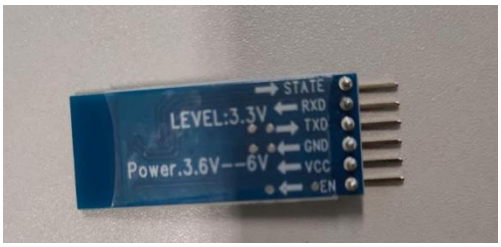
3. MATERIAL A EMPLEAR.

- Placa física Arduino Uno, sin shield

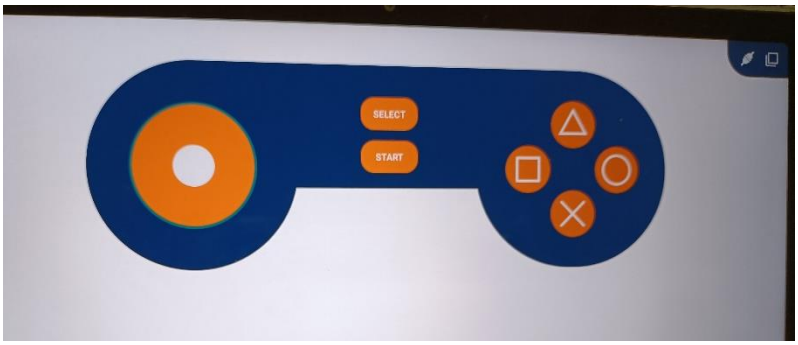


- Dispositivo HC05 para proporcionar comunicación bluetooth a Arduino

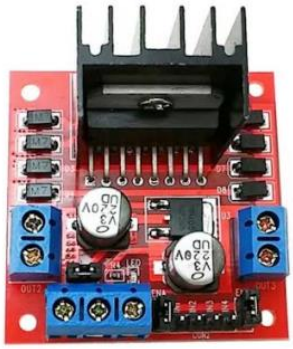




- Dispositivo con acceso a bluetooth con aplicación Dabble



- Driver L298N



- 2 pilas 9V

- 2 motores DC

- 2 ruedas

- Cable de conexión para conectar alimentación externa a Arduino.

- Cables conectores placa Arduino

- Regleta para empalmar cables

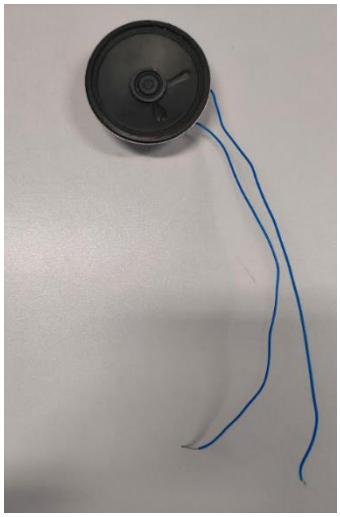
- Esfera y cuerpo interior del robot


- Cabeza del robot

- Imanes

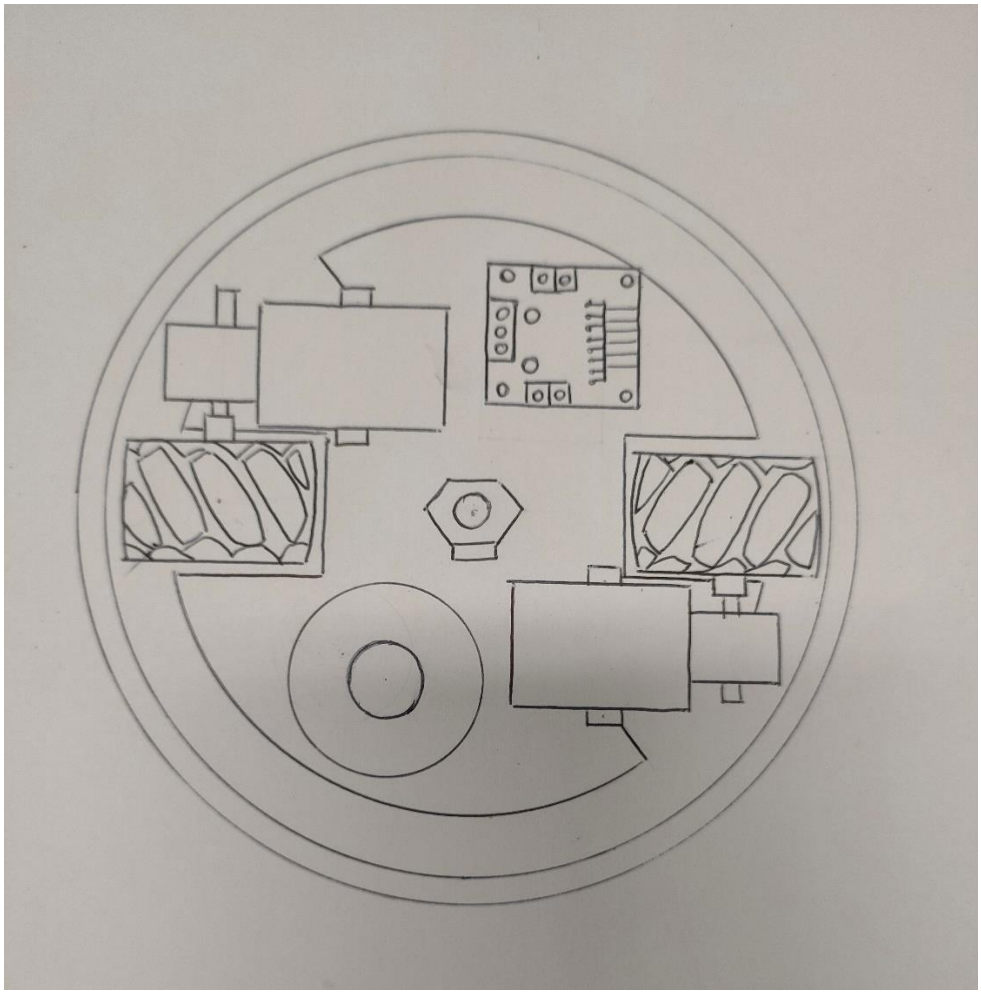

- Powerbank 5V para alimentación del sistema

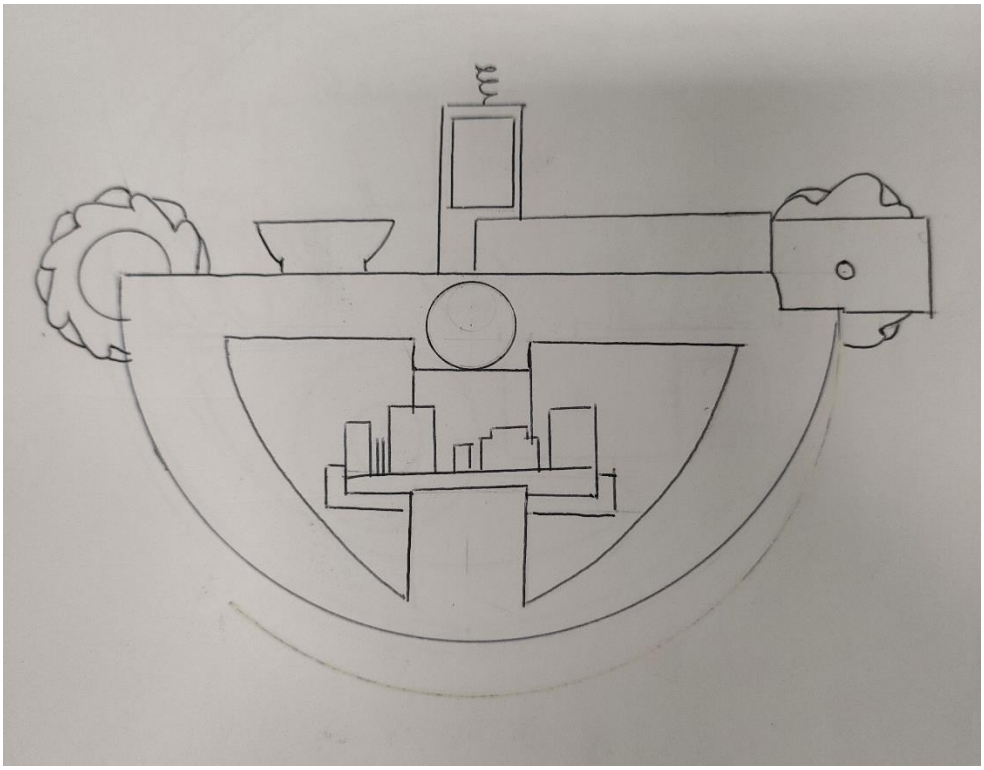
- Altavoz 8Ω 0.5W



- Contrapesos (tuercas)

4. MONTAJE





Nuestro montaje se basará principalmente en un esqueleto, este contará además con una serie de canicas en su superficie exterior para permitir que ruede por la cara interna de las dos semiesferas que compone nuestro BB8.

El esqueleto mencionado anteriormente cuenta con una serie de orificios en los que vamos a situar el resto de los elementos necesarios para el funcionamiento.

Comenzando por los elementos motrices, contamos con una estructura basada en la unión de dos motores unidos a dos ruedas diametralmente opuestas, el contacto de estas con la superficie se asegura a través de unos muelles.

Para alimentar los motores, vamos a utilizar dos pilas de 9V acopladas en paralelo para así aumentar la potencia y conseguir que las ruedas giren más rápido y con más par.

Centrándonos en la programación del droide, contamos con una placa de Arduino Uno alimentada por una batería portátil, así como con un driver que servirá de soporte para configurar el funcionamiento de los motores.

Finalmente, relativo a los cables y los métodos de conexión utilizaremos un cable USB para conectar la batería portátil a la placa de Arduino, un cable específico para unir los dos terminales de la batería con los motores y varios conectores macho-hembra para unir los diferentes pines, ya sean del driver, la placa o los motores.

5. DESCRIPCIÓN DEL PROGRAMA

- Introducción

Primero, se declaran los pines que utilizaremos para conectar nuestros dos motores. Para el caso del motor A, hemos utilizado los motores 4 y 5, mientras que, para el B, los pines 6 y 7.

La información que recibimos de la aplicación Dabble son el radio y el ángulo del joystick definida como GamePad.getAngle() y GamePad.getRadius() que lo asignaremos a las variables enteras ‘a’ y ‘b’ respectivamente.

El radio vendrá dado como un valor entre 0 y 7. Para controlar la velocidad del motor se usa el rango 0-255, siendo 0 velocidad nula y 255 máxima velocidad. Por ello, mediante una regla de tres asemejamos el radio obtenido con dicho rango de velocidades. (Radio 7 sería máxima velocidad, es decir 255). Los pines que utilizaremos para indicar la velocidad a los motores son los pines 9 y 10.

Declaramos las variables que vamos a utilizar. “vel_max” será la velocidad determinada por el módulo del joystick, mientras que “velA” y “velB” serán las velocidades asignadas a cada uno de los motores. Haremos uso de la variable “p” para calcular las velocidades de las ruedas cuando estas no tengan la velocidad directamente relacionada con el módulo del radio.

- Librerías

Dabble. h

- Máquina de estados

Para el desarrollo de este programa, utilizaremos una máquina de estados, por ello, declararemos nuestros 8 estados, siento estos el primer, segundo, tercer y cuarto cuadrante del sistema de coordenadas más el eje de abscisas cuando el valor de “x” es positivo, negativo y para el eje de ordenadas, cuando “y” es positivo y negativo.

Para establecer los cambios de estado en función de la posición del joystick declararemos condiciones en base a los ángulos que nos llevan a cada una de las funciones que definen el movimiento del robot.

- Velocidades

Antes de comenzar a explicar a explicar las funciones que definen los movimientos del motor cuando el joystick está situado en algunos de los cuadrantes, será necesario definir como toma la variable “p”, antes nombrada, los valores.

```
if(a==15 || a==165 || a==195 || a==345){
 p=1;}

if(a==30 || a==150 || a==210 || a==330){
 p=2;}

if(a==45 || a==135 || a==225 || a==315){
 p=3;}

if(a==60 || a==120 || a==240 || a==300){
 p=4;}

if(a==75 || a==105 || a==255 || a==285){
 p=5;}

}
```

Es necesario saber que la información sobre los ángulos que llega de la aplicación “Dabble” vendrá discretizada en ángulos de 15º.

Para los movimientos del robot cuando nos encontremos en alguno de los cuadrantes se basará en el movimiento máximo de una de sus ruedas, es decir, el módulo de la coordenada donde se sitúa el Joystick. Mientras que el otro motor girar a menor revolución para conseguir así un cambio de sentido de orientación del motor.

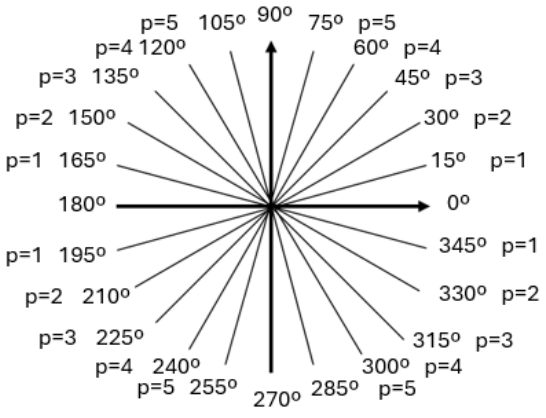
Trabajando en espejo respecto del eje X y del eje Y.

La variable “p” podrá tomar 5 valores distintos. En función de ángulo en el que se encuentre el joystick el giro del robot será más acentuado o menos.

```
if(a==15 || a==165 || a==195 || a==345){
 p=1;}
```

En el caso de que la información que recibimos sea uno de esos cuatro ángulos, se estará produciendo un cambio de sentido de robot, mientras avanza hacia delante o hacia atrás, muy brusco. Es por ello, que la rueda que se encarga de este giro, tomando un valor distinto a “vel_max”, deberá ser lo menor posible. Es por ello que “p” toma el valor 1.

De manera parecida ocurrirá con el resto de ángulo que puede tomar el joystick. A continuación, se adjunta una imagen explicativa sobre los valores que toma p en las distintas posiciones en el eje de coordenadas.



- Funciones

Se definen las funciones para el movimiento del robot que llamaremos después en nuestra máquina de estados.
A continuación, explicamos algunos de los movimientos del robot.

```
void f_arriba(){
 velA=vel_max;
 velB=velA;
 digitalWrite(motorApin1, HIGH);
 digitalWrite(motorApin2, LOW);
 digitalWrite(motorBpin1, HIGH);
 digitalWrite(motorBpin2, LOW);
 //mismo sentido
}
```

Este movimiento ocurrirá cuando nos encontremos justo en el eje de ordenadas. Para que el robot se mueva de manera recta, ambos motores deberán girar a la misma velocidad y en el mismo sentido. Por ello, para conseguir la misma velocidad asignaremos a ambos motores la misma velocidad que corresponde al módulo del joystick, es decir, vel_max. Para el mismo sentido de los motores asignaremos a ambos a los pines digitales de los motores la misma secuencia de valores digitales que debe tomar, primero HIGH y después LOW.

```
void f_izquierda(){
 velA=vel_max;
 velB=velA;
 digitalWrite(motorApin1, HIGH);
 digitalWrite(motorApin2, LOW);
 digitalWrite(motorBpin1, LOW);
 digitalWrite(motorBpin2, HIGH);
 / /sentido contrario
}
```

Para el movimiento exclusivo a la izquierda del robot, el joystick deberá está situado en la parte negativa del eje de abscisas. Conseguiremos este movimiento asignando a ambos motores la misma velocidad de giro que coincidirá con “vel_max” y girando en sentidos opuesto, por ello la secuencia de valores digitales que toman los pines de los motores serán opuestos.

```
void f_derecha(){
 velA=vel_max;
 velB=velA;
 digitalWrite(motorApin1, LOW);
 digitalWrite(motorApin2, HIGH);
 digitalWrite(motorBpin1, HIGH);
 digitalWrite(motorBpin2, LOW);
 //mismo sentido
}
```

El movimiento hacia la derecha del robot ocurrirá de manera análoga al movimiento del robot hacia la izquierda. La única variación que habrá será la secuencia de los valores digitales que toma los motores A y B, pues esta es inversa.

```
void f_izquierda_arriba(){
  velB=vel_max;
  velA=p/6*velB;
  digitalWrite(motorApin1, HIGH);
  digitalWrite(motorApin2, LOW);
  digitalWrite(motorBpin1, HIGH);
  digitalWrite(motorBpin2, LOW);
  //sentidos contrarios
}
```

Con este ejemplo, explicaremos el movimiento del robot cuando nuestro joystick se encuentra en uno de los cuadrantes. En este caso, nos encontraremos en el segundo cuadrante.
Para que pueda avanzar hacia adelante, el giro de los motores debe ser en el mismo sentido. Además, las velocidades de los motores deberán ser distintas para que gire a la vez que avance. El motor B girará a la velocidad definida por el módulo, mientras que, el motor A girará en función del valor que tome la variable “p”, la cual depende del ángulo tomado por el joystick.

- Sonido

Se definen las constantes correspondientes a cada nota en función de su frecuencia.
Declaramos en pin 13 como salida del altavoz y creamos la variable “sel” para elegir la melodía. Si pulsamos la flecha de arriba, sonará la melodía 1 y si pulsamos la de abajo, la melodía 2. Lo hacemos mediante las flechas, ya que es incompatible con el uso del joystick ya que cada melodía está formada por una secuencia de tonos definidos por su frecuencia y tiempo.

- Bluetooth

Para el control de nuestro robot utilizaremos un mando de videojuegos con un periférico de entrada, llamado joystick, que consiste en una “palanca” que gira sobre una base e informa su ángulo o dirección al dispositivo que está controlando. En nuestro caso, al tratarse de un teléfono móvil será accionado el movimiento por el dedo.

Utilizaremos la aplicación Dabble. Descargado el programa, seleccionamos “Gamepad”. Allí vincularemos nuestro Arduino a nuestro dispositivo móvil. Si todo es correcto, nos permitirá controlar de manera remota el movimiento de nuestro robot. Por defecto, el mando de videojuego nos saldrá en “Digital mode”, para poder manejarlo con “Joystick Mode” entraremos dentro de “Switch Mode” y seleccionaremos el modo deseado.

6. TABLA DE ESPECIFICACIONES TÉCNICAS

Tipo de robot	BB8
Peso	1.5kg
Alcance del bluetooth	10 metros
Velocidad	2 km/h
Espacio necesario	Ø195 mm
Grados de libertad	Movimiento en eje x e y en un plano
Tamaño de la caja de control (ancho x largo x alto)	175mmx175mmx170mm
Puertos de E/S de la placa Arduino	E: Pines digitales 2 y 3 S: Pines digitales 4,5,6,7,9,10 y 13
Fuente de alimentación de Arduino	Powerbank 5V
Comunicación	Chip Bluetooth HC-05
Programación	Interfaz Arduino IDE
Ruido	Relativamente silencioso
Consumo de energía	1000 mAh
Temperatura	El robot puede funcionar en un intervalo de 0 y 40ºC
Fuente de alimentación motores	2 pilas 9V, 170mA, 46gr
Vida útil aproximada	10000 horas
Cables	Cables entre driver y pila Cables entre driver y Arduino Cables entre driver y bluetooth Cables entre Arduino y bluetooth Cable de conexión entre Arduino y powerbank (20mm)

7. PROBLEMAS FRECUENTES Y SOLUCIONES.

- Estructura: las piezas impresas en 3D son demasiado rugosas y hay problemas de deslizamiento de las canicas sobra la esfera externa.
  Su solución sería con una lija dejarlo lo más liso posible.
- Peso: peso insuficiente, la estructura interna debe permanecer vertical porque solo se mueve la externa, cuando no hay peso suficiente la esfera tiende a moverse creando perturbaciones en el movimiento.
  Este problema lo solucionamos poniendo pequeños pesos repartidos en el interior en el núcleo con los demás componentes.
- Pilas: la potencia de la pila puede ser insuficiente provocando que las ruedas no tengan suficiente fuerza para girar la esfera.

Una posible solución es poner una pila de mayor potencia, o bien si no se dispone de ella, poner dos pilas en paralelo.

- Cabeza bb8: la cabeza diseñada va unida al cuerpo a través de un imán, si el imán no tiene suficiente fuerza no se mantiene en la posición superior de la esfera, por el contrario, si el imán tiene demasiada fuerza se quedaría en el mismo punto de la esfera externa y no podría rodar.
  Solución: escoger un imán con una fuerza acorde con nuestro diseño.
  Además, el peso de la cabeza también influye en la sujeción que ofrece el imán, ya que si es demasiado pesada con la fuerza de la gravedad caería por su propio peso.
- Espacio componentes: a la hora del montaje cuando se imprime el núcleo en 3D debe haber huecos suficientes para todos los componentes.
  Una solución es tener un buen diseño antes para que encajen todos los componentes en su hueco teniendo en cuenta también espacios para las conexiones de cables entre los distintos componentes.
- Bluetooth: vamos a usar la aplicación Dabble, esta tiene unos pines concretos para el bluetooth (2 y 3), y a la hora de conectarlo se hará directamente desde la aplicación no desde la configuración del dispositivo.
  Puede dar problema desde dispositivos IOS (Apple) por eso es preferible conectarlo con un Android.
- Muelles: para que las ruedas se ajusten bien a las paredes internas de la esfera utilizaremos unos muelles colocados en los motores. Será necesario que los muelles tengan la longitud correcta para que no se quede ni muy apretado ni con mucha holgura.
- Ruedas: las ruedas no tienen suficientemente fuerza para girar en algunos sentidos

8. LINK GITHUB

PCEDA (PCEDA) (github.com)