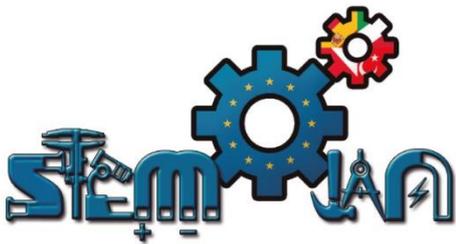


JOYSTICK



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

JOYSTICK

RESUMEN

1. Utilice la palanca de mando para controlar su robot - la velocidad y la dirección.
2. Usa el joystick para jugar el juego en PC.

OBJETIVOS DIDÁCTICOS

En la primera versión:

- ❖ Valores analógicos y digitales.
- ❖ Transformar valor analógico a valor digital.

En la segunda versión:

- ❖ Saber utilizar dos tarjetas Arduino en S4A.
- ❖ Saber controlar personajes mediante sensores.
- ❖ Saber utilizar bloques de código como bucle, detección de diferentes maneras.

Materia STEM:

Ciencia

Tecnología

Ingeniería

Matemáticas

Nivel educativo:

12-14 años

14-16 años

PLANTEAMIENTO DEL PROBLEMA

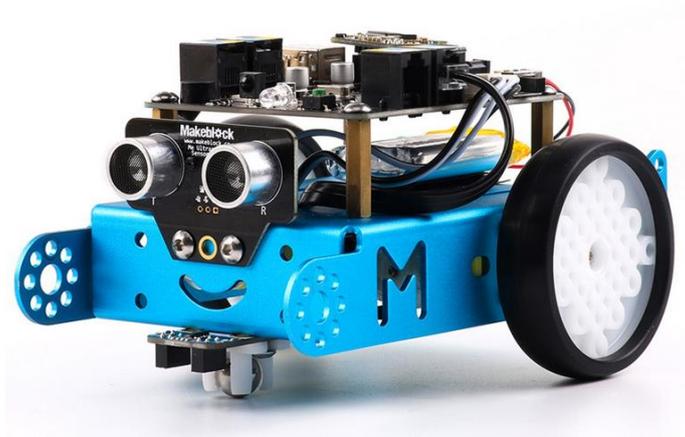
En la primera parte, usamos el mBot y Joystick para controlar el movimiento del robot. El joystick y el robot están conectados por un cable. La velocidad del robot está determinada por la desviación del joystick.

En la segunda parte, se ha realizado una actividad preparada para el uso del joystick.



LISTADO DE MATERIALES

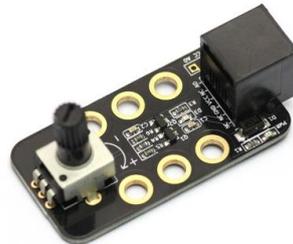
➤ mBot => Ref. 90054



❖ (x2) Me Joystick:



❖ (x2) Me Potenciómetro:



Arduino:

- ❖ (x2) placas Arduino.
- ❖ (x2) Board.
- ❖ (x2) Sensores de Luz (LDR).
- ❖ Resistencias.
- ❖ (x2) Buzzer.
- ❖ (x2) Cables USB.

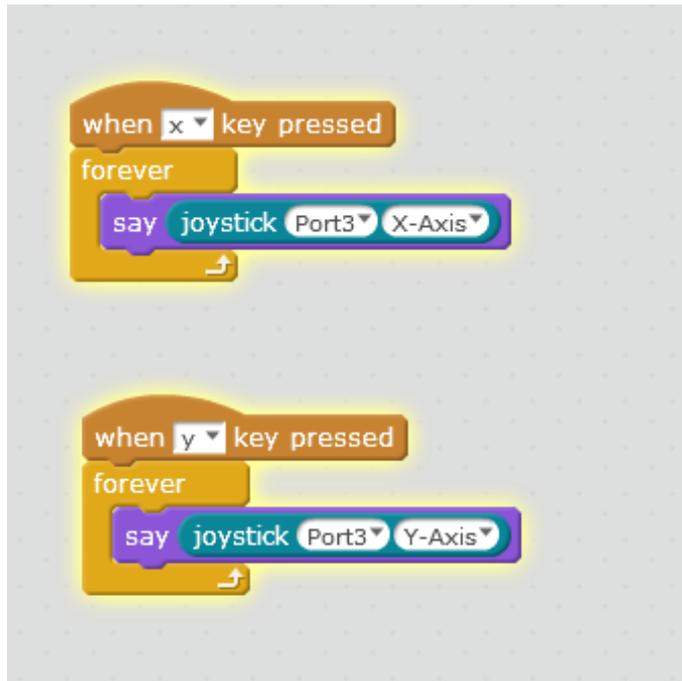
DESCRIPCIÓN DE LA ACTIVIDAD

Primera versión

Paso 1: Las lecturas del joystick:

El Módulo Joystick se utiliza para controlar la dirección de movimiento del robot y del videojuego interactivo. Tiene un puerto analógico y debe estar conectado al puerto ID negro en el mBot.

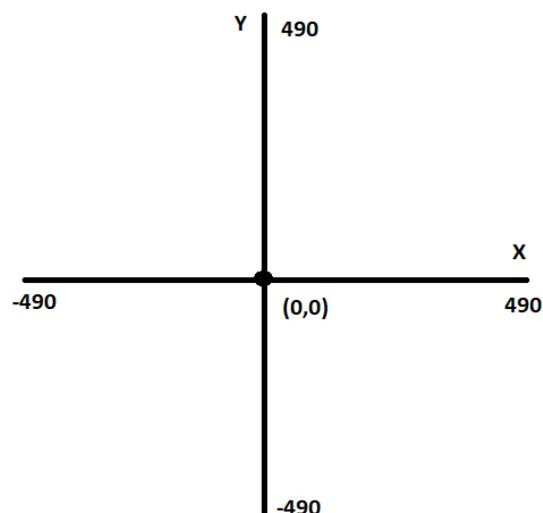
Conecte el joystick al puerto 3 del mBot y verifique los valores que le proporcionan el sensor.



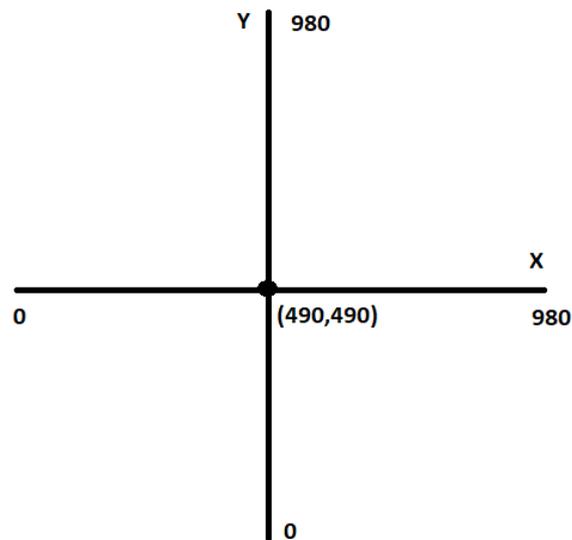
1. X to left position

En la posición neutral, el eje X y el eje Y dan el número 0, pero oscila entre -2 y 2. Cuando se mueve, el *stick* a otro número de posición éste cambia. El mínimo es -490 y el máximo es 490.

Tenga cuidado con el valor 490. Cuando coloca el joystick en la posición máxima, el valor es 486-490, porque es un valor analógico.



Consejo: Cuando utilizamos el joystick por primera vez, nuestras lecturas fueron diferentes (de 0 a 980). Pero después de cargar el programa a bordo, cambia a valores de -490 a 490.



Paso 2: Control del Robot:

En esta etapa escribimos el programa, que controla el robot, pero la velocidad no cambia.

En el bucle "forever", el robot lee la posición del *stick* y gira a la izquierda o derecha cuando nos movemos en la dirección "x".

El robot avanza o retrocede cuando nos movemos en dirección "y".

Pero cuando el *stick* está en posición (0,0) el robot continúa de carrera.

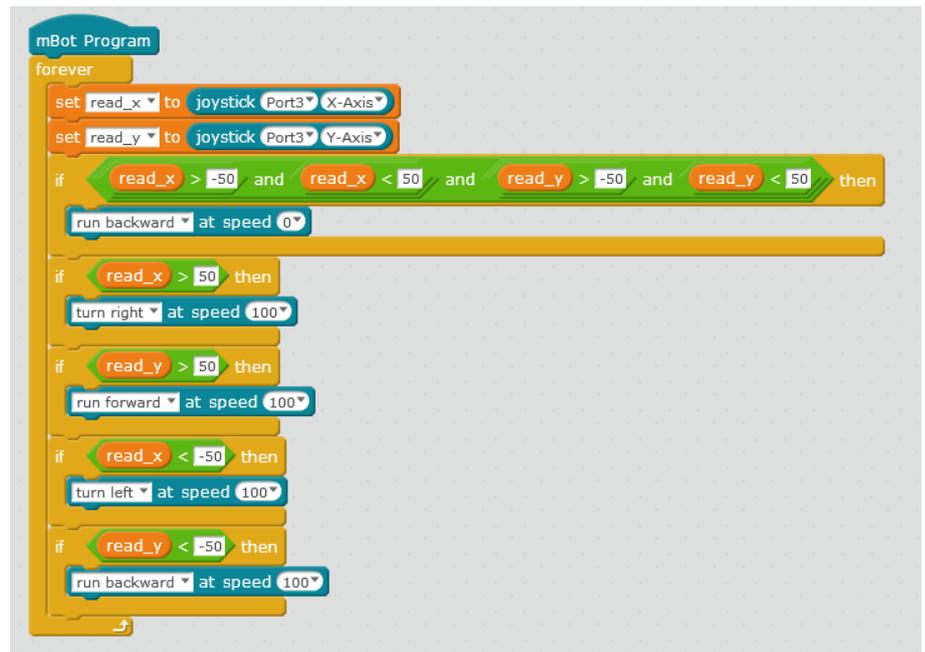
```
mBot Program
forever
  set read_x to joystick Port3 X-Axis
  set read_y to joystick Port3 Y-Axis
  if read_x > 50 then
    turn right at speed 100
  if read_y > 50 then
    run forward at speed 100
  if read_x < -50 then
    turn left at speed 100
  if read_y < -50 then
    run backward at speed 100
```

Segunda versión

Al programa anterior tenemos que agregar una *sentencia if* que detenga al robot cuando la palanca está en posición neutral.

Se realiza por condición:

$$\begin{cases} x \in (-50,50) \\ y \in (-50,50) \end{cases}$$

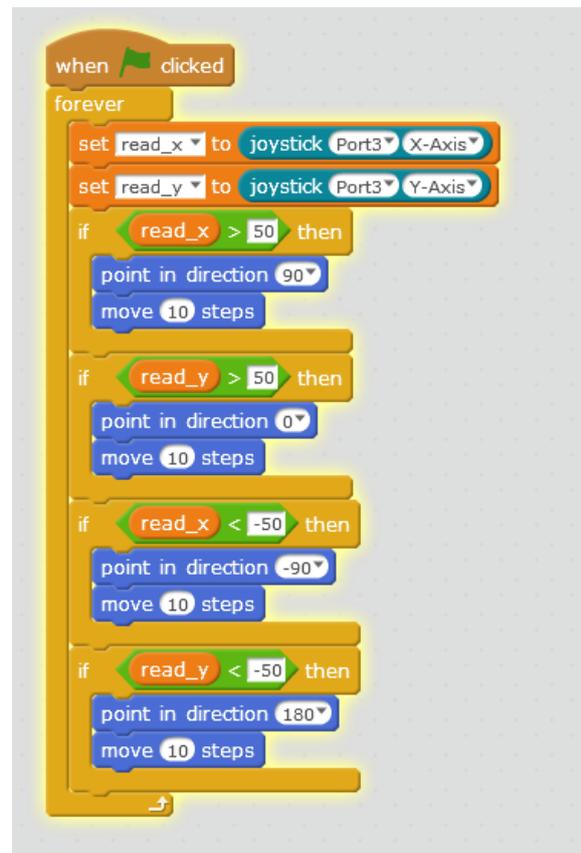


```
mBot Program
forever
  set read_x to joystick Port3 X-Axis
  set read_y to joystick Port3 Y-Axis
  if read_x > -50 and read_x < 50 and read_y > -50 and read_y < 50 then
    run backward at speed 0
  if read_x > 50 then
    turn right at speed 100
  if read_y > 50 then
    run forward at speed 100
  if read_x < -50 then
    turn left at speed 100
  if read_y < -50 then
    run backward at speed 100
```

Usa el joystick para controlar el juego de computadora:

Existe un algoritmo muy similar para controlar el *sprite* en el programa mBlock.

En esta versión el *sprite* puede ir en diagonal.



```
when clicked
forever
  set read_x to joystick Port3 X-Axis
  set read_y to joystick Port3 Y-Axis
  if read_x > 50 then
    point in direction 90
    move 10 steps
  if read_y > 50 then
    point in direction 0
    move 10 steps
  if read_x < -50 then
    point in direction -90
    move 10 steps
  if read_y < -50 then
    point in direction 180
    move 10 steps
```

Cuando quieras mover el *sprite* en modo analógico hacia el eje, debes agregar más condiciones

Derecha:

$$\begin{cases} x > 50 \\ y \in (-50, 50) \end{cases}$$

Izquierda:

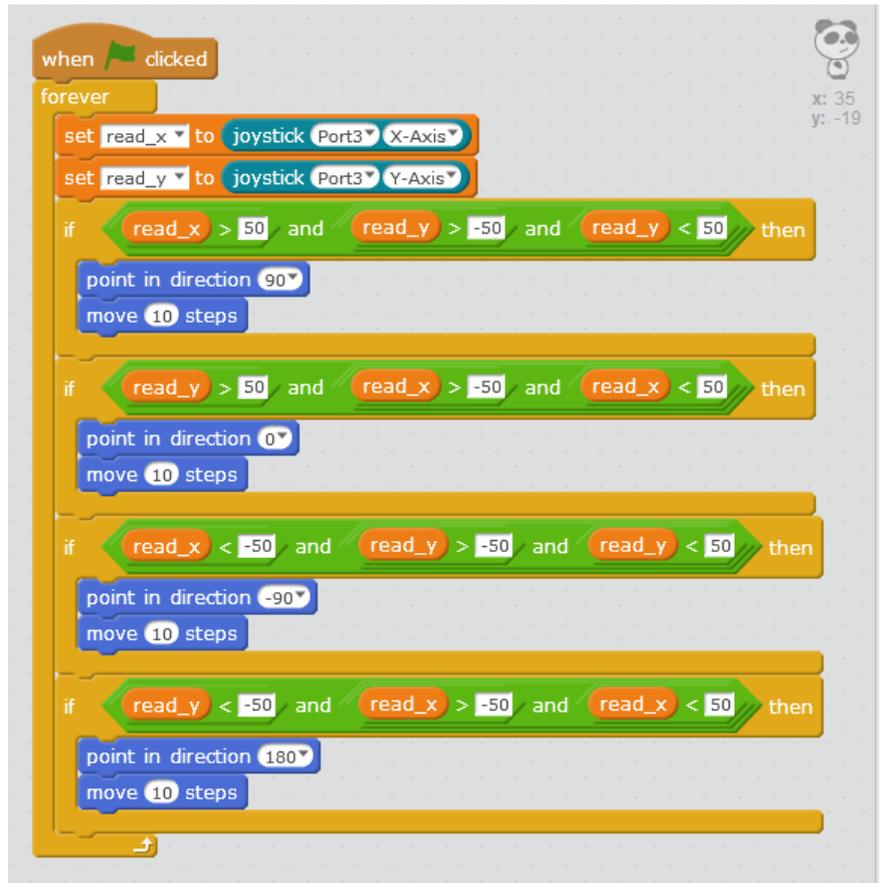
$$\begin{cases} x < -50 \\ y \in (-50, 50) \end{cases}$$

Arriba:

$$\begin{cases} x \in (-50, 50) \\ y > 50 \end{cases}$$

Abajo:

$$\begin{cases} x \in (-50, 50) \\ y < -50 \end{cases}$$



```

when clicked
  forever
    set read_x to joystick Port3 X-Axis
    set read_y to joystick Port3 Y-Axis
    if read_x > 50 and read_y > -50 and read_y < 50 then
      point in direction 90
      move 10 steps
    if read_y > 50 and read_x > -50 and read_x < 50 then
      point in direction 0
      move 10 steps
    if read_x < -50 and read_y > -50 and read_y < 50 then
      point in direction -90
      move 10 steps
    if read_y < -50 and read_x > -50 and read_x < 50 then
      point in direction 180
      move 10 steps
  
```

Control la velocidad robot:

Para controlar la velocidad del robot, tenemos que transformar los valores analógicos del intervalo (-490, 490) a digital de intervalo (-250, 250).

Debemos utilizar la función proporcional lineal $y = ax + b$

Sabemos que 0 transforma a 0, y 490 transforma a 250.

Resolvamos el sistema de ecuaciones: $\begin{cases} 0 = 0 \cdot a + b \\ 250 = 490 \cdot a + b \end{cases}$

La solución es $\begin{cases} a = \frac{250}{490} \\ b = 0 \end{cases}$

Para transformar el valor analógico de la lectura del joystick en valores digitales para controlar la velocidad, necesitamos usar la ecuación $y = \frac{250}{490}x$



```

mBot Program
forever
  set read_x to joystick Port3 X-Axis
  set read_y to joystick Port3 Y-Axis
  set speedx to round read_x * 250 / 490
  set speedy to round read_y * 250 / 490
  if read_x > -3 and read_x < 3 and read_y > -3 and read_y < 3 then
    run forward at speed 0
  if read_x > 2 then
    turn right at speed speedx
  if read_y > 2 then
    run forward at speed speedy
  if read_x < -2 then
    turn right at speed speedx
  if read_y < -2 then
    run forward at speed speedy

```

Programando en lenguaje Arduino

Es muy común cambiar el valor analógico al valor digital. En el lenguaje Arduino puedes usar la función especial para transformarlo.

Syntax de la función:

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

Parámetros

value: el número a mapear.

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.



El código para controlar el mBot (la velocidad está cambiando)

```
#include <MeMCore.h>
MeJoystick joystick(PORT_6);
MeDCMotor motor1(M1);
MeDCMotor motor2(M2);

int x = 0;
int xmapped = 0;
int ymapped = 0;
int y = 0;

void setup() {
}

void loop() {
  x = joystick.readX();
  y = joystick.readY();
  xmapped = map(x, 2, 490, 0, 255);
  ymapped = map(y, 2, 490, 0, 255);
  if (xmapped > 10 || xmapped < -10){
    motor1.run(xmapped);
    motor2.run(xmapped);
  } else if (ymapped > 10 || ymapped < -10) {
    motor1.run(ymapped);
    motor2.run(-ymapped);
  } else {
    motor1.run(0);
    motor2.run(0);
  }
}
```



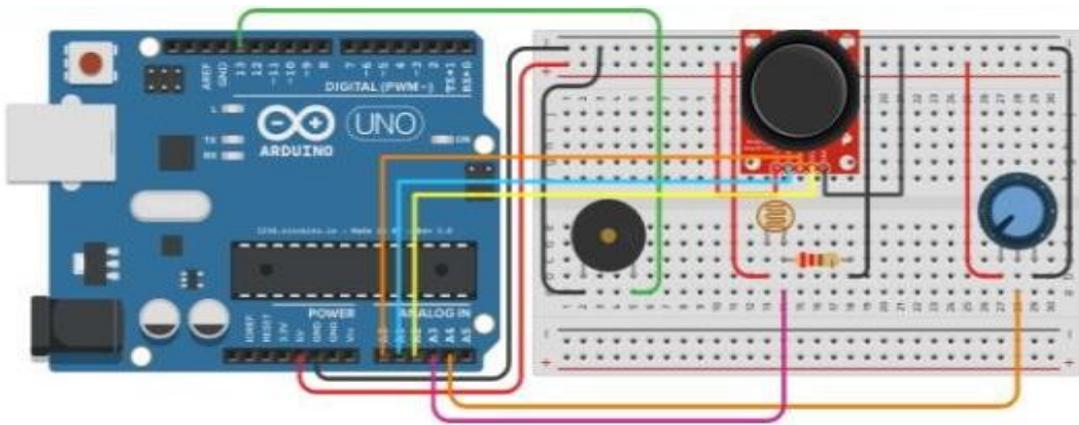
Tercera versión

Ahora te mostramos cómo programar el juego usando S4A y la placa Arduino.

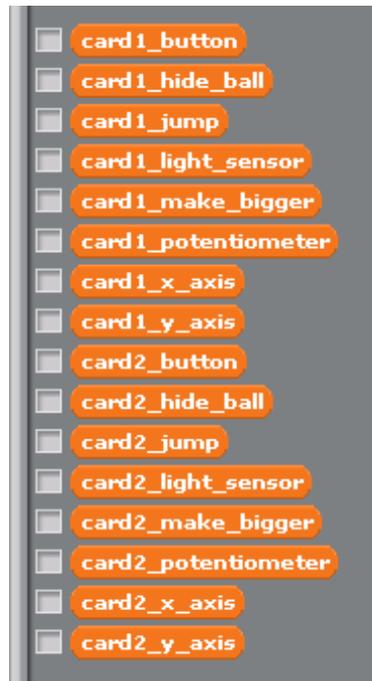
S4A es una modificación Scratch que permite la programación simple de la plataforma de hardware de código abierto Arduino. Proporciona nuevos bloques para la gestión de sensores y actuadores conectados a Arduino. También hay un panel de informes de sensores similar al de PicoBoard. Para más información, puedes ver el siguiente enlace: <http://s4a.cat/>

Debido a que el proyecto tiene bloques de código largos, cada bloque se mostrará con imágenes.

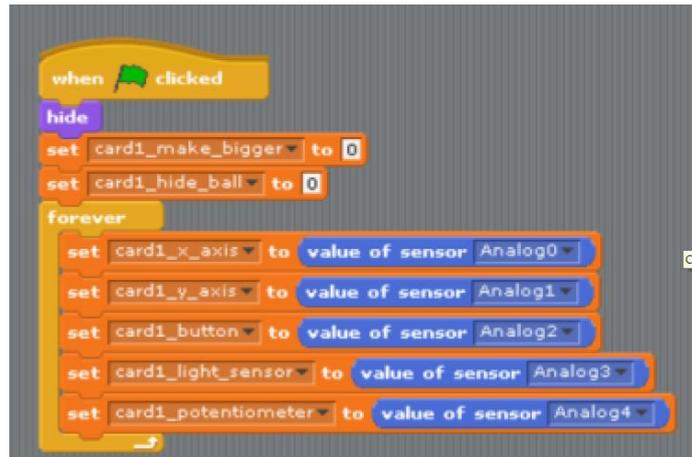
Paso 1: Esquema de conexiones



Paso 2: Hacemos variables para la primera carta de Arduino para el juego.

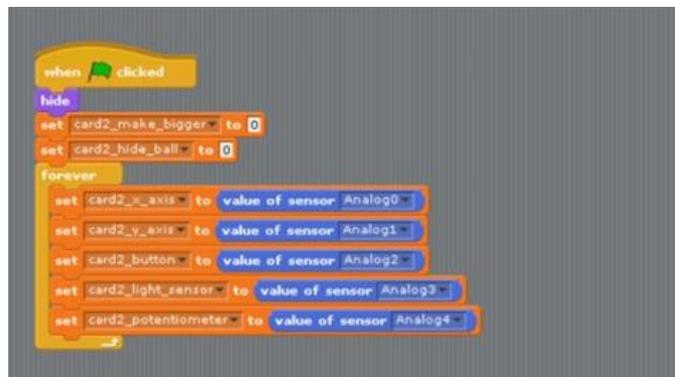


Asignamos pines a los sensores de acuerdo con los pines de la *board* (Arduino 1).



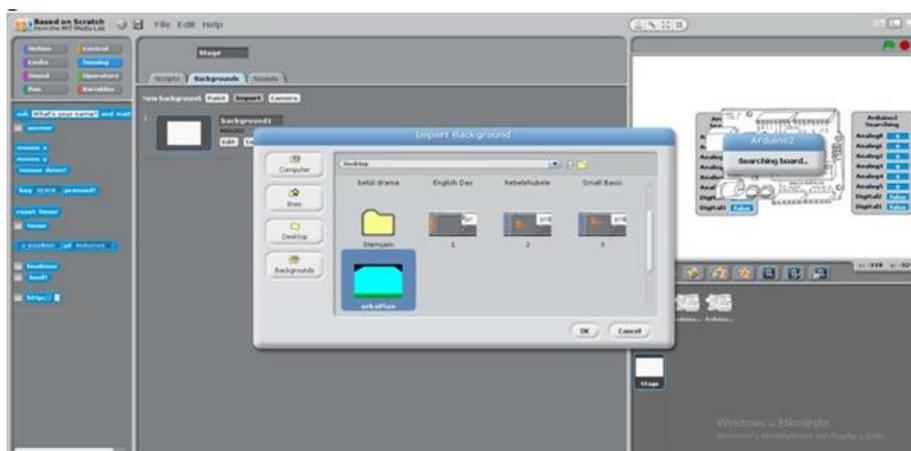
```
when clicked
hide
set card1_make_bigger to 0
set card1_hide_ball to 0
forever
set card1_x_axis to value of sensor Analog0
set card1_y_axis to value of sensor Analog1
set card1_button to value of sensor Analog2
set card1_light_sensor to value of sensor Analog3
set card1_potentiometer to value of sensor Analog4
```

Asignamos pines a los sensores de acuerdo con los pines de la *board* (Arduino 2).

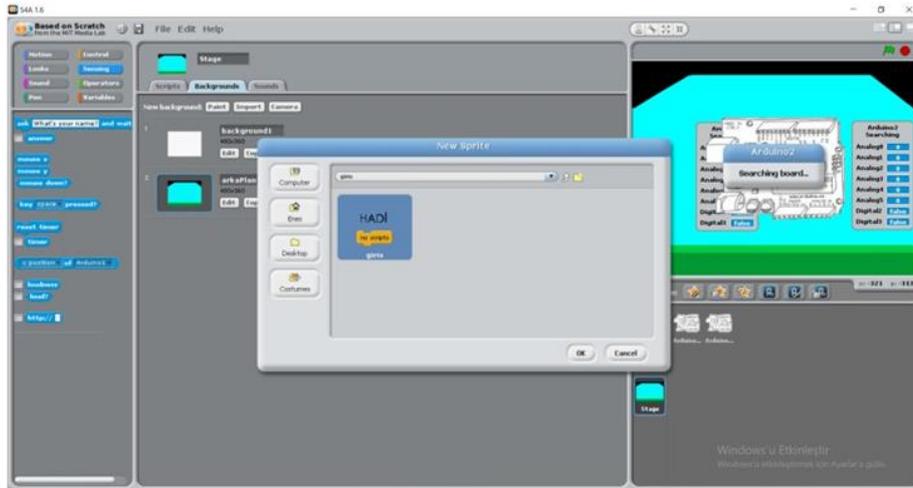


```
when clicked
hide
set card2_make_bigger to 0
set card2_hide_ball to 0
forever
set card2_x_axis to value of sensor Analog0
set card2_y_axis to value of sensor Analog1
set card2_button to value of sensor Analog2
set card2_light_sensor to value of sensor Analog3
set card2_potentiometer to value of sensor Analog4
```

Vamos a añadir nuestro fondo para S4A.



Contamos para comenzar nuestro juego y obtenemos las fotos (con el cambio de vestuario).

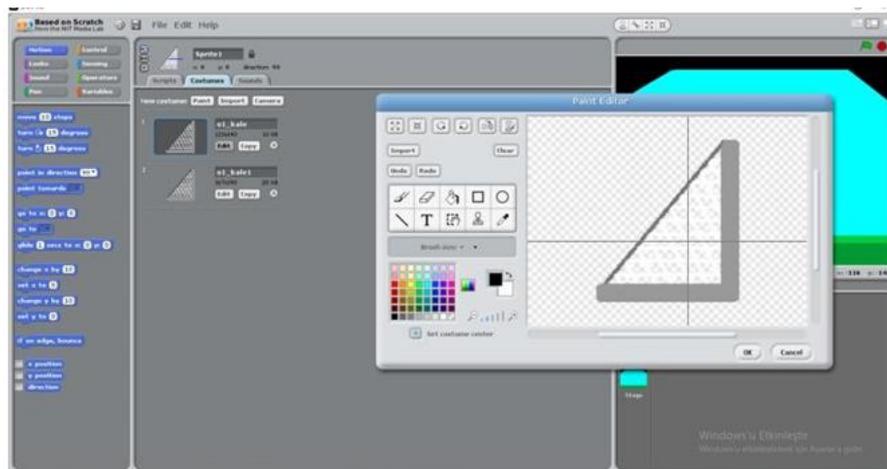


Código para el *sprite*:

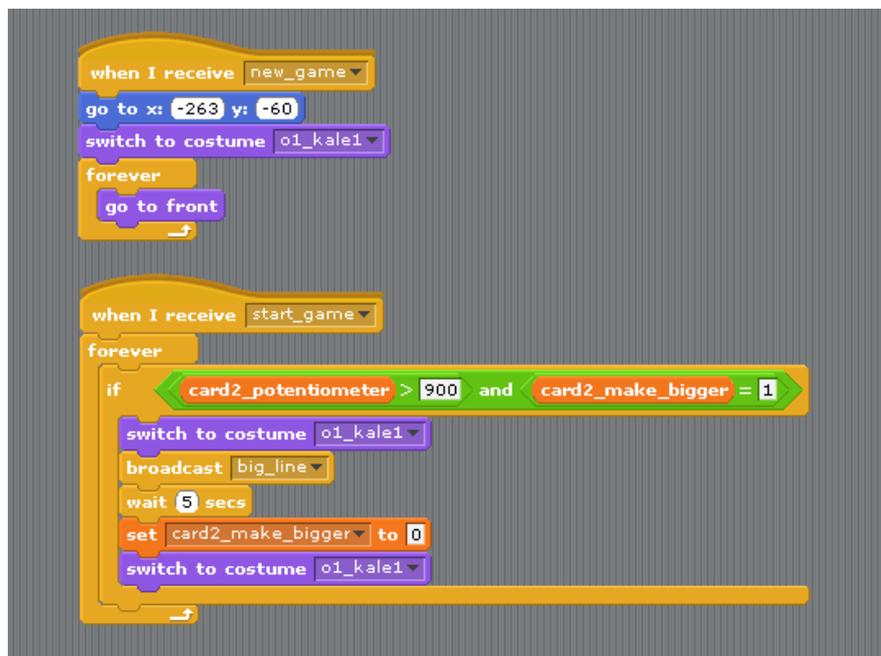
```
when clicked
hide
broadcast new_game

when I receive new_game
glide 1 secs to x: 0 y: 0
show
switch to costume Üc
wait 1 secs
switch to costume iki
wait 1 secs
switch to costume bir
wait 1 secs
switch to costume Hadi
wait 1 secs
hide
broadcast start_game
```

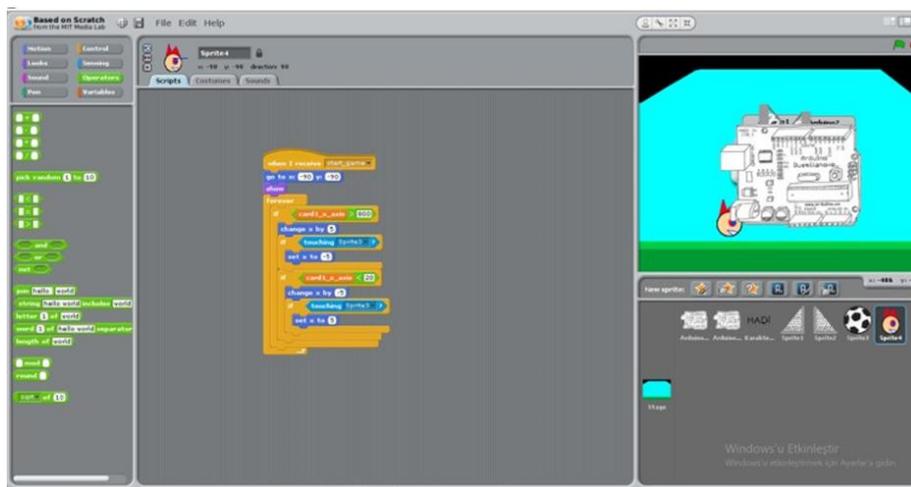
Ahora creamos la portería:



Determinamos las coordenadas de la portería.

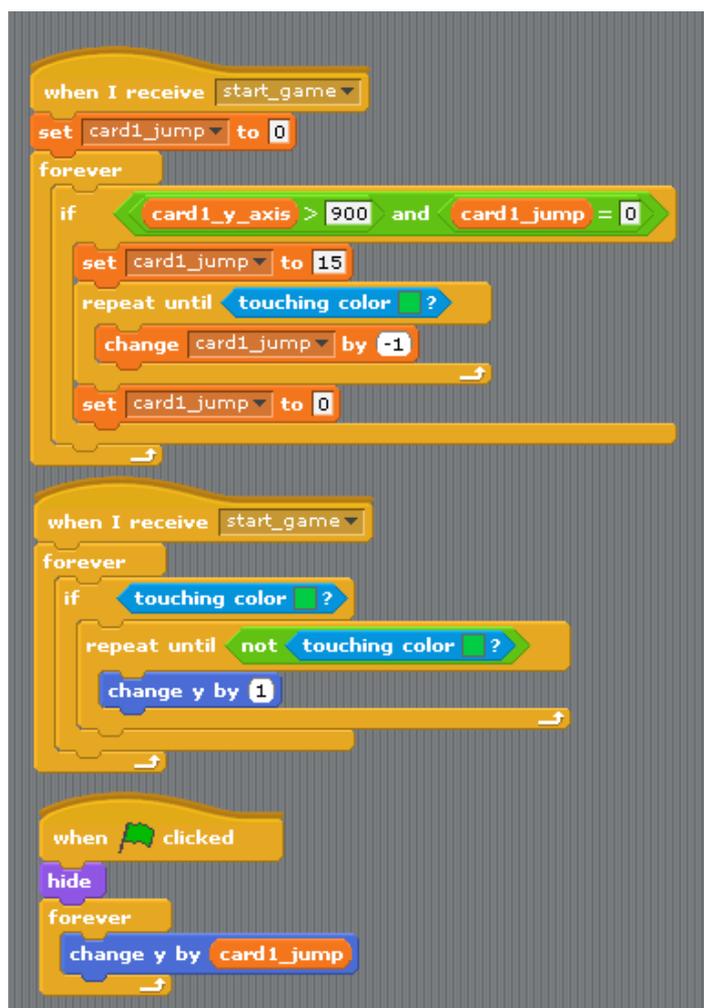


Escribimos los bloques de código para que el jugador del juego se pueda mover.

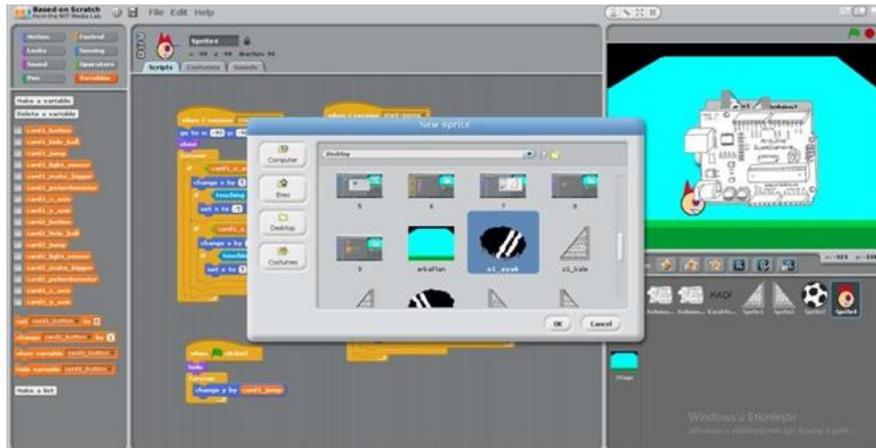


Continuamos formando los bloques de código para que se mueva el primer personaje.

Los bloques de código de salto y movimiento del primer personaje los puedes ver en la imagen. Operaciones similares y bloques de código se repetirán para el segundo personaje.



Añadimos los pies al S4A para seguir a los personajes:



Bloques de código, "foot" del personaje:

```

when I receive start_game
  show
  point in direction 90
  forever
    if card1_button = 0
      turn 15 degrees
    else
      repeat until direction = 90
        turn 15 degrees
    if direction < 0
      point in direction 0
  when clicked
    hide
    set size to 80 %
  when I receive start_game
    forever
      go to Sprite4
  
```

Código de los saltos y movimientos del segundo personaje.

```

when I receive start_game
  go to x: 92 y: -90
  show
  forever
    if 20 > card2_x_axis
      change x by -5
      if touching Sprite3?
        set x to 5
    if 800 < card2_x_axis
      change x by 5
      if touching Sprite3?
        set x to 5
  when clicked
    hide
    forever
      change y by card2_jump
  when I receive start_game
    set card2_jump to 0
    forever
      if card2_y_axis > 900 and card1_jump = 0
        set card2_jump to 15
        repeat until touching color ?
          change card2_jump by -1
        set card2_jump to 0
  when I receive start_game
    forever
      if touching color ?
        repeat until not touching color ?
          change y by 1
  
```

```

when I receive start_game
  show
  point in direction 90
  forever
    if card2_button = 0
      turn 15 degrees
    else
      repeat until direction = 90
        turn 15 degrees
    if direction < 0
      point in direction 180

when I receive start_game
  forever
    go to Sprite6

when clicked
  hide
  set size to 80 %
  
```

Escribimos los códigos de las pelotas después de agregar la bola de la biblioteca. El personaje superior se verá afectado por los personajes de ambos jugadores, por los bordes y por las líneas de gol.

```

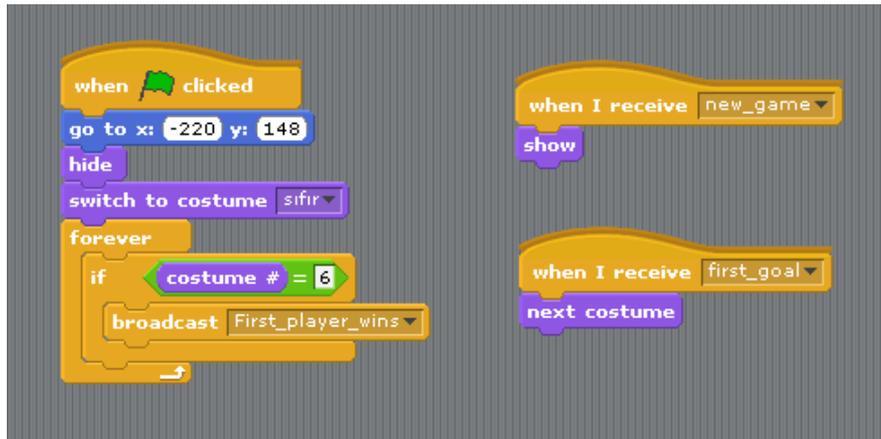
when I receive start_game
  go to x: 0 y: 0
  show
  point in direction 45
  forever
    move 10 steps
    if on edge, bounce
    if touching Sprite4? or touching Sprite5?
    if touching Sprite6? or touching Sprite7? or touching color?
      turn 45 degrees

when I receive start_game
  forever
    if card2_light_sensor < 250 and card2_hide_ball = 1
      set ghost effect to 100
      wait 5 secs
      set ghost effect to 0
      set card2_hide_ball to 0

when I receive start_game
  forever
    if card1_light_sensor < 250 and card1_hide_ball = 1
      set ghost effect to 100
      wait 5 secs
      set ghost effect to 0
      set card1_hide_ball to 0

when clicked
  hide
  set size to 60 %
  
```

Utilizamos el método "Broadcast" para que los personajes puedan seguirse en el juego. Aquí también agregamos las comunicaciones necesarias. Las líneas negras en la pantalla son los personajes que hacen contacto con la pelota.

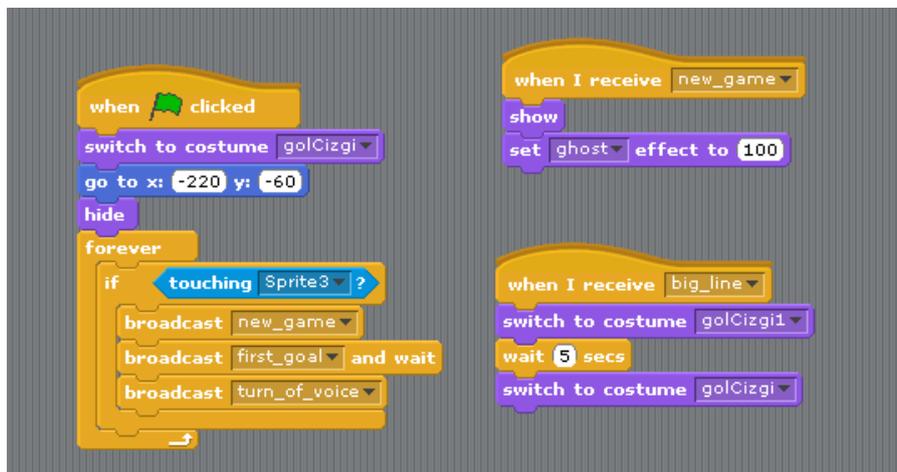


```
when clicked
  go to x: -220 y: 148
  hide
  switch to costume sifir
  forever
    if costume # = 6
      broadcast First_player_wins
  
```

```
when I receive new_game
  show

when I receive first_goal
  next costume
  
```

Este es el código para "line sprite":



```
when clicked
  switch to costume golCizgi
  go to x: -220 y: -60
  hide
  forever
    if touching Sprite3?
      broadcast new_game
      broadcast first_goal and wait
      broadcast turn_of_voice
  
```

```
when I receive new_game
  show
  set ghost effect to 100

when I receive big_line
  switch to costume golCizgi1
  wait 5 secs
  switch to costume golCizgi
  
```

BIBLIOGRAFÍA

<http://learn.makeblock.com/en/me-joystick/>

<https://www.arduino.cc/reference/en/language/functions/math/map/>

www.bilisimgarajakademisi.com

www.eba.gov.tr

www.arduino.cc

ESCALABILIDAD

La primera parte es el conocimiento básico sobre valores analógicos y digitales: se puede utilizar para otros problemas y sensores.

El juego es un proyecto más grande para los estudiantes que tienen más habilidades de programación.

MÁS INFORMACIÓN

Sugerencia sobre el juego: la desventaja de este trabajo es que los cables de puente en el tablero o en el Arduino pueden salir de vez en cuando.