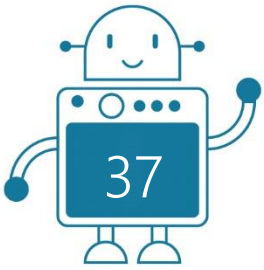


MBOT RANGER POTENCIÓMETRO ARDUINO

STEMJAM Teaching Guide

Developing make spaces to promote creativity around STEM in schools
Acronym: STEMJAM
Project no. 2016-1-ES01-KA201-025470

www.stemjam.eu

Co-funded by the Erasmus+ Programme of the European Union

MBOT RANGER POTENCIÓMETRO ARDUINO

RESUMEN

Usaremos el potenciómetro para:

- Controlar el Anillo LED del mBot Ranger (placa Auriga).
- Regular la velocidad de un ventilador, además de moverlo hacia los lados.

OBJETIVOS DIDÁCTICOS

- Utilizar el lenguaje Arduino para programar el mBot Ranger.
- Conocer el bucle *for*.
- Conocer diferentes funciones.

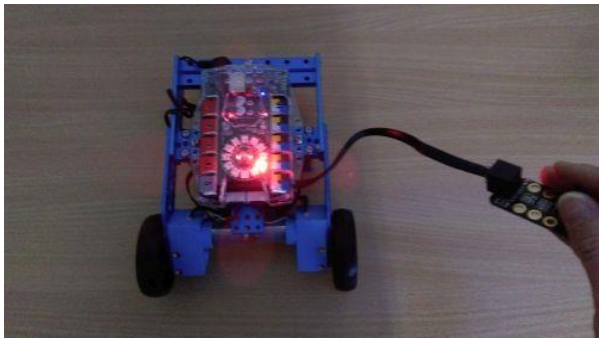
Materias STEM: Ciencia ☐ Tecnología ☐ Ingeniería☒ Matemáticas☒

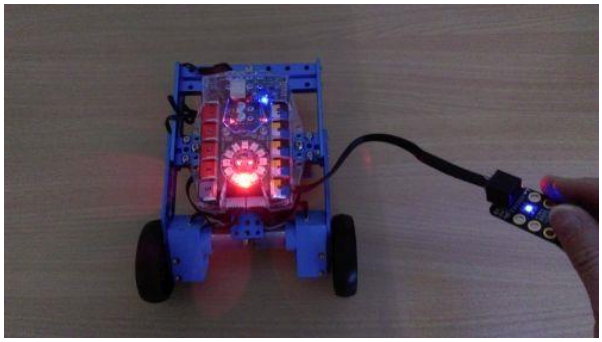
Nivel educativo: 12-14 años☐ 14-16 años☒

PLANTEAMIENTO DEL PROBLEMA

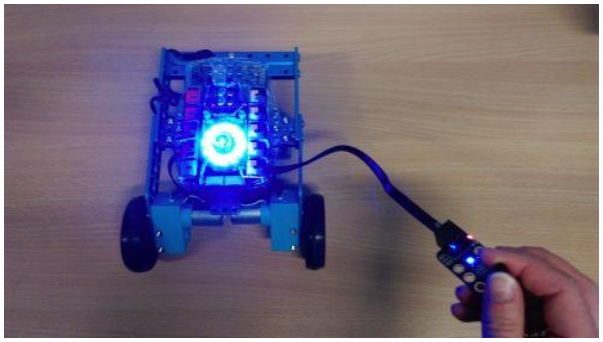
Algunos estudiantes no saben qué es un potenciómetro, por lo que a través de esta actividad sabrán más al respecto y podrán aplicarlo en diferentes usos diarios. Usamos el lenguaje Arduino para escribir el programa que controla el anillo de led en Ranger.

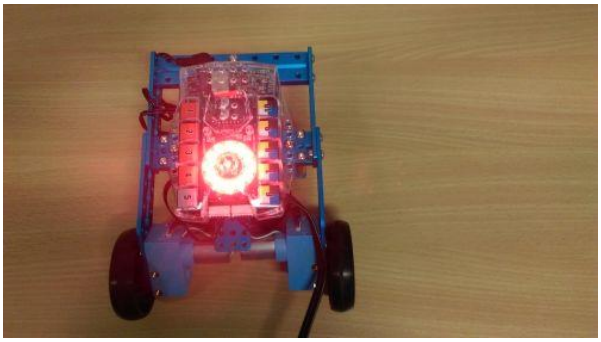
Propuesta 1: al mismo tiempo, solo un led brilla, pero cuando el usuario gira el potenciómetro, el siguiente led brilla:





Propuesta 2: Todos los leds se encienden en el mismo color, pero el color cambia cuando el usuario gira el potenciómetro.

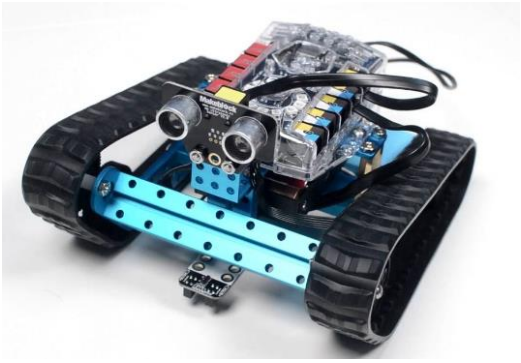




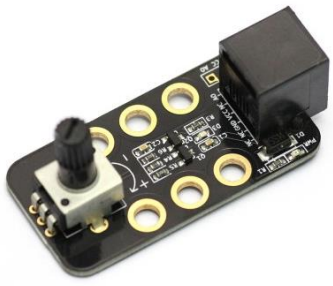
En la segunda parte, usamos el potenciómetro para controlar el ventilador. Podemos encenderlo y apagarlo, así como regular la velocidad.

LISTADO DE MATERIALES

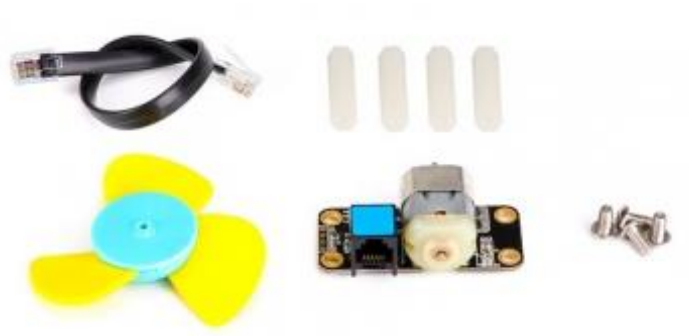
- El robot Ranger con anillo led:



- **Potenciómetro**:



❖ Pack de **Ventilador**:



❖ **Sensor de Ultrasonidos**:

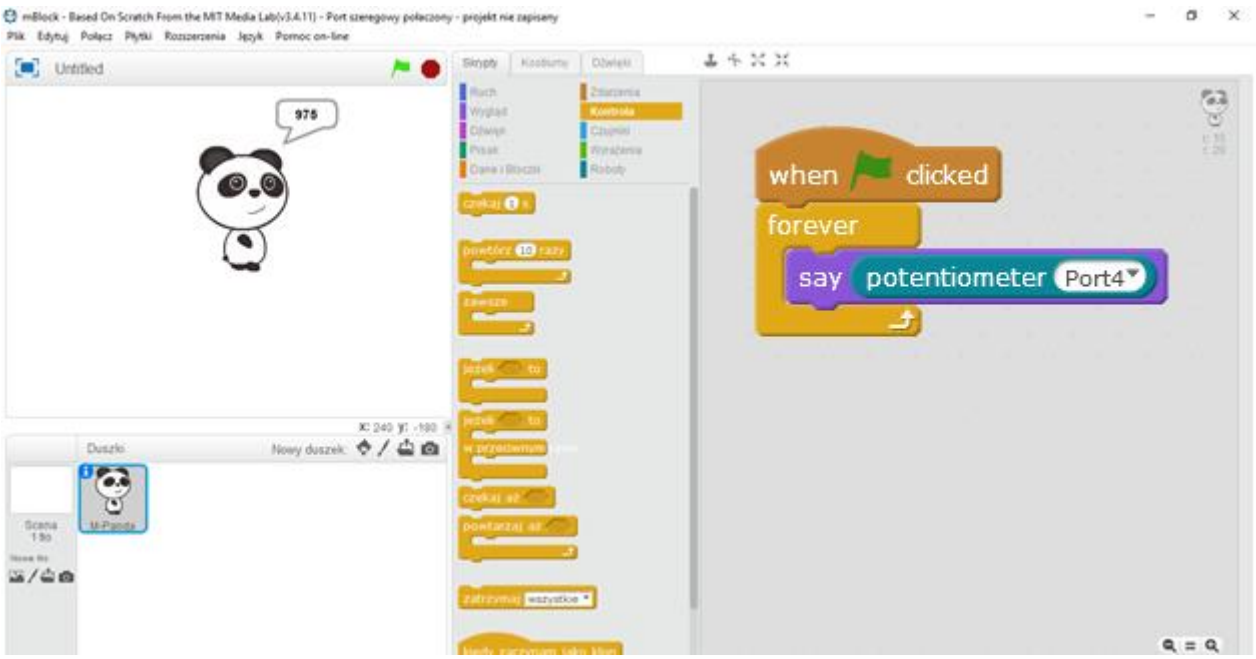
ELEMENT	ID	CABLE	AMOUNT	PORT 6				PORT 7				PORT 8				PORT 9				PORT 10				P.MOT1	P.MOT2
				Y	B	W	Bl	Y	B	W	Bl	Y	B	W	Bl	Y	B	W	Bl	Y	B	W	Bl	W*	W*
mBot Ranger			1																						
Motor 1	W*																							W*	
Motor 2	W*																								W*
Me RJ 25 adapter	Y																								
	B																								
	Bl																								
Mini Pan-Tilt kit It has 2 servos. We have to connect the servo to a RJ25 adapter																									
Mini Gripper We have to connect the servo to a RJ25 adapter																									
Me 7-Segment serial display	B																								
Me Led Matrix 8x16	B																								
Me Ultrasonic sensor	Y	(1)	1	Y																					
Me Temperature Sensor - Waterproof	Y																								
Me Line Follower	B																								
Me Potentiometer sensor	Bl	(1)	1												Bl										
Me TFT LCD Screen	W																								
Me Sound sensor	Bl																								
Me Touch sensor	B																								
Mini Fan Pack	B	(1)	1																		B				
Me Temperature and Humidity sensor	Y																								
Me 130 Motor Fan Pack	B																								
RJ25 cables			3																						
Structures and beams																									
Laptops			1																						
Attrezzo (not essential)																									

DESCRIPCIÓN DE LA ACTIVIDAD

Primera versión

Este es un potenciómetro de tipo dial de 50k con conocimiento que se puede girar hasta 270 grados. Puede convertir el movimiento giratorio en una entrada analógica que se puede usar para controlar la velocidad de un robot móvil, el brillo de los LED RGB u otros.

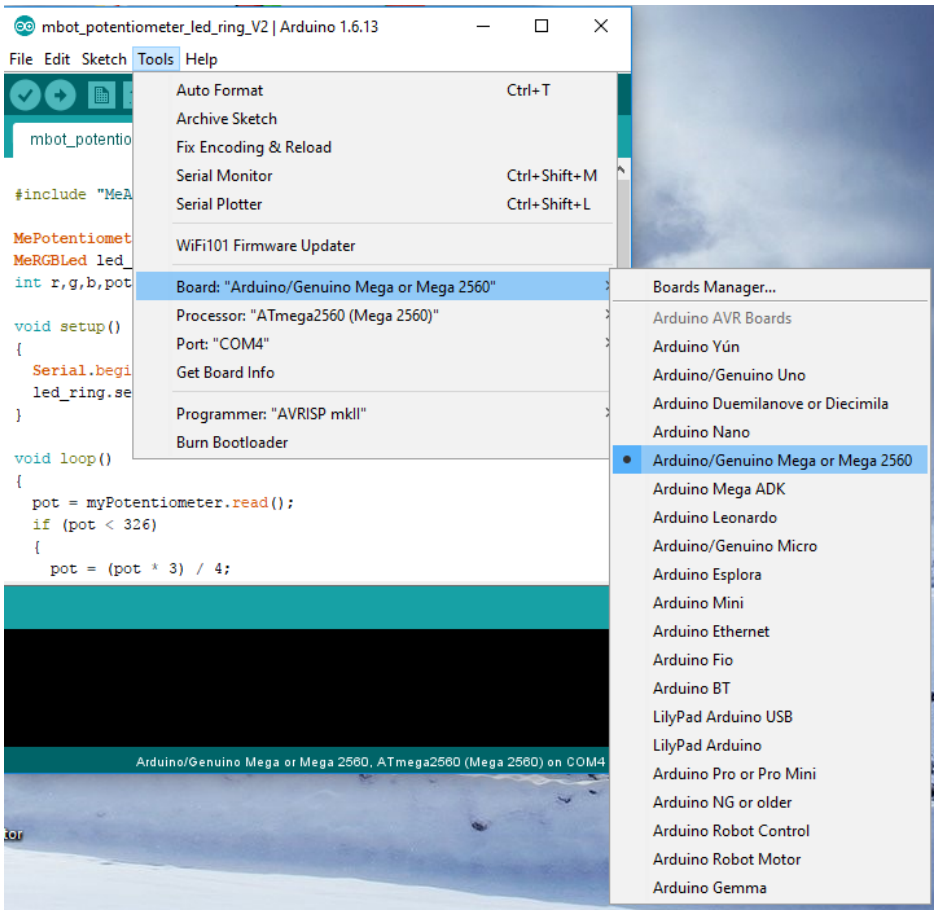
El tipo de señal es analógico. Rango de 0 a 970. Es fácil verificarlo en el programa mBlock.



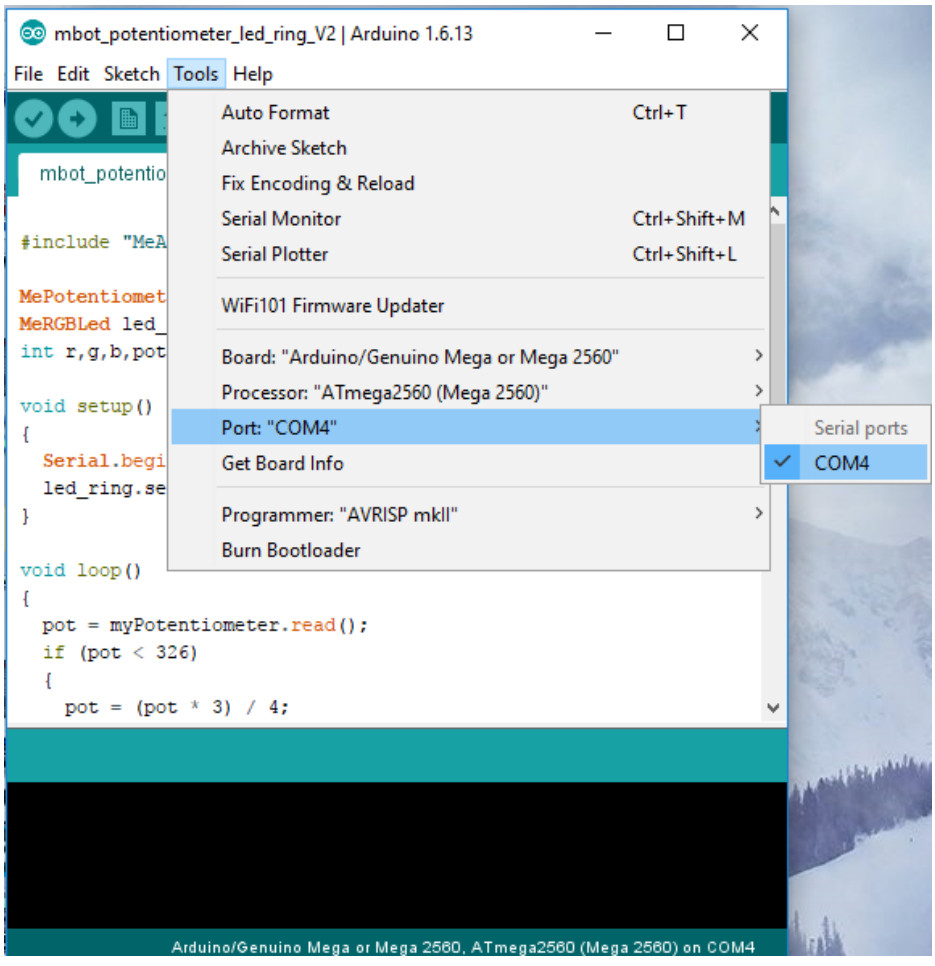
Arduino

El objetivo de la actividad es trabajar con el mBot Ranger en lenguaje Arduino.

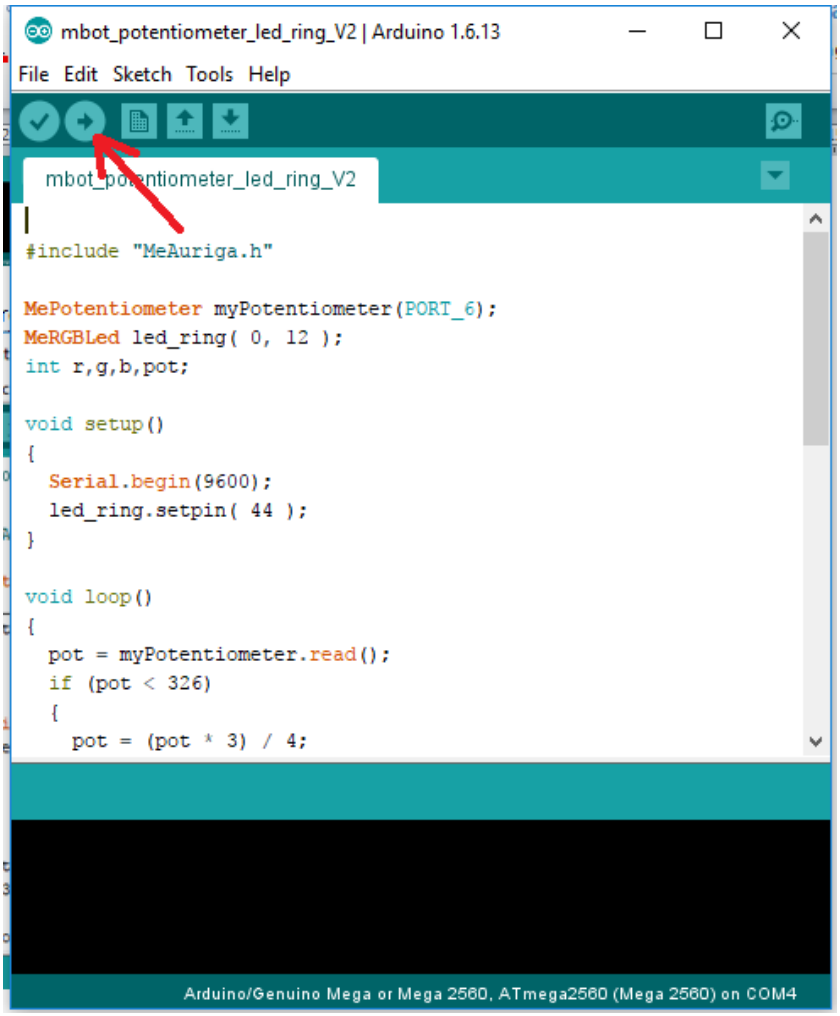
Para trabajar con el mBot Ranger en el software Arduino IDE, tenemos que cambiar la *board* a Mega 2560.



Cuando conecte el Ranger al PC, en el menú Herramientas, encontrará el puerto COM.



Cuando la programación esté lista, hacemos click en *Upload:*



<u>Pasos para desarrollar el programa para controlar el anillo led.</u>

Primero se debe aprende sobre el método para controlar el anillo de led.

Aquí se muestra un fragment de la documentación:

bool	**setColorAt** (uint8_t index, uint8_t red, uint8_t green, uint8_t blue)
	set the rgb value of the led with the index.

void	**show** ()
	become effective of all led's change.

El anillo tiene 12 leds. Tienen números del 1 al 12.

Para establecer el color de todos los leds usamos la instrucción: setColor (0, r, g, b) - 0 significa todos los leds. Luego coloca tres números del 0 al 255.

Después de cada instrucción, *setColor* usa show() y delay(), para ver la acción.

Por ejemplo:

```
#include "MeAuriga.h"
MeRGBLed led_ring( 0, 12 );
void setup()
{
  Serial.begin(9600);
  led_ring.setpin( 44 );
}
void loop()
{
  led_ring.setColor(0, 0, 0, 0);
  led_ring.show();
  delay (500);
  led_ring.setColor(2, 100, 0, 0);
  led_ring.show();
  delay(200);
  led_ring.setColor(5, 0, 0, 100);
  led_ring.show();
  delay(200);
  led_ring.setColor(12, 0, 100, 0);
  led_ring.show();
  delay(200);
  }
```

Este programa funciona de la siguiente manera:

1. Apaga todos los leds.
2. Espera 500ms.
3. Enciende el Led número 2 con el color Rojo.
4. Espera 200ms.
5. Enciende el Led número 5 con el color Azul.
6. Espera 200ms.
7. Enciende el Led número 12 con el color Verde.
8. Espera 200ms.
9. Vuelve al paso 1.

Al final del bucle enciende tres leds.

Utilice el *bucle for* para encender todos los leds uno por uno:

```
#include "MeAuriga.h"

MeRGBLed led_ring( 1, 12 );

void setup()
{
  Serial.begin(9600);
  led_ring.setpin( 44 );
}

void loop()
{
 for (int i=1;i<=12;i++)
    {
      led_ring.setColor(0, 0, 0, 0);
      led_ring.show();
      delay (100);
      led_ring.setColor(i, 100, 0, 0);
      led_ring.show();
      delay(200);
    }
}
```

En un momento, solo un led se enciende porque cada iteración comienza con la instrucción *setColor* `(0, 0, 0, 0)`.

Cuando en el *bucle for*, i=12 todos lo sleds estarán encendidos:

```
void loop()
{
  led_ring.setColor(0, 0, 0, 0);
```

```
  led_ring.show();
  delay (100);
  for (int i=1;i<=12;i++)
    {
      led_ring.setColor(i, 100, 0, 0);
      led_ring.show();
      delay(200);
    }
}
```

Y la última version para el *bucle for*:

```
void loop()
{
   for (int i=1;i<=12;i++)
    {
      led_ring.setColor(i, 100, 0, 0);
      led_ring.show();
      delay(200);
    }
   for (int i=1;i<=12;i++)
    {
      led_ring.setColor(i, 0, 0, 0);
      led_ring.show();
      delay(200);
    }
}
```

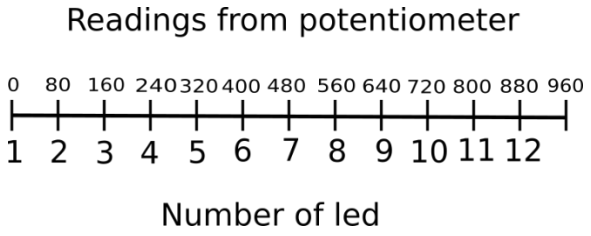
Esta vez, el segundo *bucle for* apaga el led uno por uno.

La función *map*

El potenciómetro da el número de 0 a 970, pero queremos controlar 12 leds:

$$970 \div 12 \approx 80$$

Mira la imagen:



Para transformar las lecturas del potenciómetro que es analógico a número entero, podemos usar la función *map*:

Sintaxis de la función:

map(value, fromLow, fromHigh, toLow, toHigh)

Parámetros

value: el número del *map*.

fromLow: el límite inferior del rango actual del valor.

fromHigh: el límite superior del rango actual del valor.

toLow: el límite inferior del rango objetivo del valor.

toHigh: el límite superior del rango objetivo del valor.

En el programa solo escribimos:

```
mapped = map(myPotentiometer.read(), 0, 970, 1, 12);
```

El valor mapeado es el número de led.

Versión final del programa:

```
#include "MeAuriga.h"
MePotentiometer myPotentiometer(PORT_6);
MeRGBLed led_ring( 0, 12 );
int mapped;

void setup()
{
  Serial.begin(9600);
  led_ring.setpin( 44 );
}

void loop()
{
  led_ring.setColor(0, 0, 0, 0);
  led_ring.show();
  mapped = map(myPotentiometer.read(), 0, 970, 1, 12);
  led_ring.setColor(mapped, 100, 0, 0);
  led_ring.show();
  delay(100);
}
```

Programa para controlar el color

La segunda versión es que queremos controlar el color de los leds.

Del rango 0 al 970 tenemos que combinar tres valores que nos dan los valores r, g, b en la instrucción:

```
setColor(0, r, g, b);
```

Dividiremos las lecturas del potenciómetro en cuatro secciones:

El valor *pot* es la lectura del potenciómetro, pero necesitamos reducirlo a valores de 0 a 255.

Lecturas	$pot \in \langle 0, 255)$	$pot \in \langle 255, 510)$	$pot \in \langle 510, 765)$	$pot \in \langle 765, 970)$
Reducción		$pot = pot - 255$	$pot = pot - 510$	$pot = pot - 765$
Pot después de la reducción	$pot \in \langle 0, 255)$	$pot \in \langle 0, 255)$	$pot \in \langle 0, 255)$	$pot \in \langle 0, 205)$
La definición del rojo	$r = 255 - pot$	$r = 1$	$r = pot$	$r = 255 - pot$
La definición del verde	$g = pot$	$g = 255 - pot$	$g = 1$	$g = pot$
La definición del azul	$b = 1$	$b = pot$	$b = 255 - pot$	$b = 1$
El color para pot=0	(255,0,1)	(1,255,0)	(0,1,255)	(255,0,1)
El color para pot=125	(130,125,1)	(1,130,125)	(125,1,130)	(130,125,1)
El color para pot=254	(1,254,1)	(1,1,254)	(254,1,1)	(50,205,1)

Como ves, cuando el valor del potenciómetro cambia el intervalo, el color no cambia:

```
#include "MeAuriga.h"

MePotentiometer myPotentiometer(PORT_6);
MeRGBLed led_ring( 0, 12 );
int r,g,b,pot;

void setup()
{
  Serial.begin(9600);
  led_ring.setpin( 44 );
}

void loop()
{
  pot = myPotentiometer.read();
```

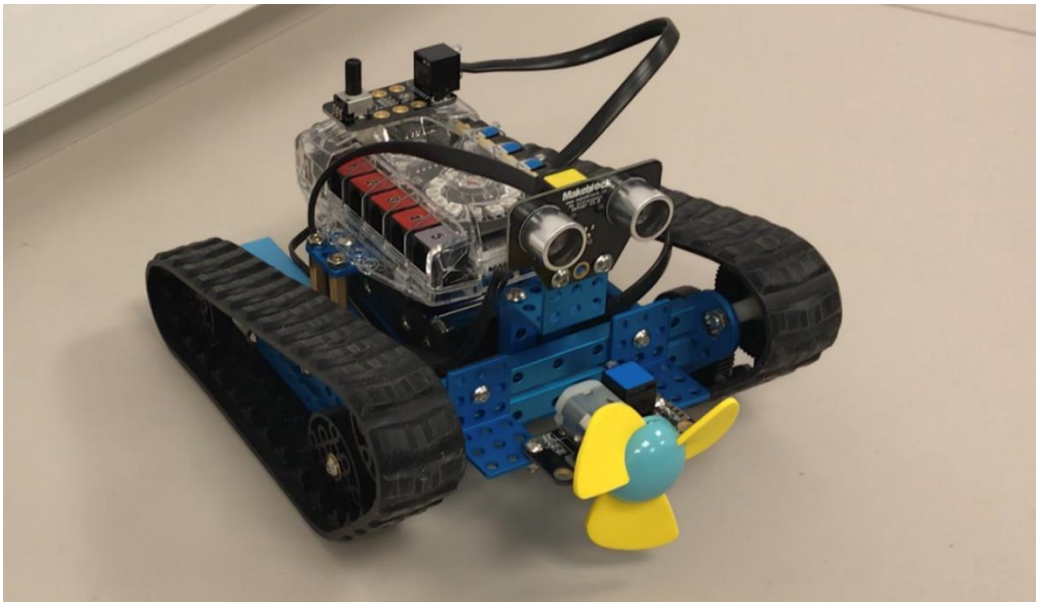
```
if (pot < 255)
  {
    r = 255 - pot;
    g = pot;
    b = 1;
  }
  else if (pot < 510)
  {
    pot = pot-255;
    r = 1;
    g = 255 - pot;
    b = pot;
  }
  else if (pot < 765)
  {
    pot = pot-510;

    r = 1;
    g = 255 - pot;
    b = pot;
  }
  else
  {
    pot = pot-765;

    r = pot;
    g = 1;
    b = 255 - pot;
  }
  led_ring.setColor(0, r, g, b);
  led_ring.show();
  delay(100);
}
```

Segunda versión

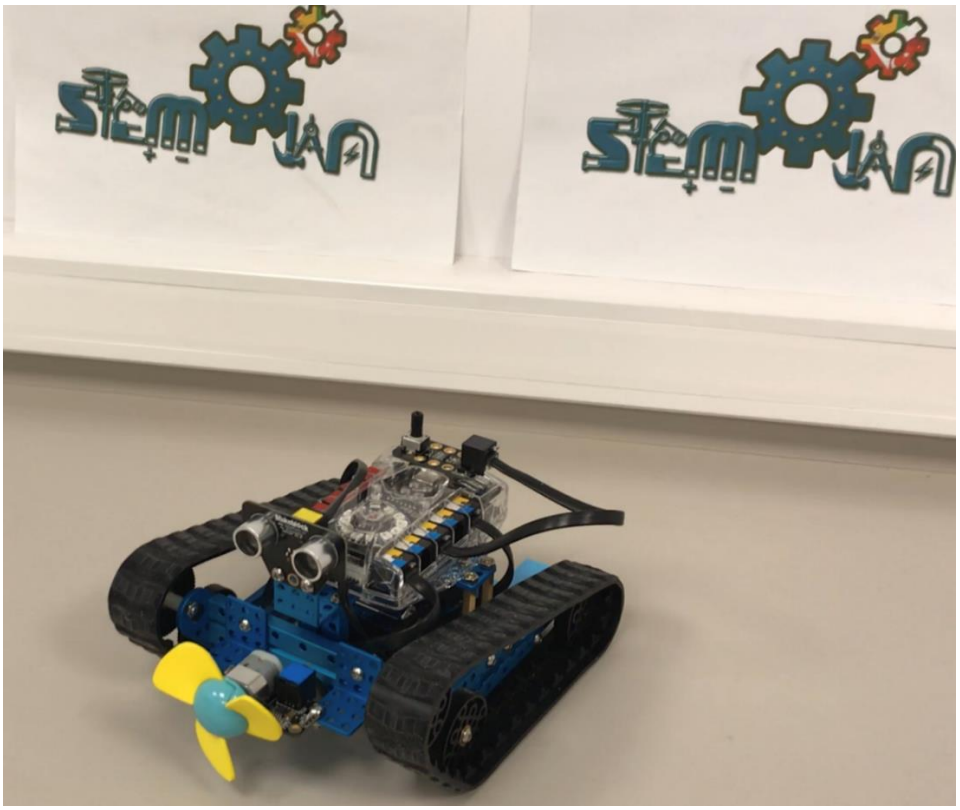
El Ranger, con la ayuda del potenciómetro, actuará como un ventilador y podrá encenderlo y apagarlo, así como regular la velocidad.

El ventilador Makeblock, como la mayoría de los motores binarios de Arduino, solo tiene dos estados, se enciende y se apaga, y no es posible regular la velocidad. Sin embargo, puede regular a intervalos regulares de tiempo, determinado con el potenciómetro, detener por un breve momento el motor, lograremos regular su velocidad con precisión.

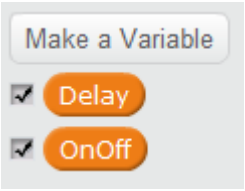
a. Primero, conectamos el sensor de potenciómetro y el *pack* de ventiladores a mBot Ranger.



b. Ahora procederemos a programar el mBot Ranger:



Utilizaremos estas variables para:

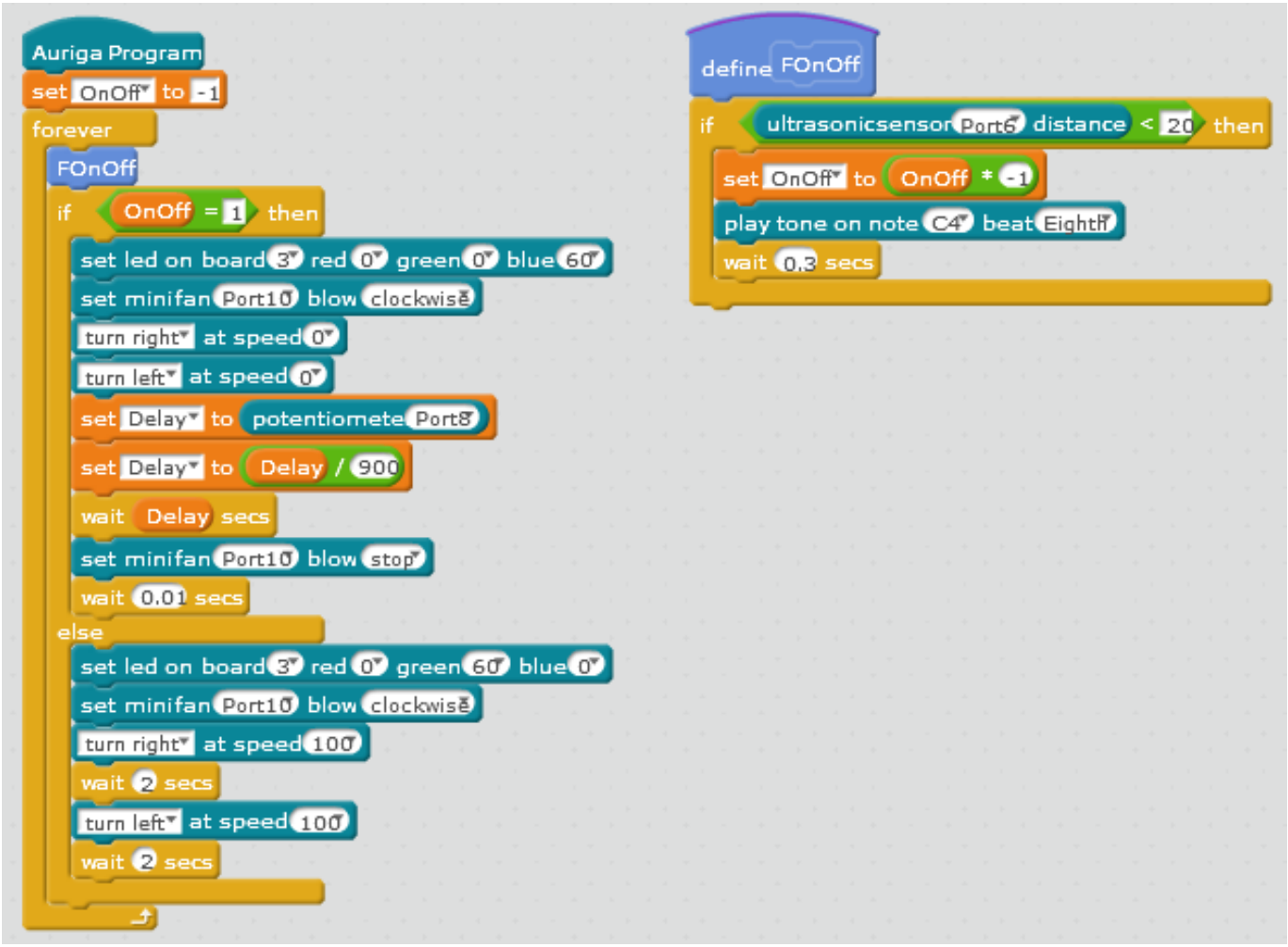


- Delay: El tiempo de espera.
- OnOff: Para indicar si el ventilador está encendido o apagado.

Utilizamos la función *for*:

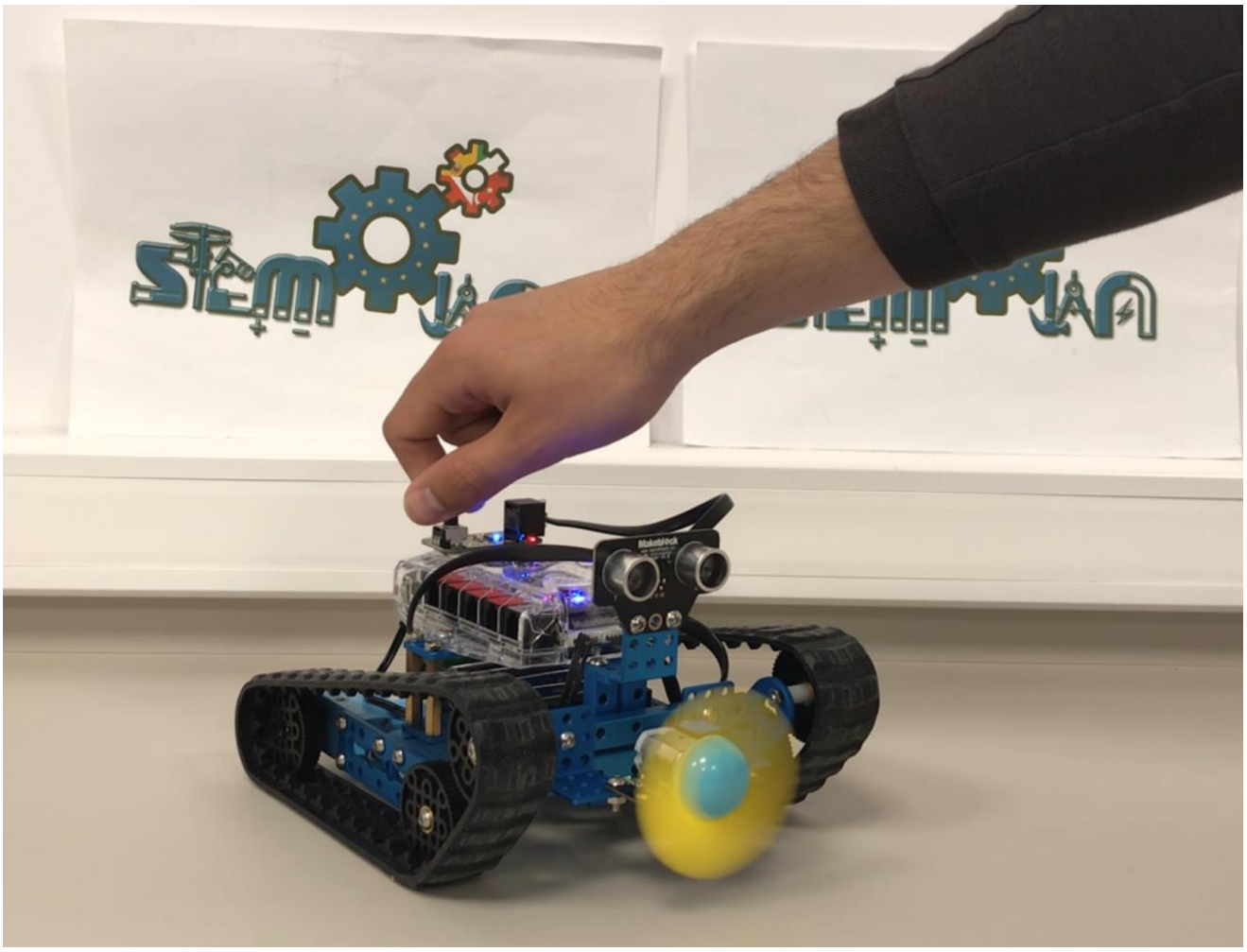
- FOnOff: Función responsable de apagar o encender el ventilador.

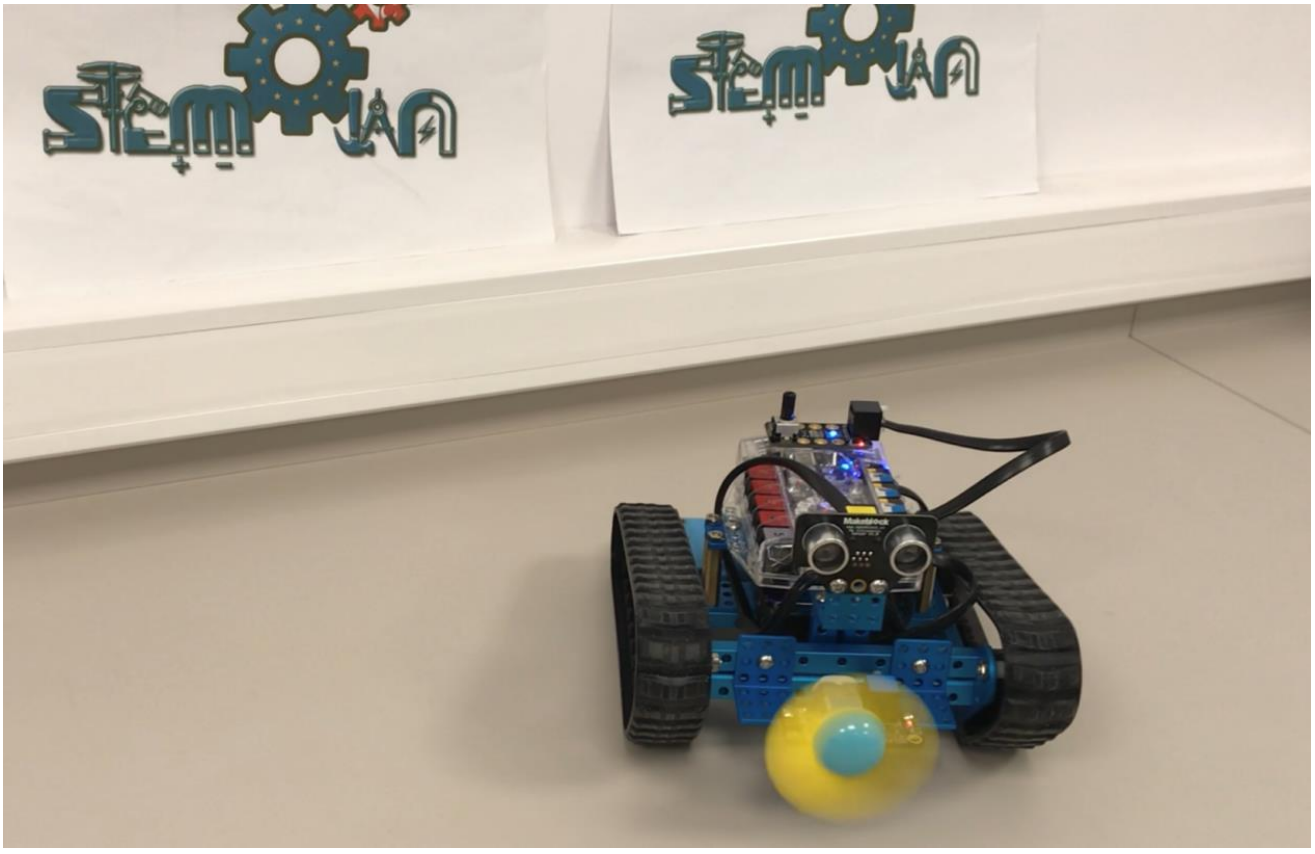
Una vez que sepamos las variables que necesitamos y la función, mostraremos el código del programa completo:



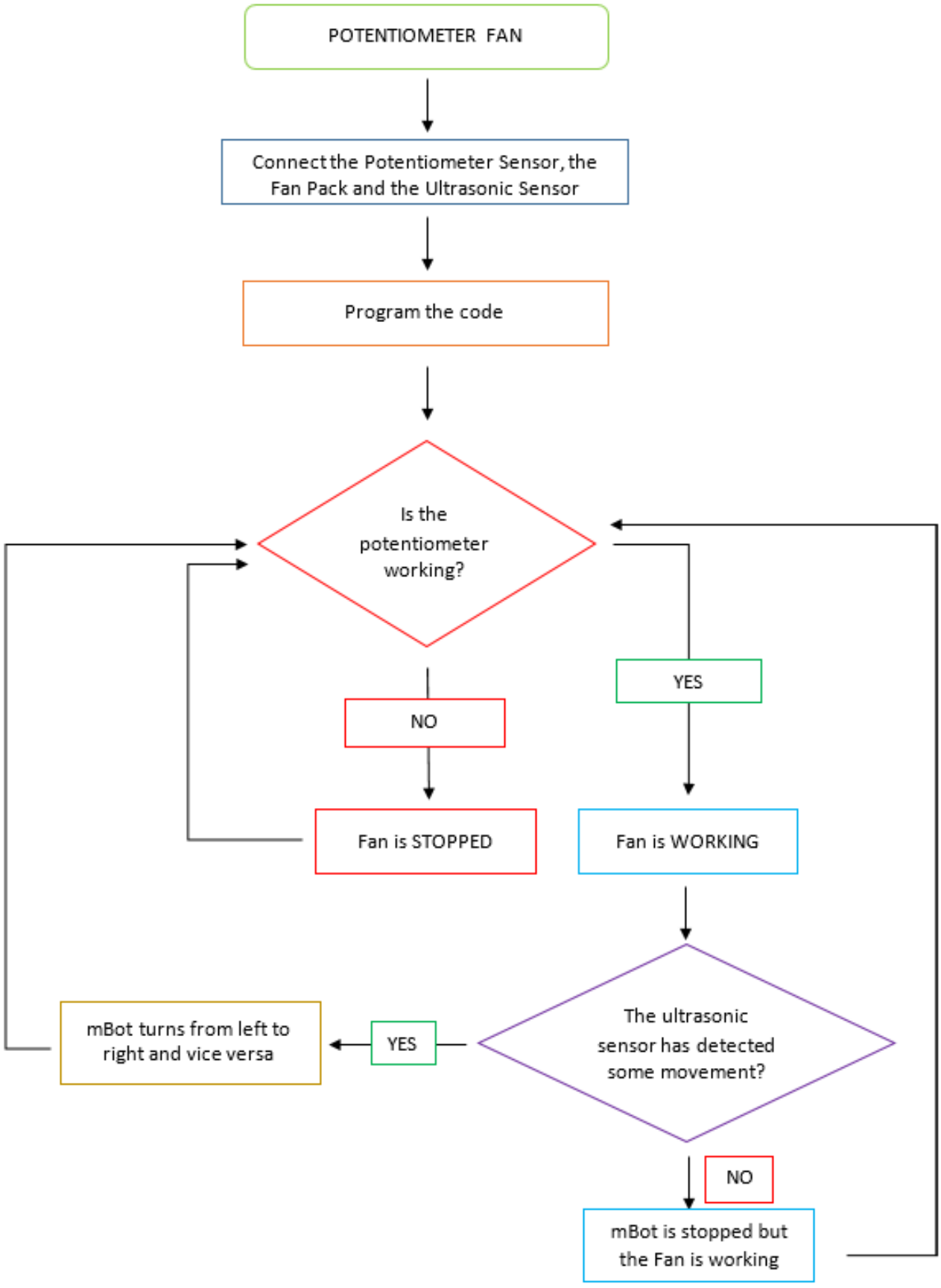
El bucle del programa consultará en la función "FOnOff" el valor proporcionado por el potenciómetro, ya que este valor aumenta o disminuye la velocidad del ventilador.

Para hacer que la actividad sea más atractiva, hemos incorporado un sensor de ultrasonidos, que cuando pones tu mano u otro objeto cerca de ella, se activarán los motores de las ruedas Ranger, que comenzarán a girar como si fuera un ventilador de pie.





Segunda versión



EVALUACIÓN DE LOS ESTUDIANTES

Los alumnus tendrán la capacidad de modificar los colores del LED y su duración. También podrán utilizar los intervalos que ellos deseen.

BIBLIOGRAFÍA

https://github.com/Makeblock-official/Makeblock-Libraries

http://wiki.makeblock.cc/library/docs/class_me_r_g_b_led.html

MÁS INFORMACIÓN

Para mostrar las lecturas del potenciómetro u otro sensor, puede utilizar las instrucciones:

Serial.println(k)

Donde k se encarga de leer los valores que proporciona el potenciómetro.

Coloca la instrucción en la sección *loop()* y haz click donde indica la flecha: