32

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

ABSTRACT

1. Use joystick to control your robot – the speed and direction
2. Use joystick to play the game on PC

DIDACTIC OBJECTIVES

In the first part:

- ❖ Analog and digital values.
- ❖ Transform analog value to digital value.

In the second part:

- ❖ Knowing how to use two Arduino cards in S4A.
- ❖ Knowing how to control characters using sensors.
- ❖ Knowing how to use code blocks like loop, detection in different ways.

STEM Subject: Science☐ Technology ☐ Engineering☒ Mathematics☐

Education Level: 12-14 years☐ 14-16 years☒

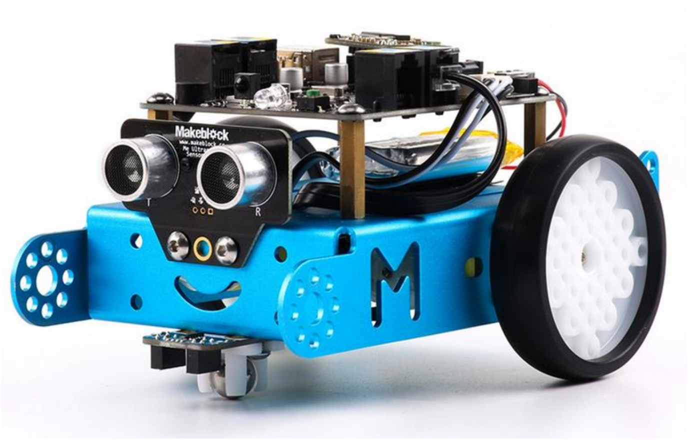
PROBLEM STATEMENT

In the first part use mbot robot and Makeblock – Joystick to control the movement of robot. The joystick and robot are connected by a cable. The speed of the robot is determined by the joystick's deflection

In the second part you controlle a prepared activity by joystick except mouse or keyboard – you will write the game.

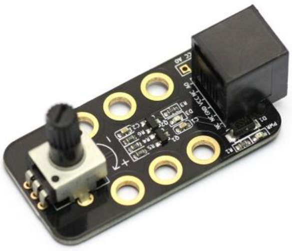
BOM (Bill of Materials Needed)

- mBot => Ref. 90054



- (x2) Me **Joystick**:

- (x2) Me **Potentiometer**:



Arduino:

- (x2) Arduino Card.
- (x2) Breadboard.
- (x2) Light Sensor(LDR).
- Resistance.
- (x2) Buzzer.
- (x2) Usb cables.

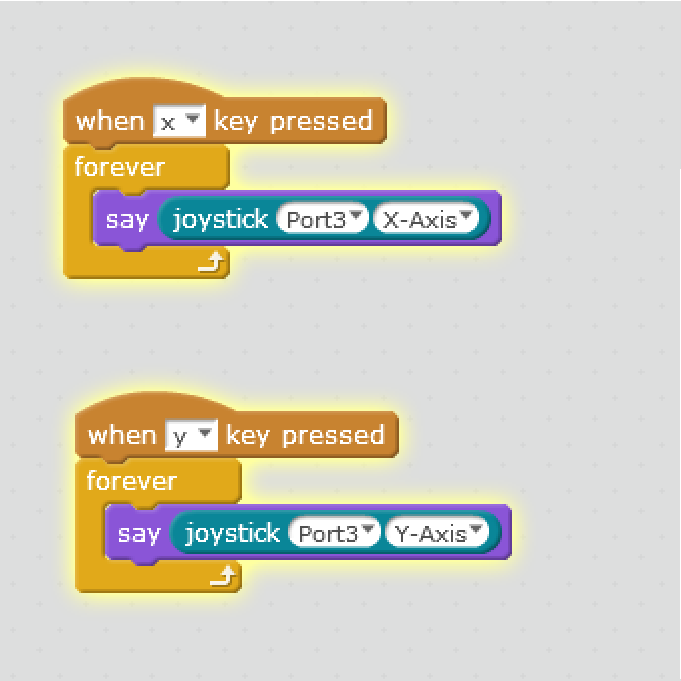
ACTIVITY DESCRIPTION

First version

Step 1: The readings from joystick

Me Joystick Module is used to control the moving direction of cart and the interactive video game. It has an analog port and should be connected to the port with black ID on mBot.

Connect joystick to port 3, connect mBot on serial port and check the values which gives you sensor.

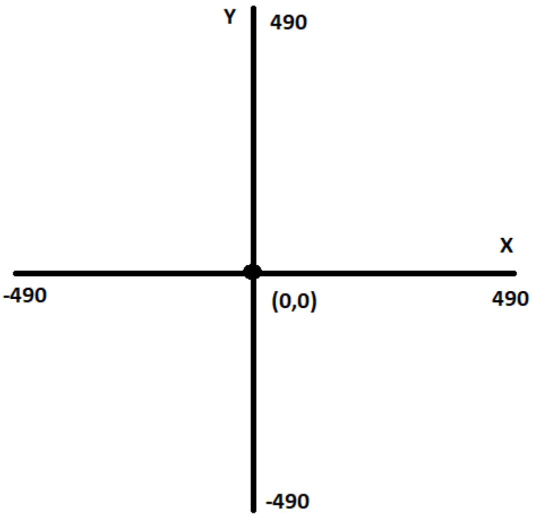


```
when x key pressed
forever
    say joystick Port3 X-Axis

when y key pressed
forever
    say joystick Port3 Y-Axis
```

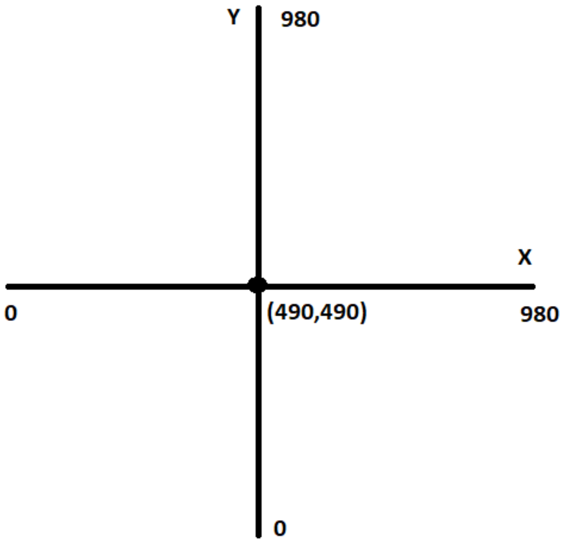
1. X to left position

In neutral position X-Axis and Y-Axis give number 0, but it oscilate between -2 and 2. When you moves stick to other position number is changing. The minimum is -490, the maximum is 490

Be careful about the value 490. When you push joystick to maximum position the value is 486-490, because it is analog value.



Tip: When I used joystick first time my readings were different (From 0 to 980) . But after uploading program to board it changes to values -490 to 490.



<u>Step 2</u>: Robot control

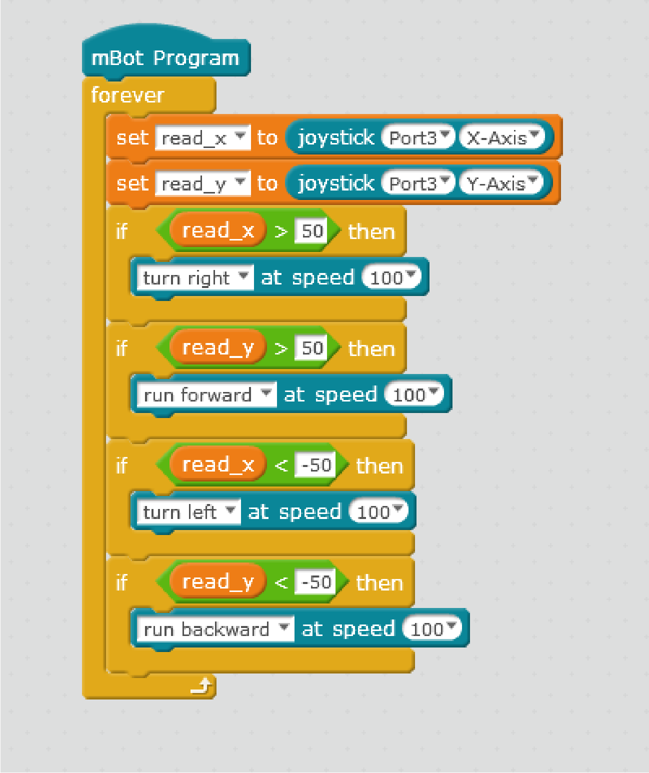
On this stage we write the program, which control the robot, but the speed doesn't change

First version

In forever loop robot read the position of stick and turn left or right when we move on x direction

The robot go forward or backward when we move on y direction.

But when the sctick is in (0,0) position robot continue of running



```
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
```

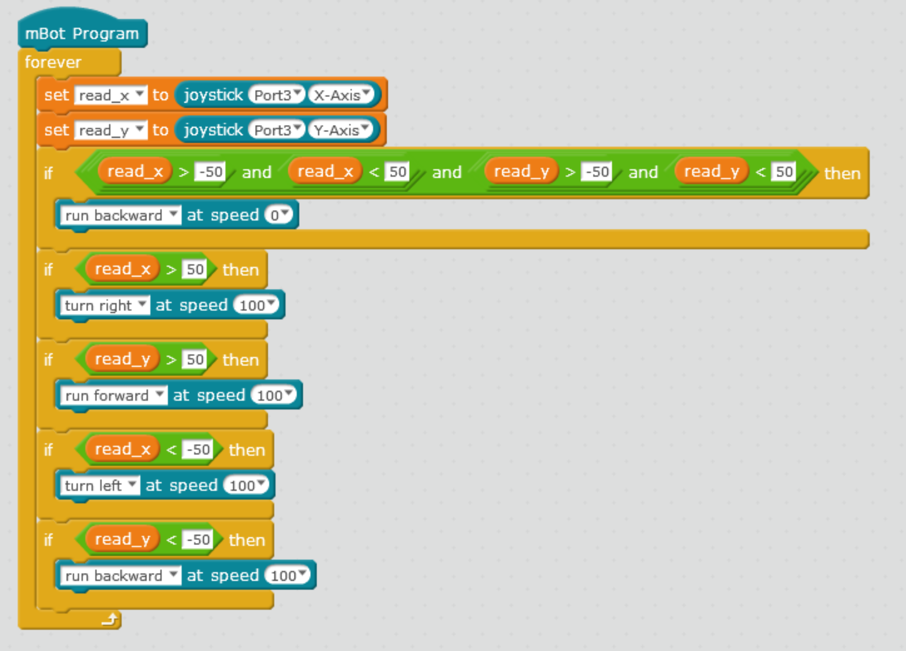
Second version

To previous program we have to add if-statement which stop the robot when the stick is in neutral position.

It is realised by condition:

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

In other project you can change the number 50 to 2.

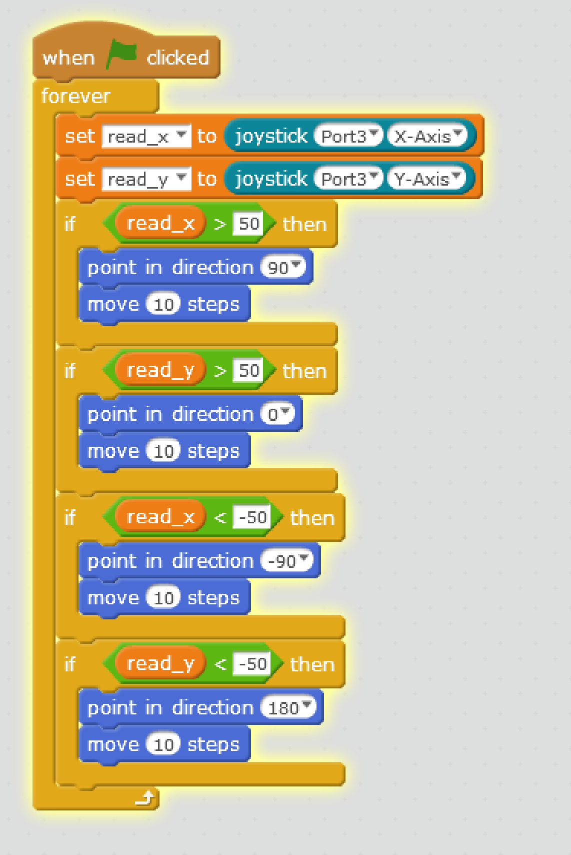


```
mBot Program
forever
    set read_x to joystick Port3 X-Axis
    set read_y to joystick Port3 Y-Axis
    if read_x > -50 and read_x < 50 and read_y > -50 and read_y < 50 then
        run backward at speed 0
    if read_x > 50 then
        turn right at speed 100
    if read_y > 50 then
        run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

Use joystick to control computer game

There is very similar algorithm to control the spirit in mblock program.

In this version the spirit can go diagonally



```
when [flag] 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
```

When you want to go with panda only analog to axis, you should add more condition

Right:

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

Left:

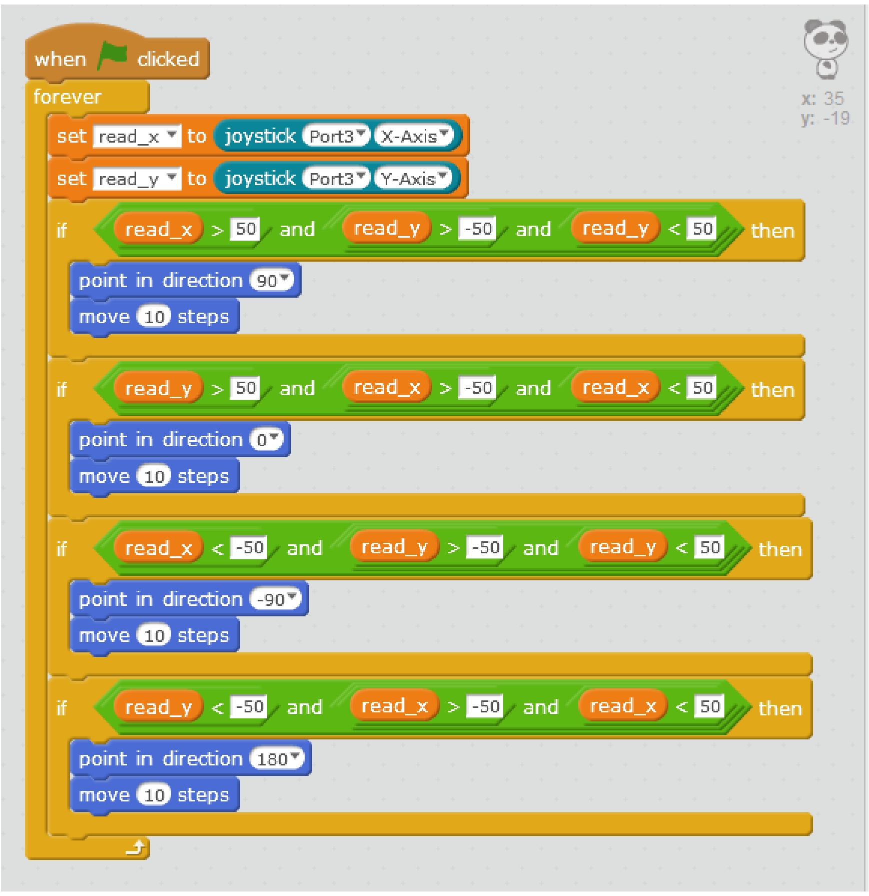
$$\begin{cases} x < -50 \\ y \in (-50{,}50) \end{cases}$$

Up:

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

Down:

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



```
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
```

<u>Control the speed of robot</u>:

To control the speed of robot we have to transform the analog values from interval (-490,490,) to digital from interval (-250,250)

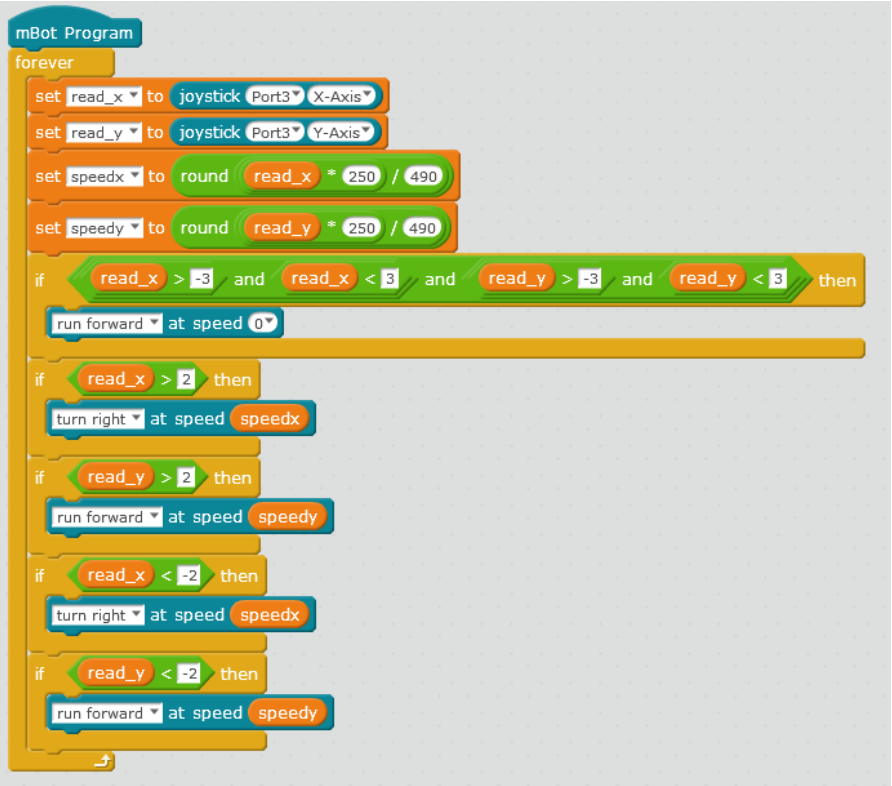
We should use proportional and the linear function $y = ax + b$

We know that 0 transform to 0, and 490 transform to 250

Let’s solve the system of equations: $\begin{cases} 0 = 0 \cdot a + b \\ 250 = 490 \cdot a + b \end{cases}$

The solution is $\begin{cases} a = \frac{250}{490} \\ b = 0 \end{cases}$

To transform analog value from joystick reading to digital values to control the speed we need to use the equation $y = \frac{250}{490}x$



```
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
```

<u>Programming in arduino language</u>

It is very common to change analog value to digital value. In Arduino language you can use the special function to transform it. It is colled map

<u>Syntax of function</u>:

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

<u>Parameters</u>

value: the number to map

fromLow: the lower bound of the value’s current range

fromHigh: the upper bound of the value’s current range

toLow: the lower bound of the value’s target range

toHigh: the upper bound of the value’s target range

The code to control the mBot (the speed is changing)

```
#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);

 }

}
```

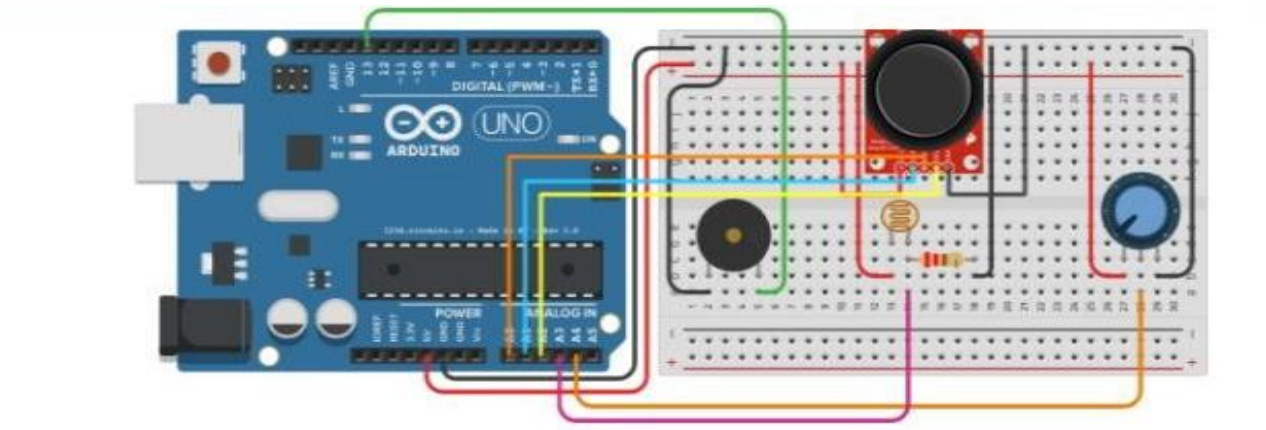
Thrid version

Now we show you how to program the game using S4A and Arduino board

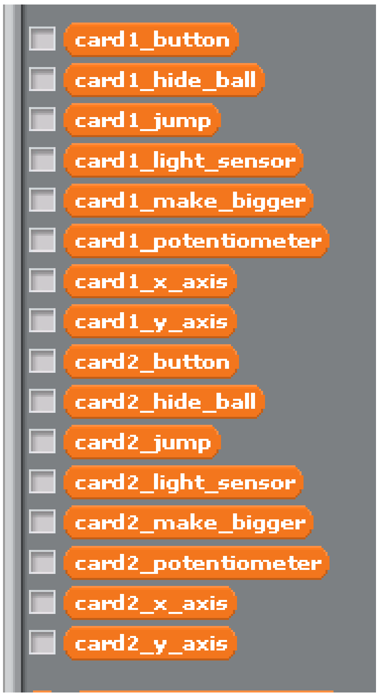
S4A is a Scratch modification that allows for simple programming of the Arduino open source hardware platform. It provides new blocks for managing sensors and actuators connected to Arduino. There is also a sensors report board similar to the PicoBoard one. More information you will find here: http://s4a.cat/

Because of the project has long code blocks, each block will be shown with pictures.

Step 1: Fritzing scheme was installed

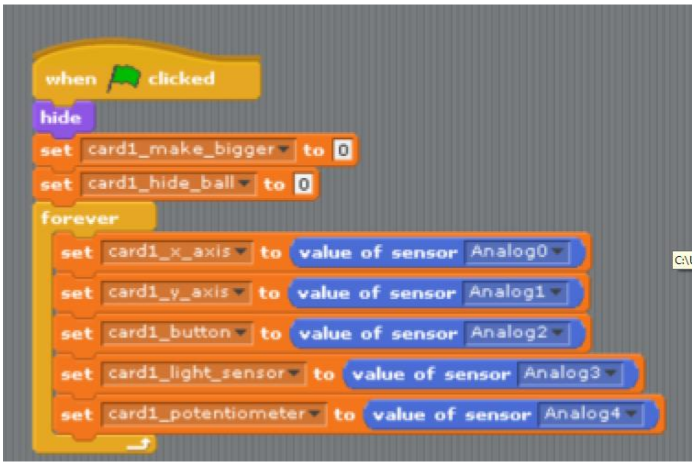


Step 2: We make variables for the first Arduino card for head-ballgame



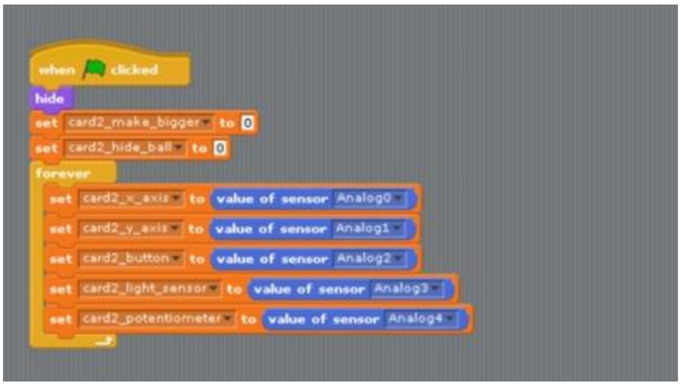
□ card1_button
□ card1_hide_ball
□ card1_jump
□ card1_light_sensor
□ card1_make_bigger
□ card1_potentiometer
□ card1_x_axis
□ card1_y_axis
□ card2_button
□ card2_hide_ball
□ card2_jump
□ card2_light_sensor
□ card2_make_bigger
□ card2_potentiometer
□ card2_x_axis
□ card2_y_axis

Let's assign pins to the sensors according to the pins on the breadboard. (Arduino 1)



```
when [flag] 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
```

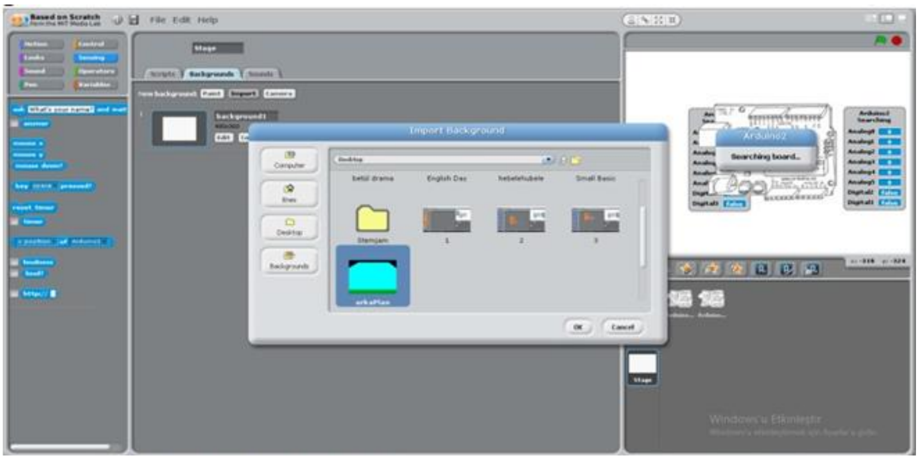
Let's assign pins to the sensors according to the pins on the breadboard (Arduino 2)



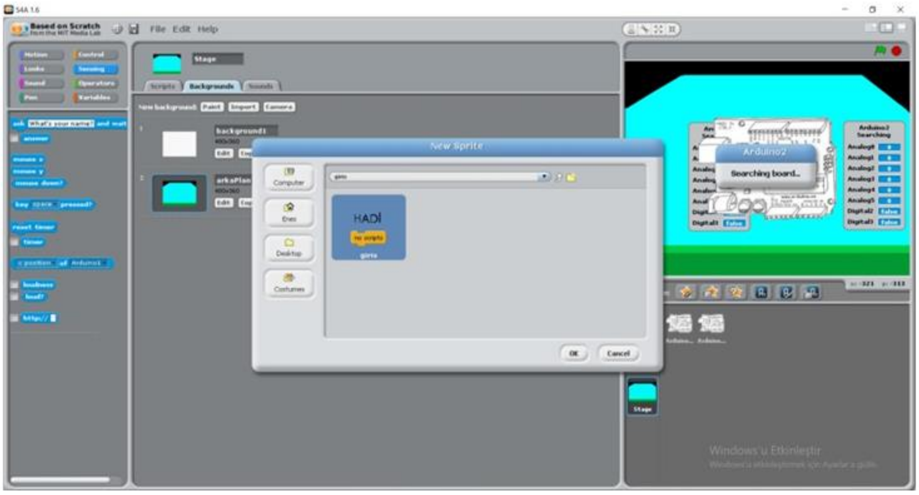
```
when [flag] clicked
hide
set card2_make_bigger to 0
set card2_hide_ball to 0
forever
    set card2_x_axis to value of sensor Analog0
    set card2_y_axis to value of sensor Analog1
    set card2_button to value of sensor Analog2
    set card2_light_sensor to value of sensor Analog3
    set card2_potentiometer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

Let's add our background for S4A.

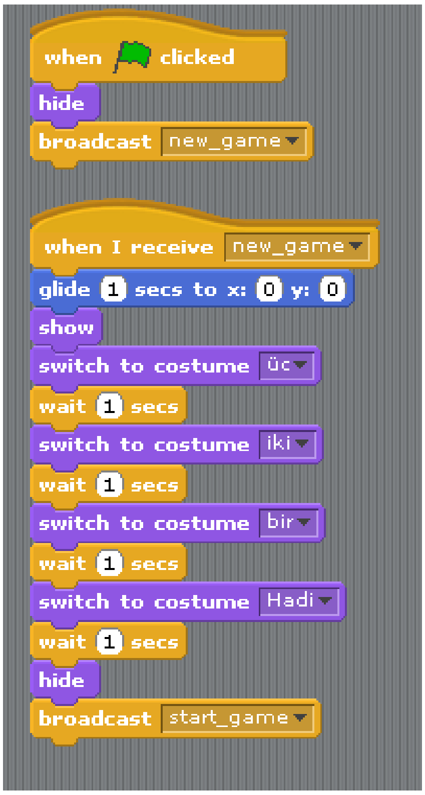
Note: Game files will be added into the “Google Drive”



We count down to start our game and we get the pictures (with the costume change)

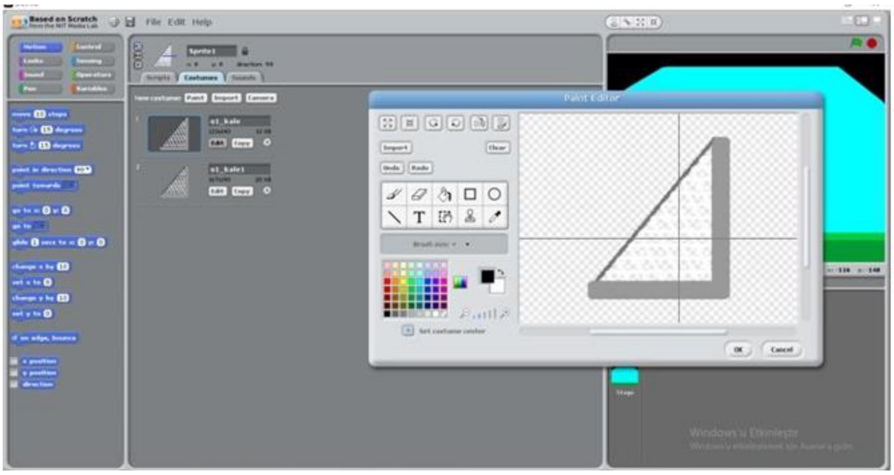


Code about sprite file



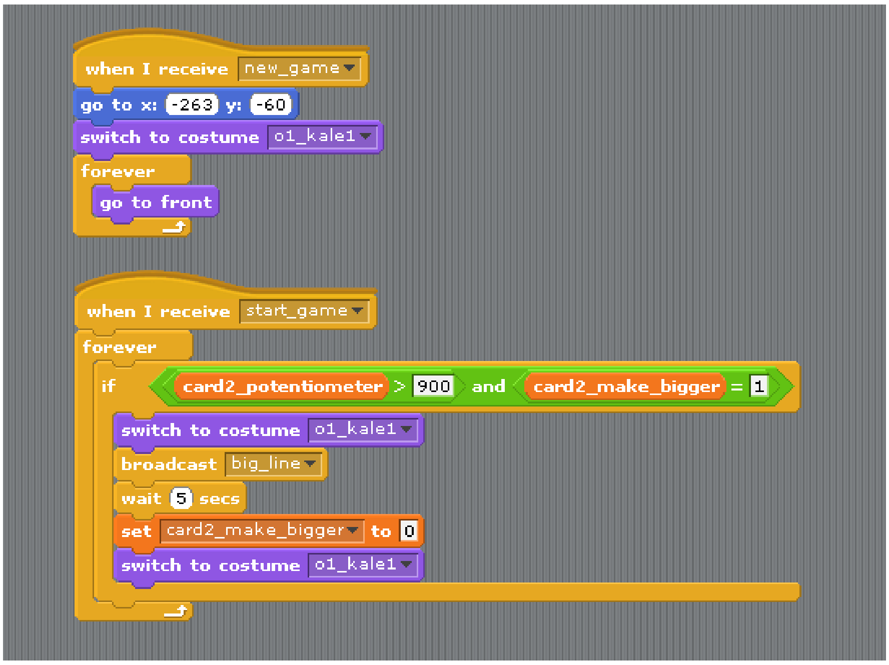
```
when 🏁 clicked
hide
broadcast new_game

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
```

We upload the GOAL to programme



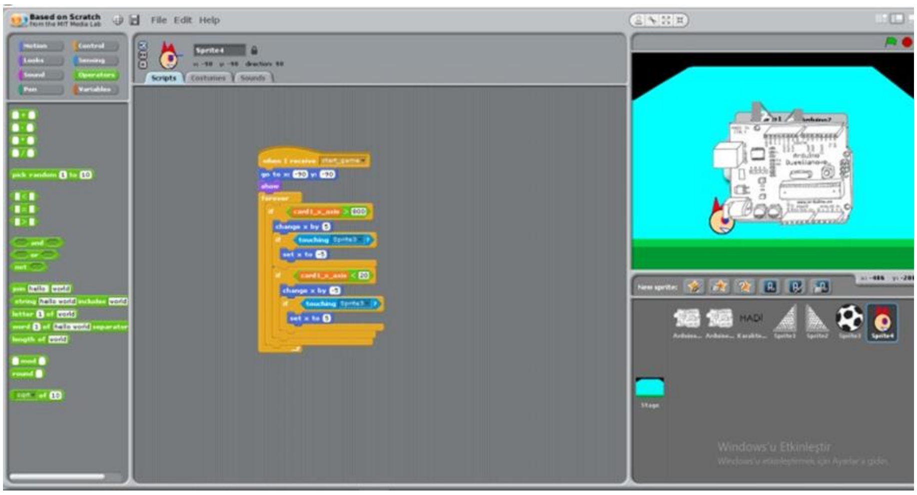
We determinate the GOAL coordinates.

We create code blocks for the blocks. We must arrange after uploading GOALS to S4A. Otherwise, it may not be where we are located.



```
when I receive new_game
go to x: -263 y: -60
switch to costume o1_kale1
forever
    go to front

when I receive start_game
forever
    if < card2_potentiometer > 900 and card2_make_bigger = 1 >
        switch to costume o1_kale1
        broadcast big_line
        wait 5 secs
        set card2_make_bigger to 0
        switch to costume o1_kale1
```

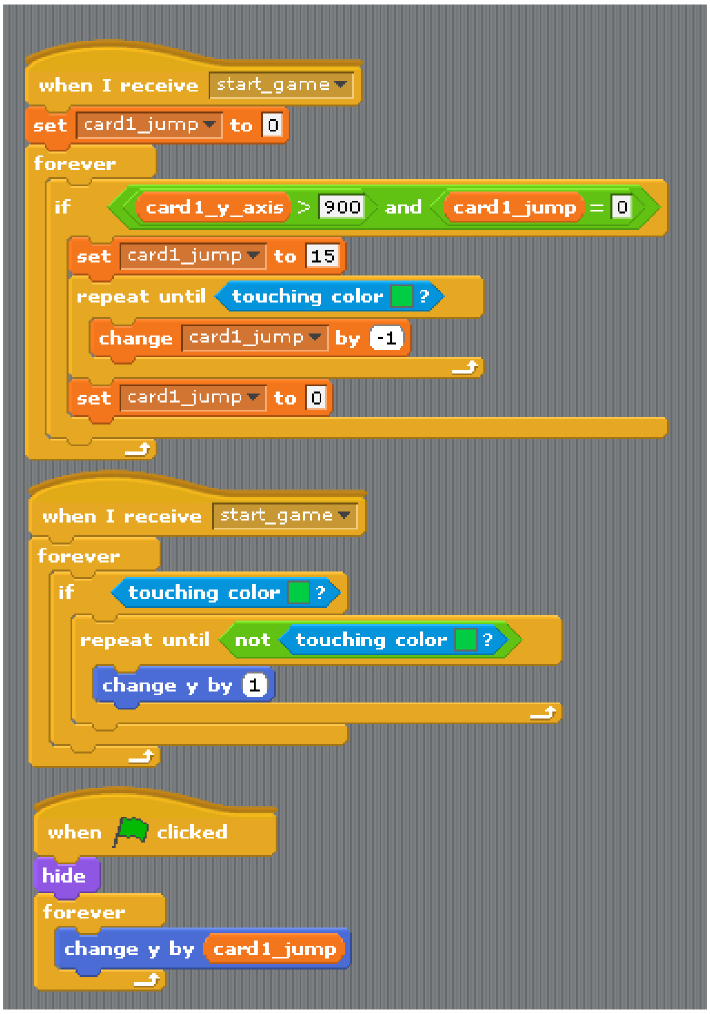
We write the blocks of code that the player needs to move.



```
when I receive start_game
go to x: -90 y: -90
show
forever
    if card1_x_axis > 800
        change x by 5
        if touching Sprite3 ?
            set x to -5
        if card1_x_axis < 20
            change x by -5
            if touching Sprite3 ?
                set x to 5
```

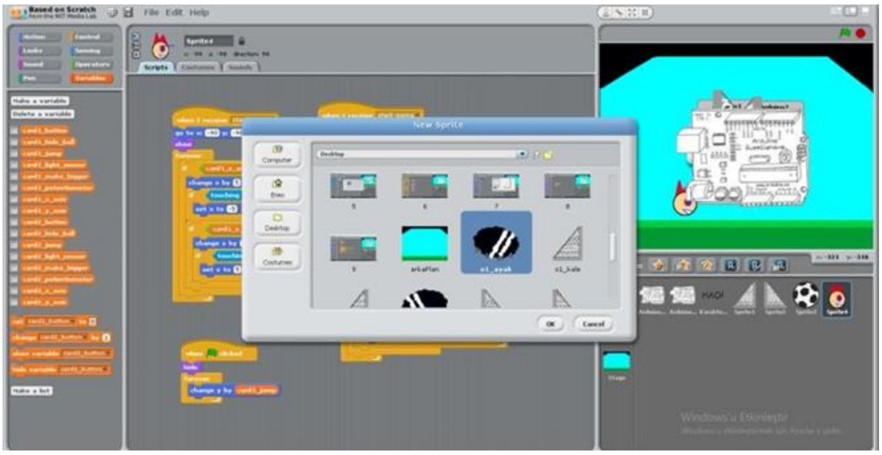
Let's continue form the code blocks for the first character to move.

The first character's jump and move code blocks that you can see on picture. Similar operations and code blocks will be repeated for the second character.

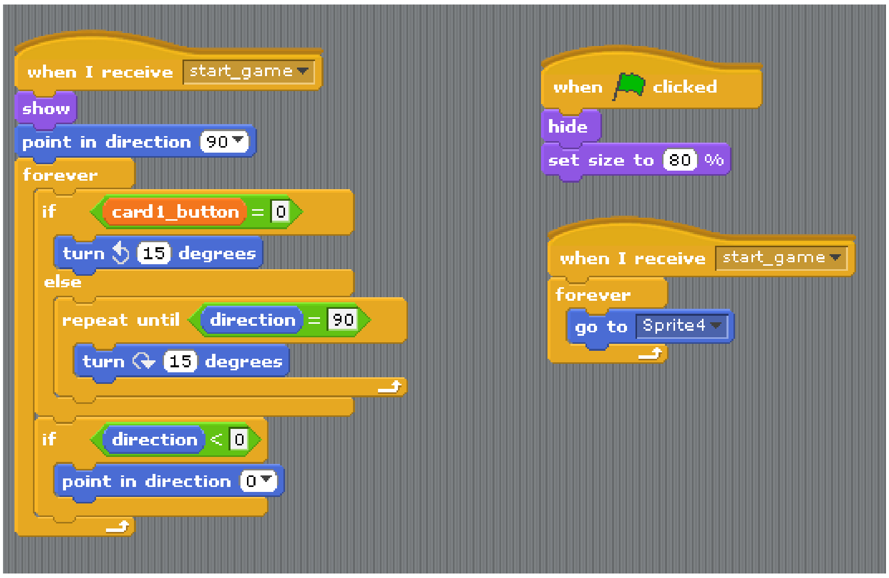


```
when I receive start_game
set card1_jump to 0
forever
    if card1_y_axis > 900 and card1_jump = 0
        set card1_jump to 15
        repeat until touching color 🟩 ?
            change card1_jump by -1
        set card1_jump to 0

when I receive start_game
forever
    if touching color 🟩 ?
        repeat until not touching color 🟩 ?
            change y by 1

when 🏁 clicked
hide
forever
    change y by card1_jump
```

We add the feet to the S4A to follow the characters.



Code blocks, “foot” characters fallow players

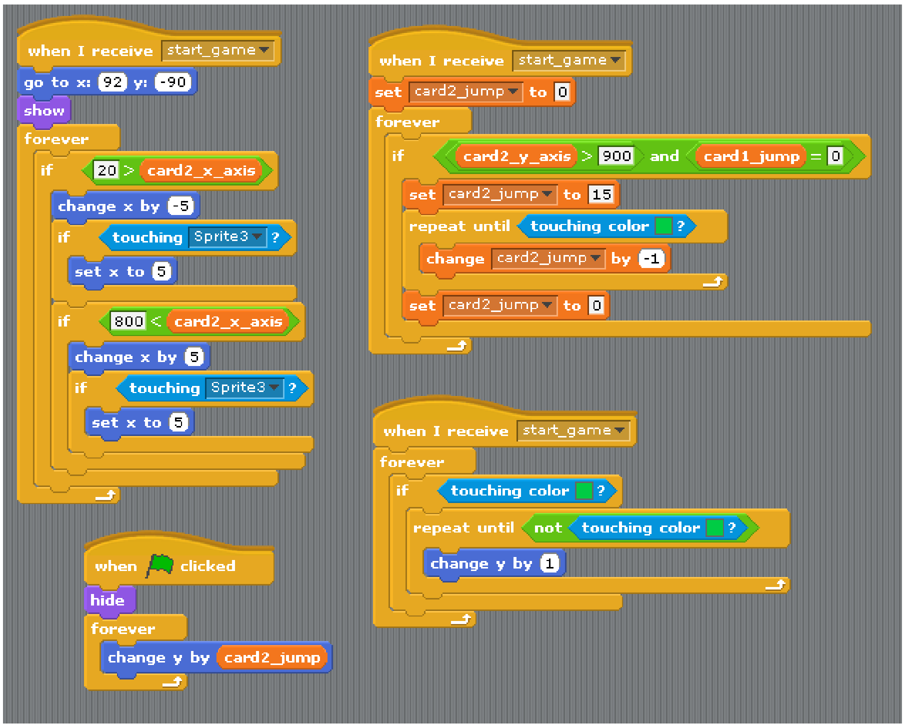


```
when I receive start_game
show
point in direction 90
forever
    if card1_button = 0
        turn ↺ 15 degrees
    else
        repeat until direction = 90
            turn ↻ 15 degrees
    if direction < 0
        point in direction 0
```

```
when 🏁 clicked
hide
set size to 80 %
```

```
when I receive start_game
forever
    go to Sprite4
```

The second character’s moves and jumps codes



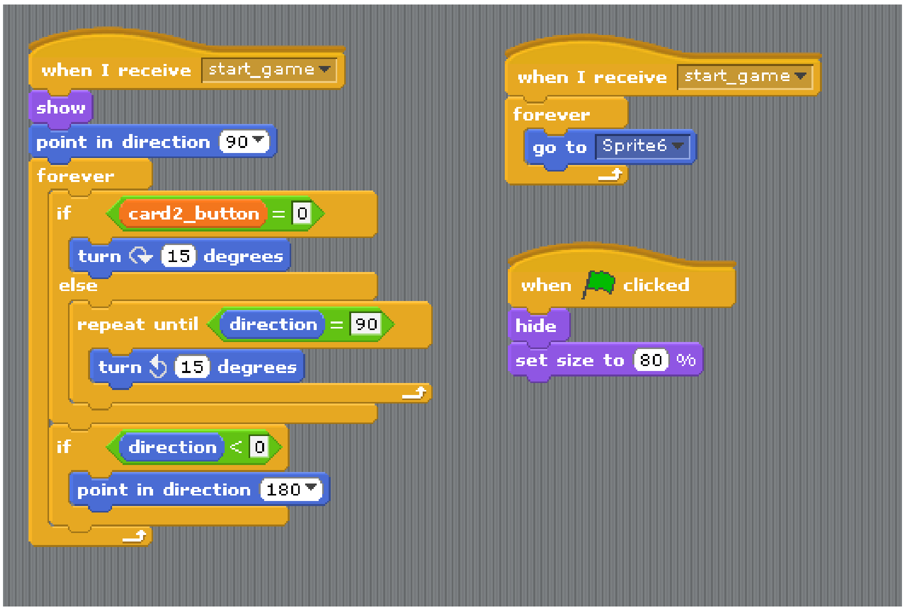
```
when I receive start_game
go to x: 92 y: -90
show
forever
    if 20 > card2_x_axis
        change x by -5
        if touching Sprite3 ?
            set x to 5
        if 800 < card2_x_axis
            change x by 5
            if touching Sprite3 ?
                set x to 5
```

```
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
```

Code blocks followed by the second character of the second leg.

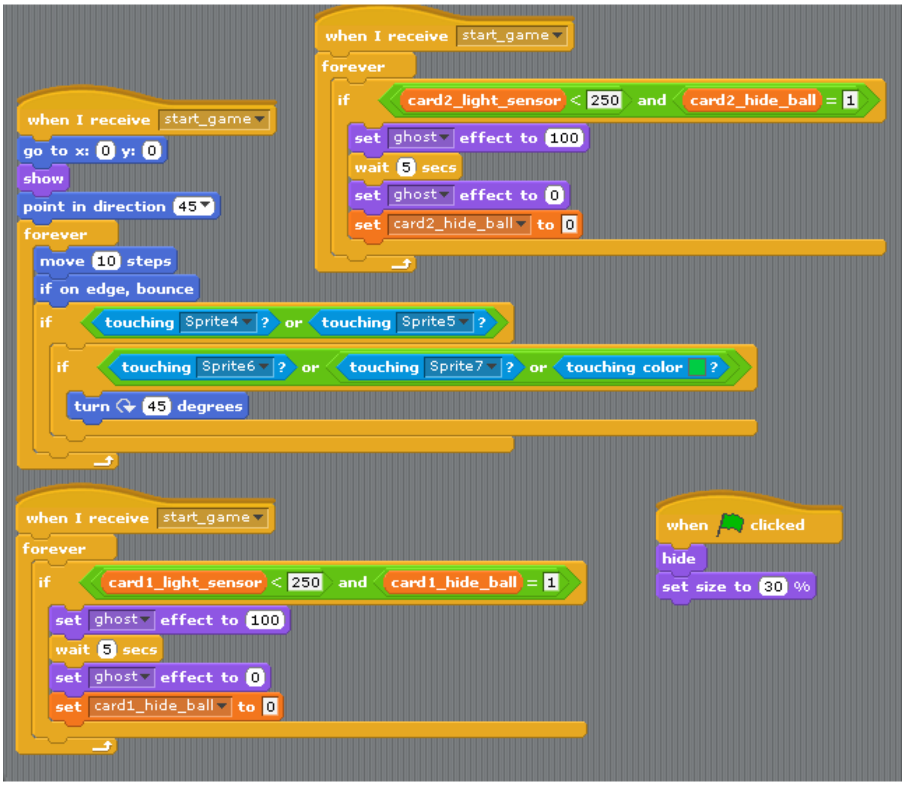


```
when I receive start_game
show
point in direction 90
forever
    if <(card2_button) = 0>
        turn ↻ 15 degrees
    else
        repeat until <(direction) = 90>
            turn ↺ 15 degrees
    if <(direction) < 0>
        point in direction 180
```

```
when I receive start_game
forever
    go to Sprite6
```

```
when 🏁 clicked
hide
set size to 80 %
```

We write the ball codes after we add the ball character from the library. The top character will be affected by both player characters, by the edges and by the goal lines.



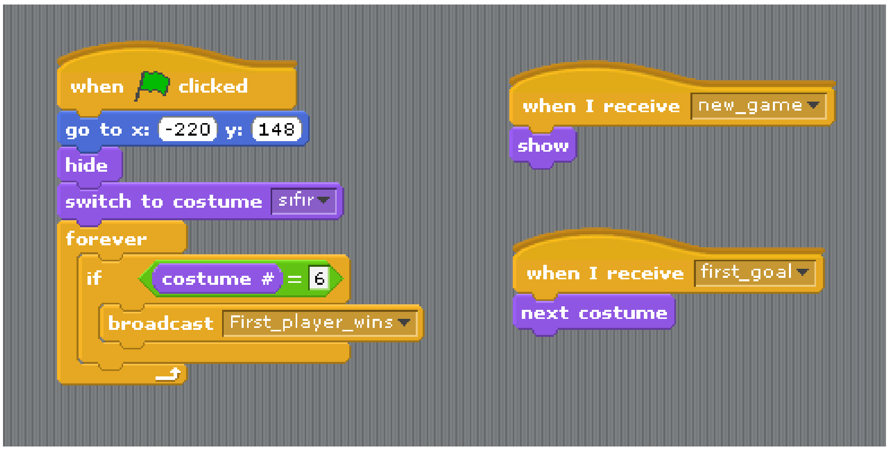
```
when I receive start_game
forever
    if <<(card2_light_sensor) < 250> and <(card2_hide_ball) = 1>>
        set ghost effect to 100
        wait 5 secs
        set ghost effect to 0
        set card2_hide_ball to 0
```

```
when I receive start_game
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 <<(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
```

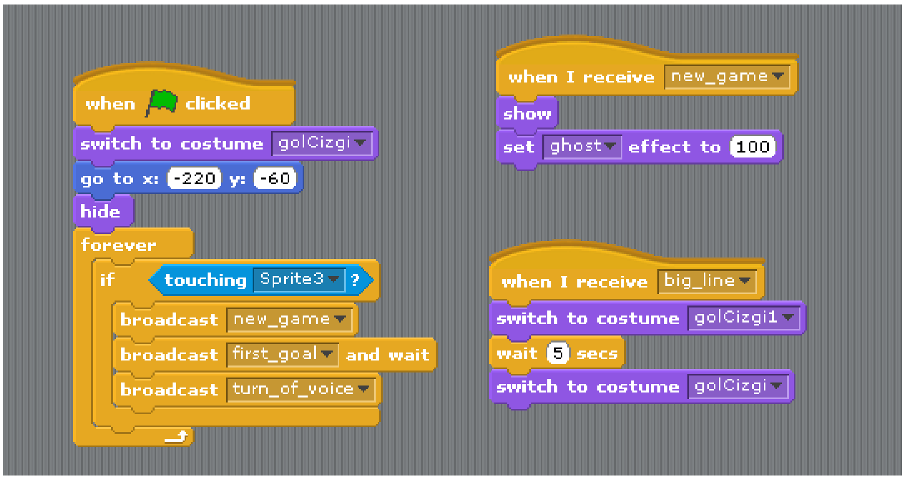
```
when 🏁 clicked
hide
set size to 30 %
```

We use the "Broadcast" method so that characters can follow each other in game. Here we also add the necessary communications. The black lines on the screen are the characters make contact with ball.



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

when I receive new_game
show

when I receive first_goal
next costume
```

This is the code for “line spirit” in the middle of the goal



```
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
```

STUDENTS’ EVALUATION

When 5-6th grade students played many menus on scratch.

They are also very pleased that they designed the game.

BIBLIOGRAPHY

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

SCALABILITY

The first part is the basic knowledge about analog and digital values – it can be used to another problems and sensors.

The game is bigger project to students who have more programming skills.

MORE INFORMATION

Tip about the game: The disadvantage of this work is that the jumper cables on the breadboard or on the arduino can come out from time to time.