

SCENARIO

Title	CAT vs MICES	
Summary	This activity consists in that the students know the vector axes and acquire skill with the joystick to play the video game. In addition, the student will have to program a video game that will be connected to the Arduino and the Joystick.	
Author/s	AIJU	

DIDACTIC OBJECTIVES

- Know the vector axes.
- Programming a Videogame connected to Arduino and the Joystick.

Physics **X** Mathematics **X** Information Technology Robotics Programming **X**

Education Level: 10-12 years 12-14 years **X**

PROBLEM STATEMENT

At first, programming and the Arduino world can be difficult, but if students are motivated with the incentive to create their own video game and even their own console, the interest and desire to learn is awakened.

BOM (Bill Of Materials needed)

- Arduino Device
- (x4) Cables
- Joystick

„InnoExperiment – Innovative Approach to Teaching through Experiment”

Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)

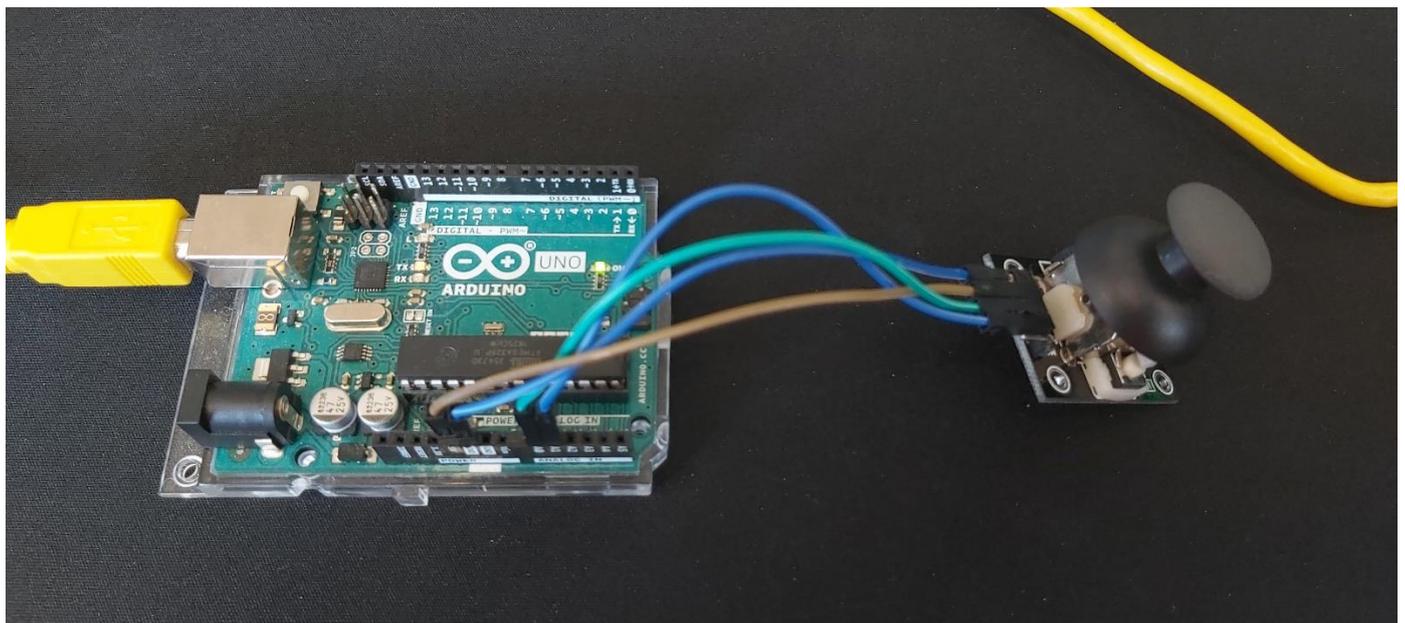
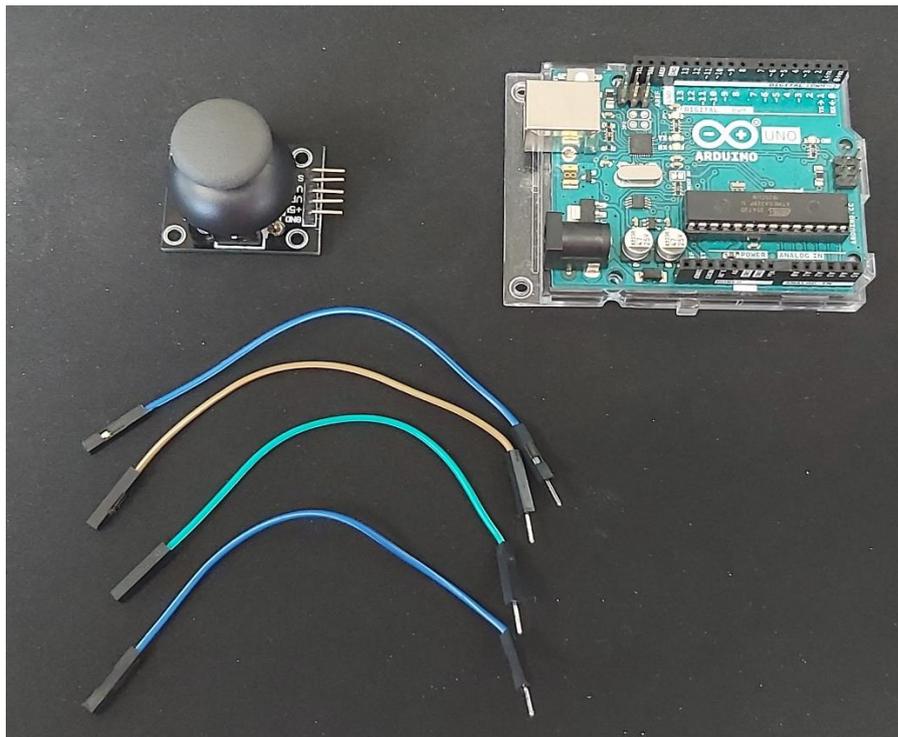


Erasmus+



InnoExperiment

INNOVATIVE APPROACH TO TEACHING THROUGH EXPERIMENT



„InnoExperiment – Innovative Approach to Teaching through Experiment”

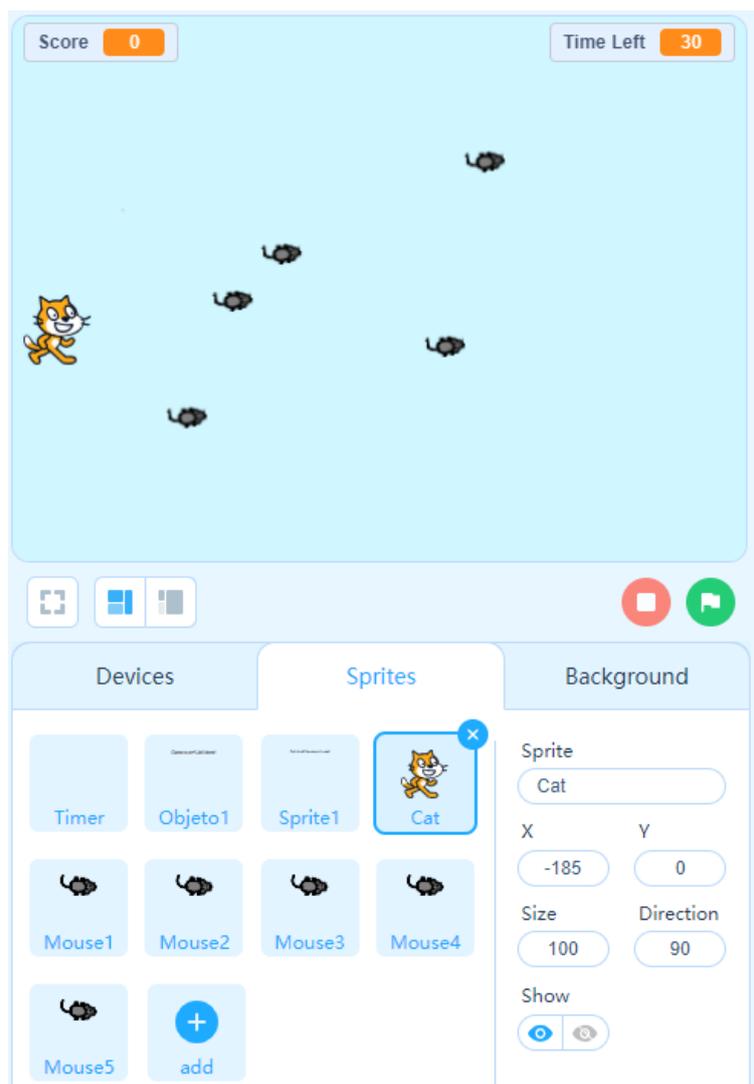
Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)



ACTIVITY DESCRIPTION

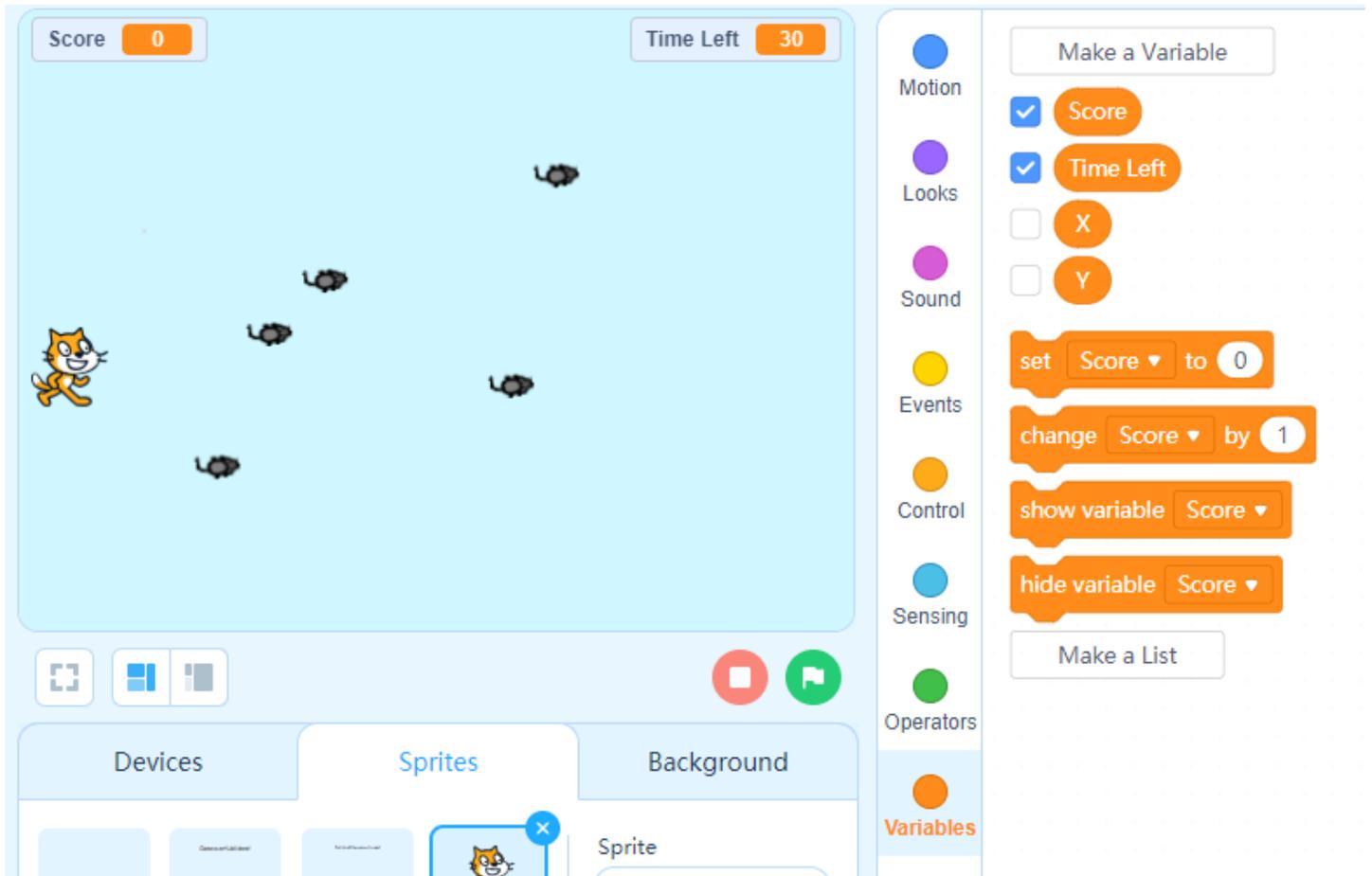
For the development of the activity, we will use software that allows us to unify the game developed in Scratch with the use of the Arduino board. In this case, we have used the mBlock software: (<https://mblock.makeblock.com/en-us/>)

First of all, we will make the graphic composition of the activity:



We add the “Score” and “Time Left” variables, for the calculation of the score and the remaining time.

In addition, we will create the X and Y variables that will be responsible for collecting the information generated by the Arduino board.



Once we have the graphic composition and the variables created, we will start with the programming:

1. In the first place we will program the Arduino board, where we will save in the variables X (movement in the X axis) and Y (movement in the Y axis) the data that when moving the Joystick is generated, so that this data is sent to the game character at through the two variables:

```

when clicked
  forever
    set X to read analog pin (A) 0
    set Y to read analog pin (A) 1
  
```

2. When the joystick is in the centered position, the values for variable X are usually between 505 and 510, and for variable Y, it is usually between 510 and 515, so we will have to program 4 movements:

- If we want to move to the left, we will tell the system that when X is greater than 510 send "go_left" to the programming of the "Cat" object:

```

if X > 510 then
  broadcast go_left
  
```

- If we want to move to the right, when X is less than 505 the object will move to the right, through the call "go_right":

```

if X < 505 then
  broadcast go_right
  
```

- If we want the object to move up, through the "go_up" call, when Y is greater than 515 this movement will occur:

```

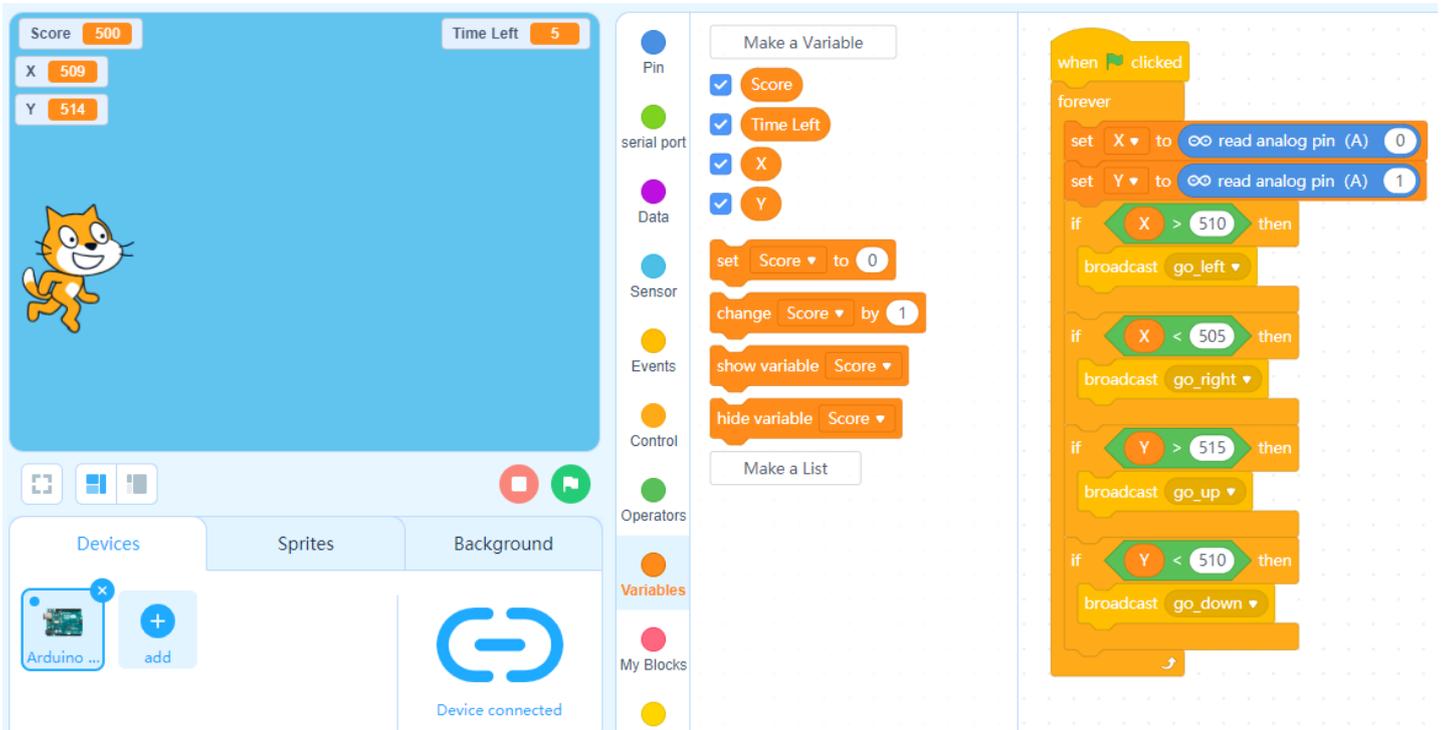
if Y > 515 then
  broadcast go_up
  
```

- And finally, when Y is less than 510, the object will scroll down through the "go_down" call:

```

if Y < 510 then
  broadcast go_down
  
```

3. Therefore, in this way the programming referring to the Arduino board would be:



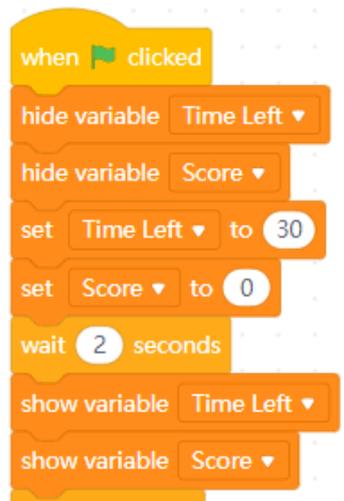
The screenshot shows the Scratch IDE interface. On the left, the game stage displays a score of 500, X at 509, and Y at 514. A timer shows 5 seconds left. The Scratch code on the right is as follows:

```

when clicked
  forever
    set X to read analog pin (A) 0
    set Y to read analog pin (A) 1
    if X > 510 then
      broadcast go_left
    if X < 505 then
      broadcast go_right
    if Y > 515 then
      broadcast go_up
    if Y < 510 then
      broadcast go_down
  
```

The code also includes a 'Make a Variable' block with 'Score' checked, and 'set Score to 0', 'change Score by 1', 'show variable Score', and 'hide variable Score' blocks.

4. Next, we proceed to program the chronometer, which will count down. Really, the important thing is to start the variable "Time Left" in the seconds that we want the chronometer to have. In this case, it will be 30 seconds. Also, we indicate that the variables "Score" and "Time Left" are shown in the game so that the user knows his score and the time he has left to finish capturing the mice:



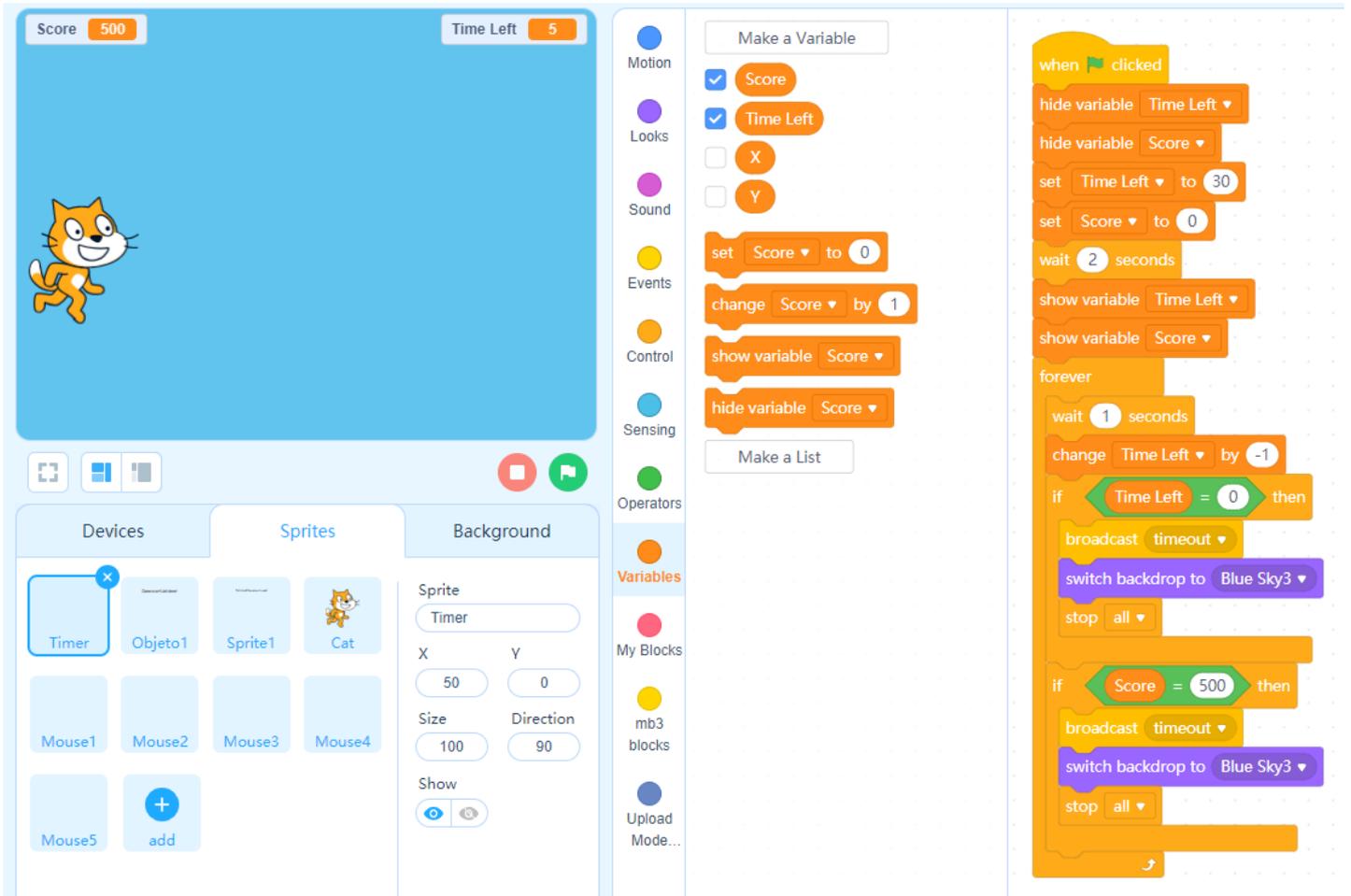
```

when clicked
  hide variable Time Left
  hide variable Score
  set Time Left to 30
  set Score to 0
  wait 2 seconds
  show variable Time Left
  show variable Score
  
```

5. In the loop, we will indicate that every second deducts one on the chronometer and if the time reaches 0 or if the player captures all 5 mices, the game stops and the game background is changed:

```
forever
  wait 1 seconds
  change Time Left by -1
  if Time Left = 0 then
    broadcast timeout
    switch backdrop to Blue Sky3
    stop all
  if Score = 500 then
    broadcast timeout
    switch backdrop to Blue Sky3
    stop all
```

6. The programming for this section would be as follows:

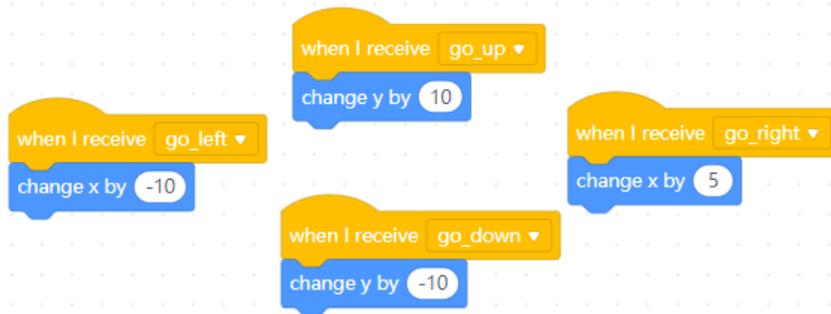


The screenshot shows the Scratch programming environment. The stage displays a score of 500 and a time left of 5. A cat sprite is visible. The script area contains the following code:

```

when clicked
  hide variable Time Left
  hide variable Score
  set Time Left to 30
  set Score to 0
  wait 2 seconds
  show variable Time Left
  show variable Score
  forever
    wait 1 seconds
    change Time Left by -1
    if Time Left = 0 then
      broadcast timeout
      switch backdrop to Blue Sky3
      stop all
    if Score = 500 then
      broadcast timeout
      switch backdrop to Blue Sky3
      stop all
  
```

7. Now we will proceed with the programming of the "Cat". When the game starts, the cat will wait to receive the orders sent by the Arduino board through the previously established messages:



The script for the Cat sprite is as follows:

```

when I receive go_up
  change y by 10
when I receive go_left
  change x by -10
when I receive go_right
  change x by 5
when I receive go_down
  change y by -10
  
```

Values 10 or -10 indicate movement speed.

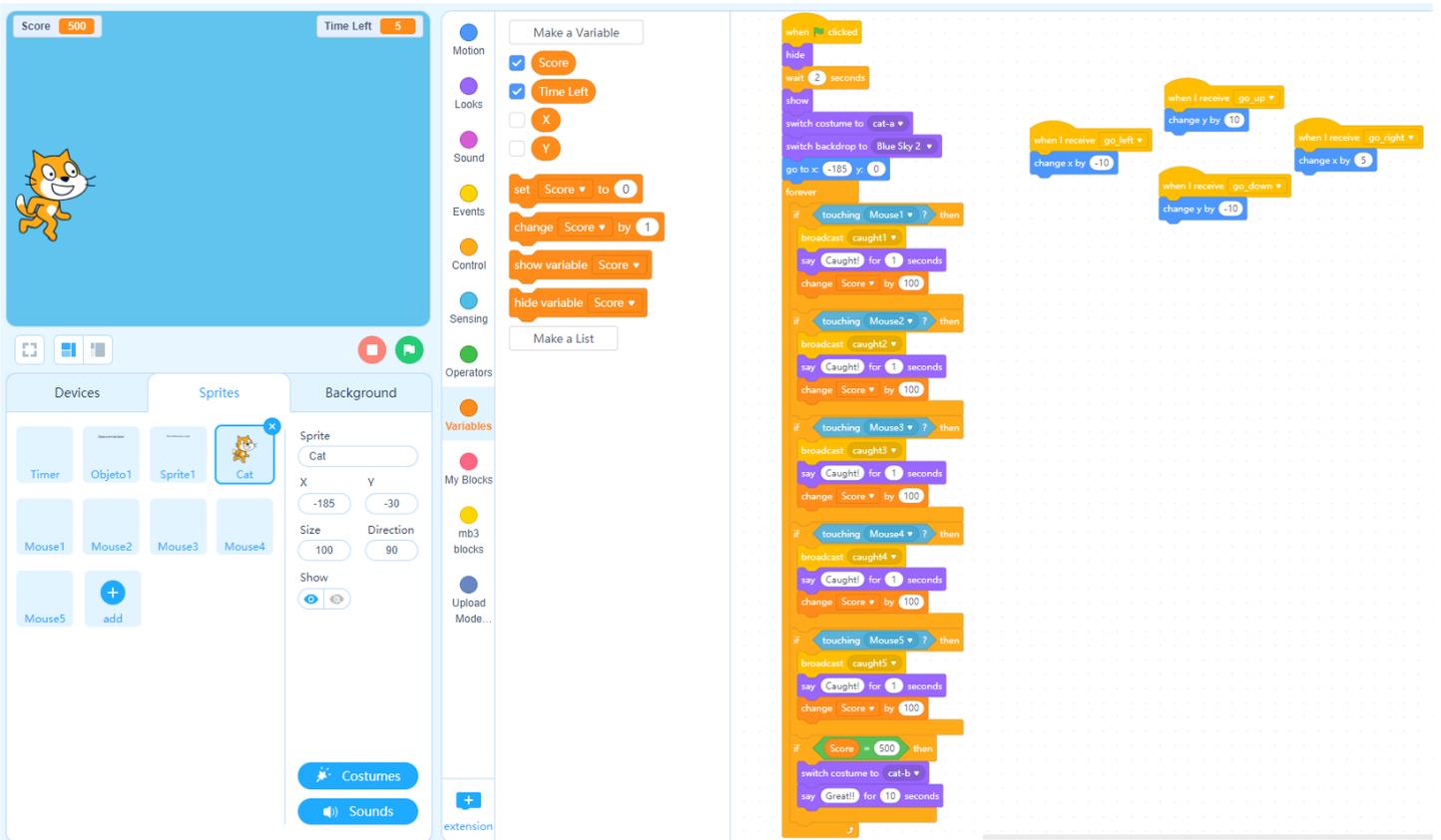
8. Then we will create a loop where we will program the cat's reaction when capturing a mouse: "Caught!":

```

forever
  if touching Mouse1 ? then
    broadcast caught1
    say Caught! for 1 seconds
    change Score by 100
  if touching Mouse2 ? then
    broadcast caught2
    say Caught! for 1 seconds
    change Score by 100
  if touching Mouse3 ? then

```

9. The complete "Cat" programming would be the following:



The screenshot displays the Scratch programming environment. On the left, the stage shows a cat sprite with a score of 500 and 5 seconds left. The 'Sprites' panel shows the 'Cat' sprite selected. The 'Scripts' panel contains the following code:

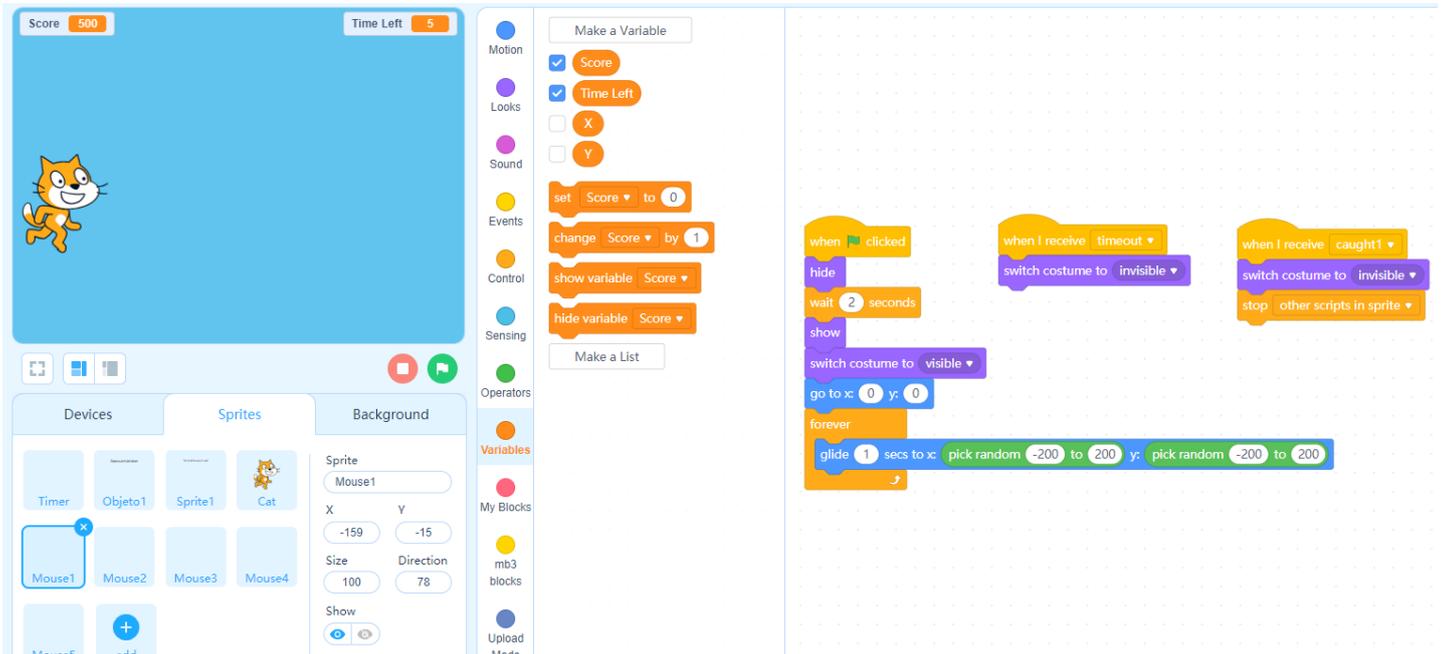
```

when clicked
  hide
  wait 2 seconds
  show
  switch costume to cat-a
  switch backdrop to Blue Sky 2
  go to x: -185 y: 0
  forever
    if touching Mouse1 ? then
      broadcast caught1
      say Caught! for 1 seconds
      change Score by 100
    if touching Mouse2 ? then
      broadcast caught2
      say Caught! for 1 seconds
      change Score by 100
    if touching Mouse3 ? then
      broadcast caught3
      say Caught! for 1 seconds
      change Score by 100
    if touching Mouse4 ? then
      broadcast caught4
      say Caught! for 1 seconds
      change Score by 100
    if touching Mouse5 ? then
      broadcast caught5
      say Caught! for 1 seconds
      change Score by 100
    if Score = 500 then
      switch costume to cat-b
      say Great! for 10 seconds

```

The 'Variables' panel shows the 'Score' variable set to 0, with 'change Score by 1' blocks. The 'Motion' panel shows 'go to x: -185 y: 0' and 'wait 2 seconds' blocks. The 'Looks' panel shows 'switch costume to cat-a' and 'switch backdrop to Blue Sky 2' blocks. The 'Stage' panel shows 'show' and 'hide' blocks. The 'Stage' panel also shows 'Score' and 'Time Left' variables.

10. Finally, we will program the movements of the mice:



The screenshot shows the Scratch IDE interface. On the left, there's a game preview window with a score of 500 and time left of 5. Below it are the Sprites and Background panels. The Sprites panel shows a Cat sprite and several Mouse sprites. The Background panel shows a blue background. On the right, the Script Editor shows a 'forever' loop with a 'glide' block. The 'glide' block is set to 1 second, with x and y coordinates set to 'pick random -200 to 200'. Other blocks in the script include 'when clicked', 'hide', 'wait 2 seconds', 'show', 'switch costume to visible', and 'go to x: 0 y: 0'.

The most interesting block would be the one that makes the mouse move randomly across the game screen:



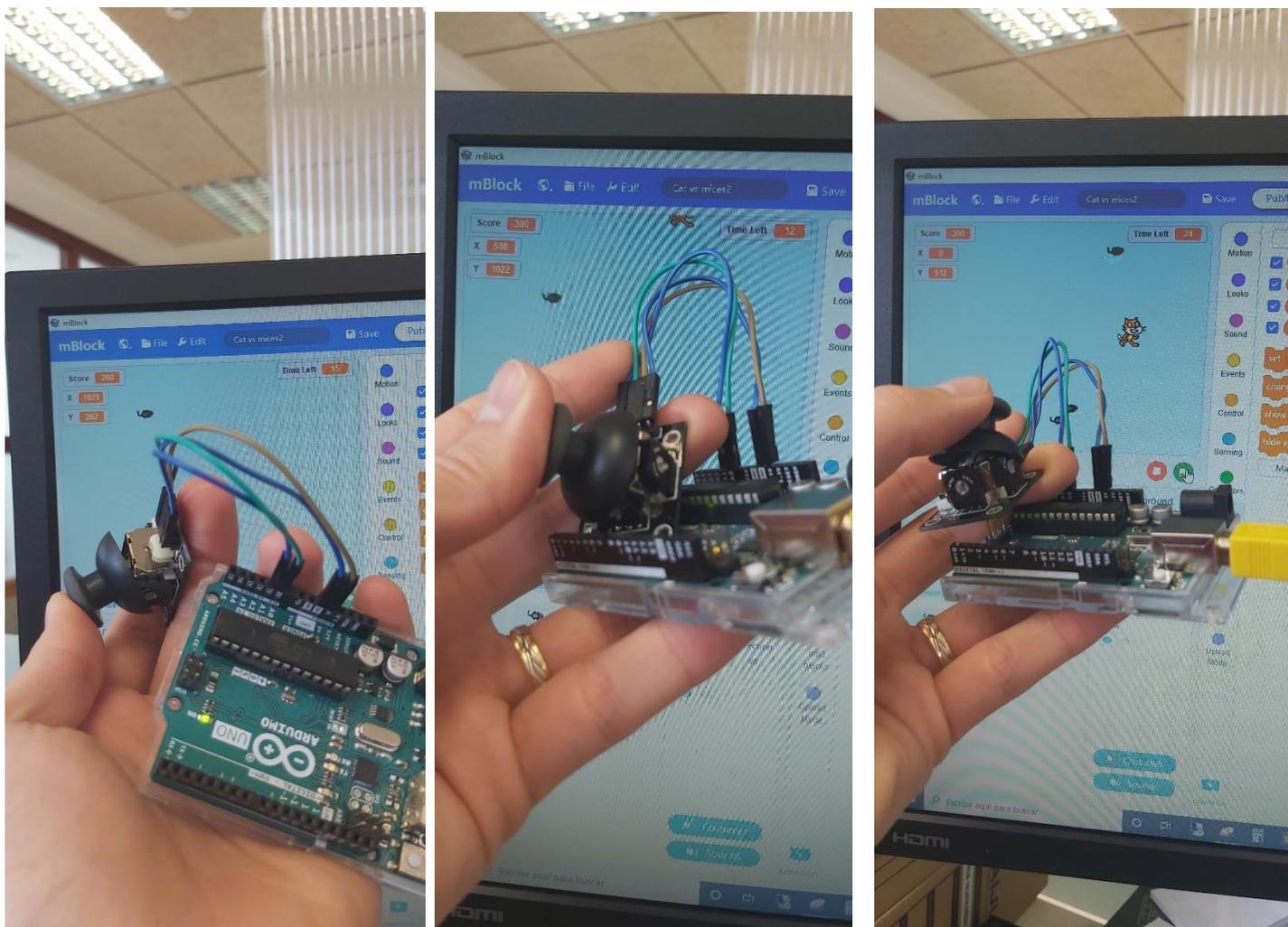
The image shows a close-up of a 'forever' loop block containing a 'glide' block. The 'glide' block is set to 1 second, with x and y coordinates set to 'pick random -200 to 200'.

SCALABILITY

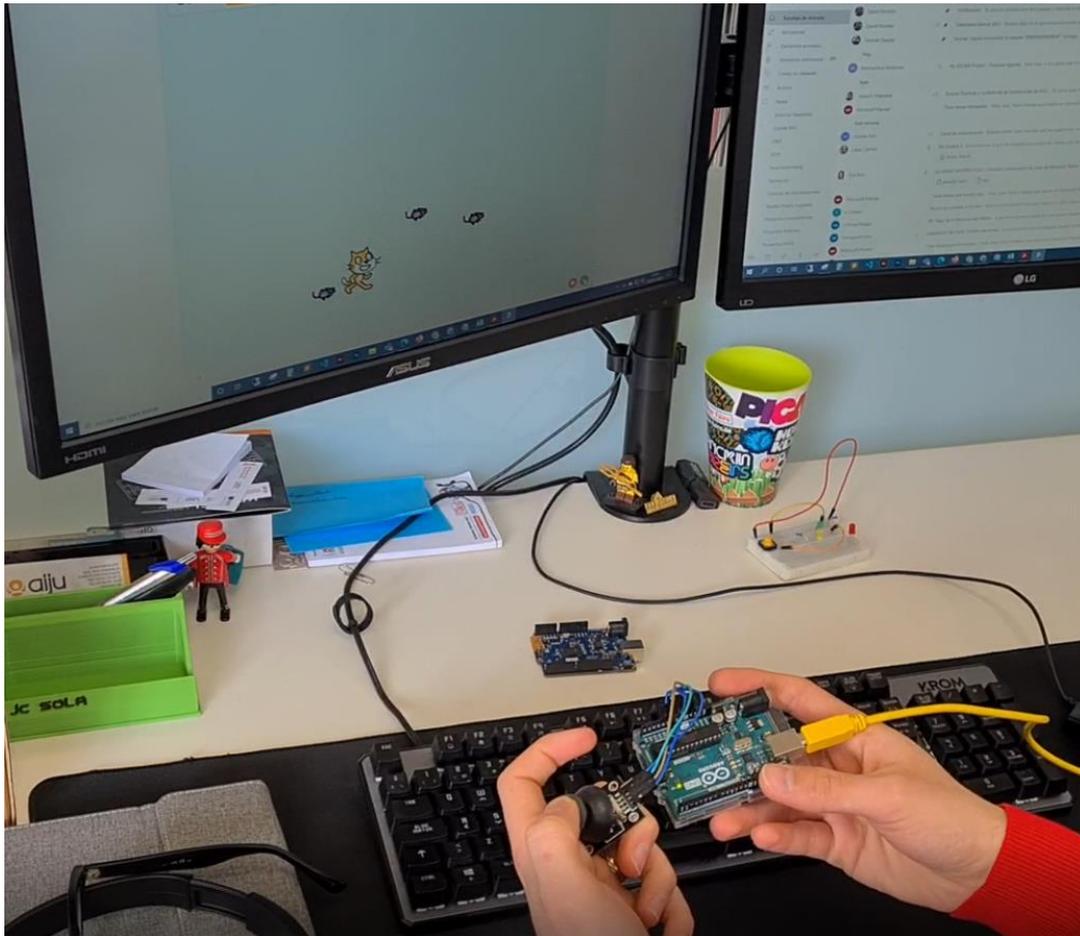
Regarding the concept of scalability, the complexity could be increased by adding more objects in the game or even buttons on the Arduino board that allow some special function for the video game character.

„InnoExperiment – Innovative Approach to Teaching through Experiment”
Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)

RESOURCES



„InnoExperiment – Innovative Approach to Teaching through Experiment”
Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)



„InnoExperiment – Innovative Approach to Teaching through Experiment”
Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)

„InnoExperiment – Innovative Approach to Teaching through Experiment”
Project Leader: Zespół Szkolno – Przedszkolny w Goniądzu (ZSP)

