MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"NOVOSIBIRSK NATIONAL RESEARCH UNIVERSITY
STATE UNIVERSITY"
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

15.03.06 - Mechatronics and Robotics
Focus (profile): Artificial Intelligence

TERM PAPER

Job topic: **THE “STYLO” GAME**

Malyarchuk Stanislav, 23931
Chernodolya Valeria, 23931

Novosibirsk
2024

CONTENTS

1. INTRODUCTION......3
2. REQUIREMENTS DEFINITION......4
3. ANALOGUES......5
4. HARDWARE......7
5. SOFTWARE......18
6. CONCLUSION......19
7. SOURCES USED IN DEVELOPING......20

1. INTRODUCTION

The aim of this project is to create a game using the logic circuit design and modeling program "Logisim", the processor "CdM 8" and the program CocoIDE, which writes assembly language programs for the processor.

The name of the game is "Stylo". It is necessary to use the keyboard to control the car and avoid objects on the way.

According to the aim, it was necessary to solve the following problems:

- To examine some games and their analogs and determine the direction of development;
- To study the programs "Logisim", "CocoIDE" and the assembly language;
- To learn about the "CdM 8" processor, its commands and interaction with the development environment;
- Define functional requirements.

2. REQUIREMENTS DEFINITION

The following are the functional requirements that had to be realized:

1. Controlled left-to-right movement of the car;
2. Uncontrolled movement of obstacles towards the player from top to bottom;
3. Keyboard control;
4. Scoring;
5. Playability;
6. Ability to pause and restart the game;
7. Speeding up and slowing down the car.

3. ANALOGUES

In order to choose a game to be realized, a couple of different genres were reviewed, but the choice was made on the "arcade" and "endless runner" genre, in which you have to overcome obstacles to complete the level.

1. One of the most popular games in this genre is "Crossy Road", in which the chicken (or other characters) needs to pass through busy highways, rivers, railroad tracks and so on.

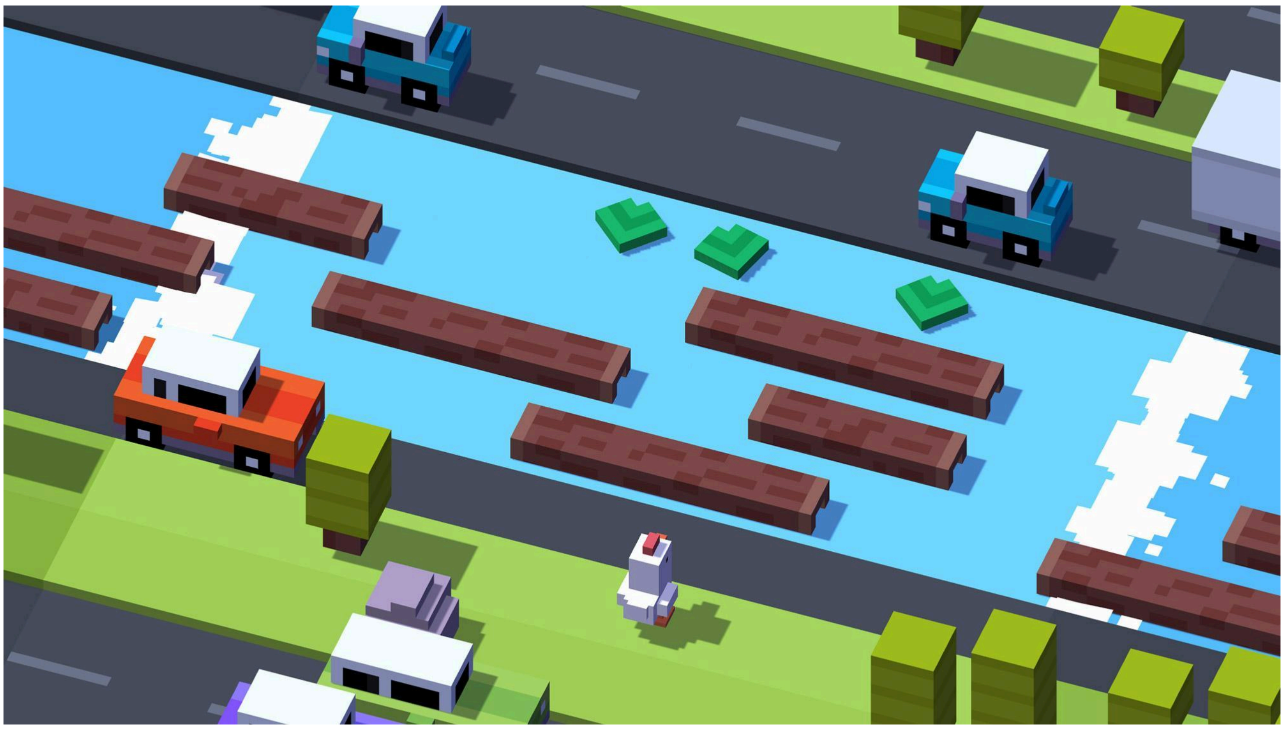


Figure 1. - Screenshot of the game "Crossy Road"

2. Another one of the popular games in this genre is Temple Run. In this game, explorers try to escape from the temple, but they are stalked by demonic monkeys.

Figure 2. - Screenshot of the game "Temple Run"

After studying and analyzing these examples, our team decided to create a project based on the idea of endless arcade runners. Because of limited resources, we needed a not very complicated version that could be made with the help of CdM-8 and our knowledge in the field of circuitry.

4. HARDWARE

The hardware part of the project consists of logic circuits made in the program “Logisim”, in which they can be easily created and edited. Standard Logisim libraries were used, as well as the library “CdM-8-mark5-banks” for working with the processor “CdM 8”.

The interaction of the player with the game takes place on the circuit “Main”.

Main

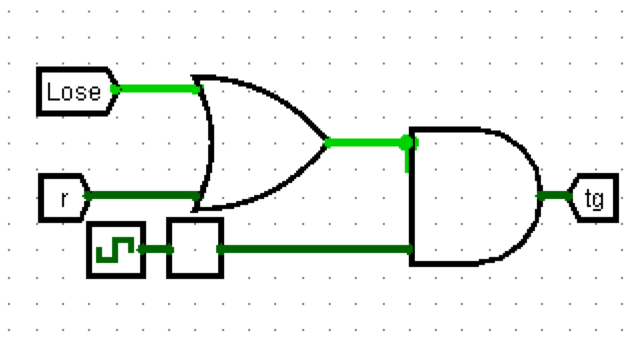


Figure 3. Part of the “Main” sub-circuit of the circuit

In this part of the circuit, there is a label “Lose”. When “Lose” = 1, the program executes, everything is fine. The “r” label is transferred from the “Keyboard” circuit. The most important thing in this circuit is the clock. It passes its value to the “tacktDel” subcircuit, which splits it 16 times and then the “AND” element checks that we have either “Lose” or “r” running and the clock is running. The result is passed through the “tg” tunnel to the keyboard and to almost all circuits, as this is the frequency the project is running at.

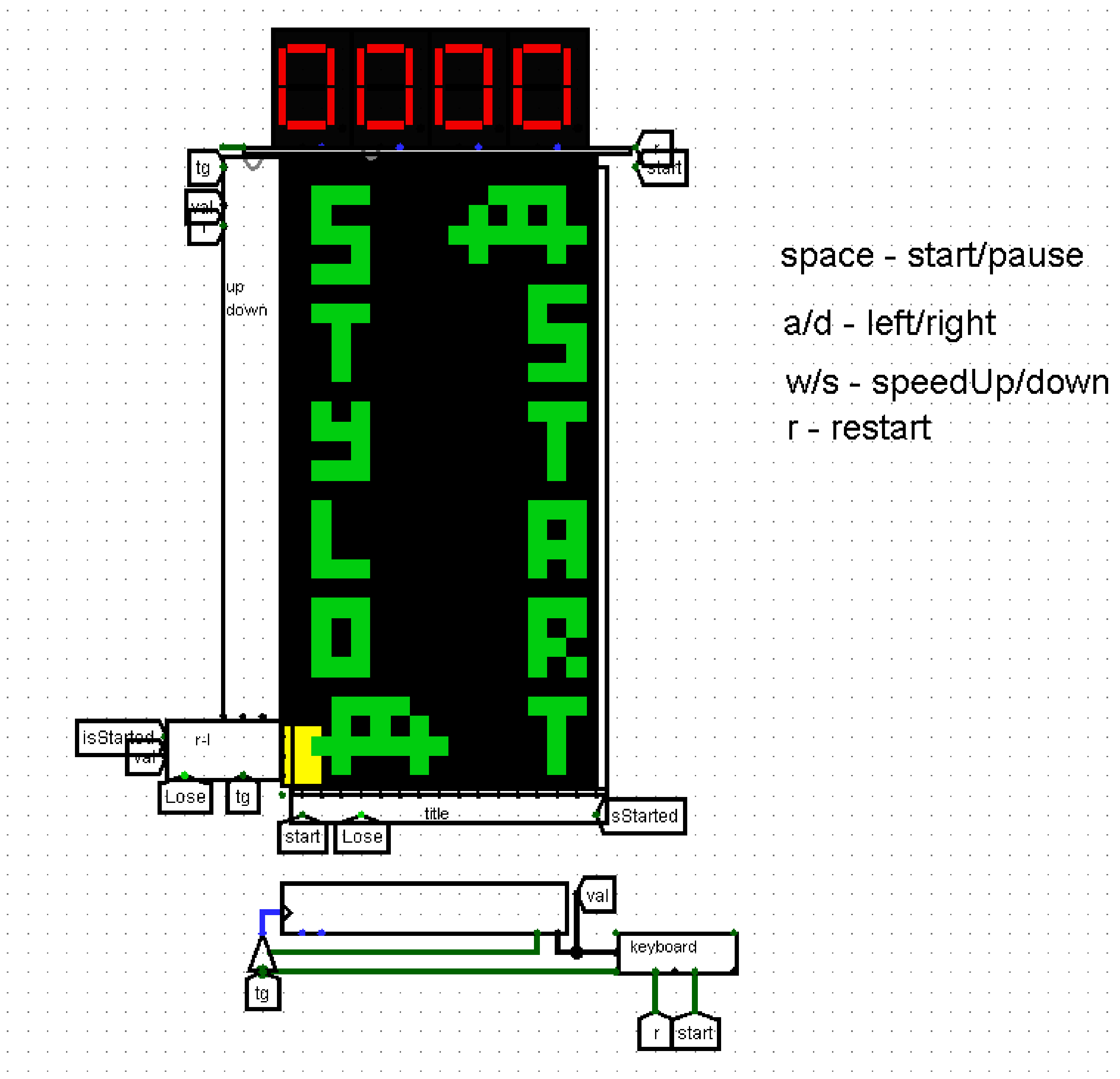


Figure 4. Part of the subcircuit of the "Main" circuit

Now let's consider the main LED matrix. On the right of it is the keyboard. The keyboard passes the value in the form of ASCII numeric code (American standard code for information interchange) to the "Keyboard" circuit. In the "Keyboard", when we press the "Space" key, we feed the "Start" label, which in turn gives the value for the "r" label. The "Up-down" and "Right-left" subcircuits are used for animation.

Keyboard

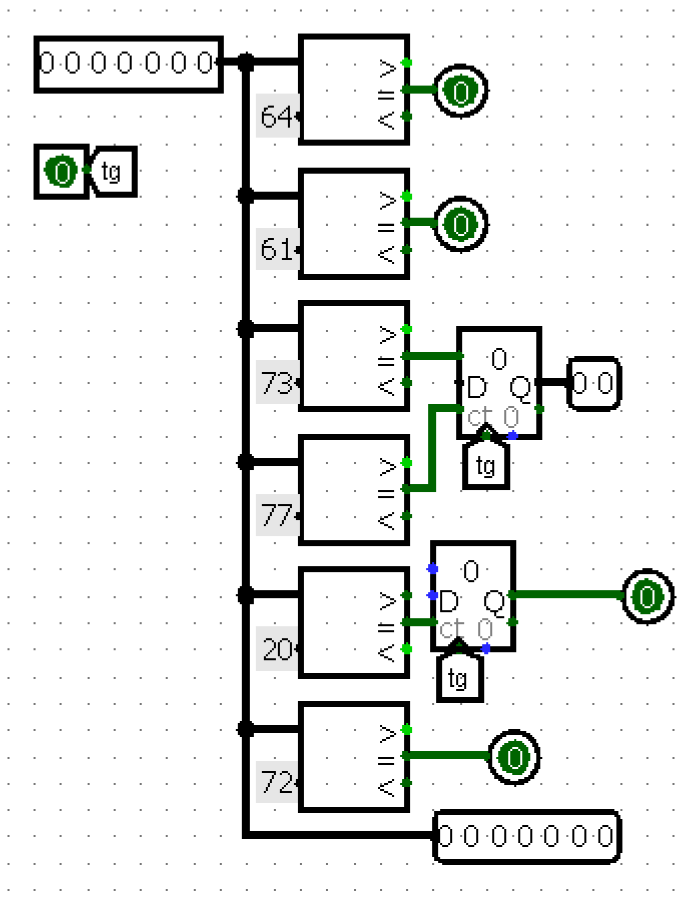


Fig. 5. "Keyboard" subcircuit

In the "Keyboard" subcircuit there is a comparison with numbers. There is a comparator and constants 64, 61, 73, 77, 20, 72 in the hexadecimal number system. With these numbers are encoded ASCII codes, namely "d", "a", "s", "w" (responsible for controlling the machine left-right and accelerating the obstacle), "space" - the beginning of the game and pause and "r" - restart respectively. The initial screen is displayed with a counter and its looping.

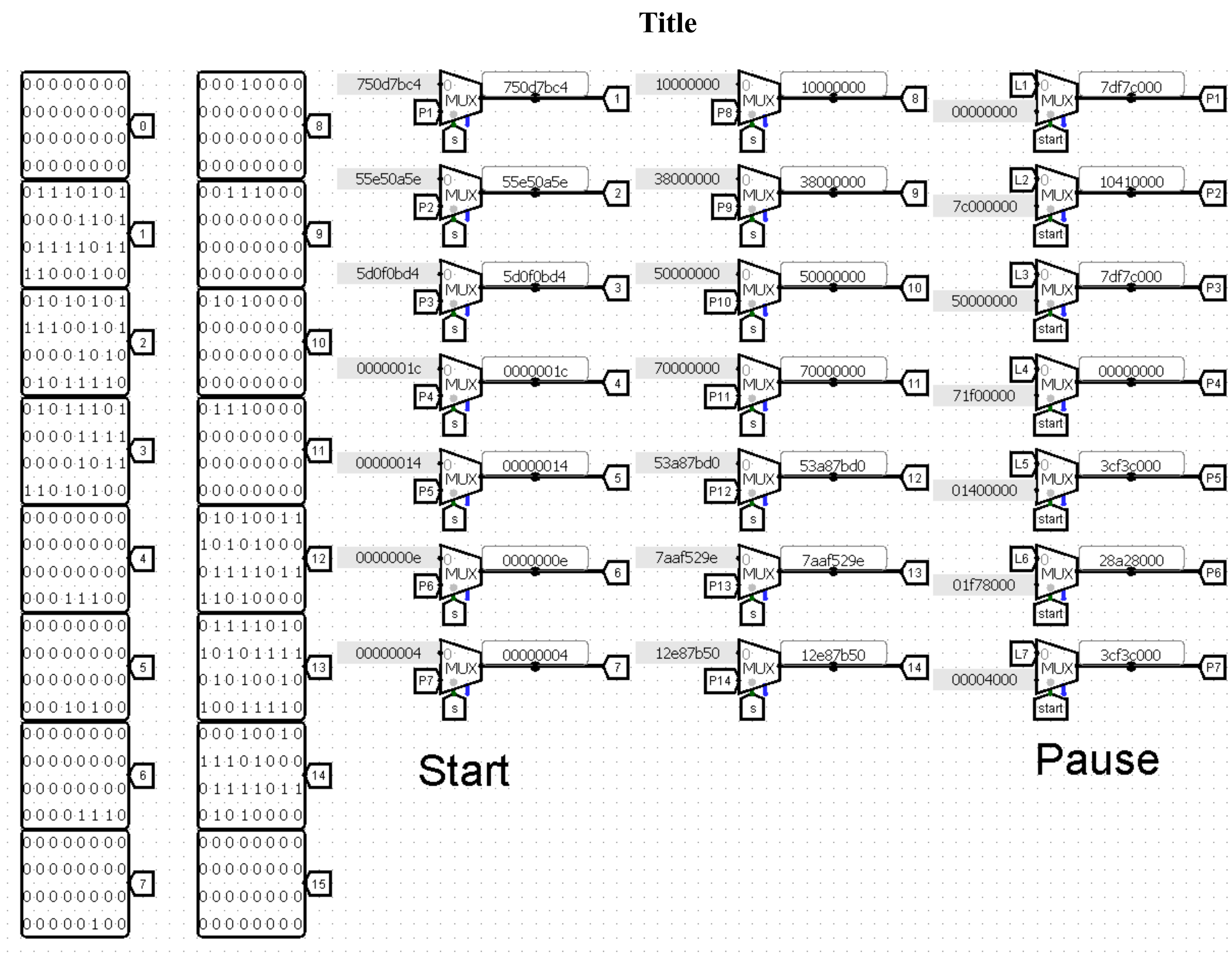


Figure 6. Part of the subcircuit "Title"

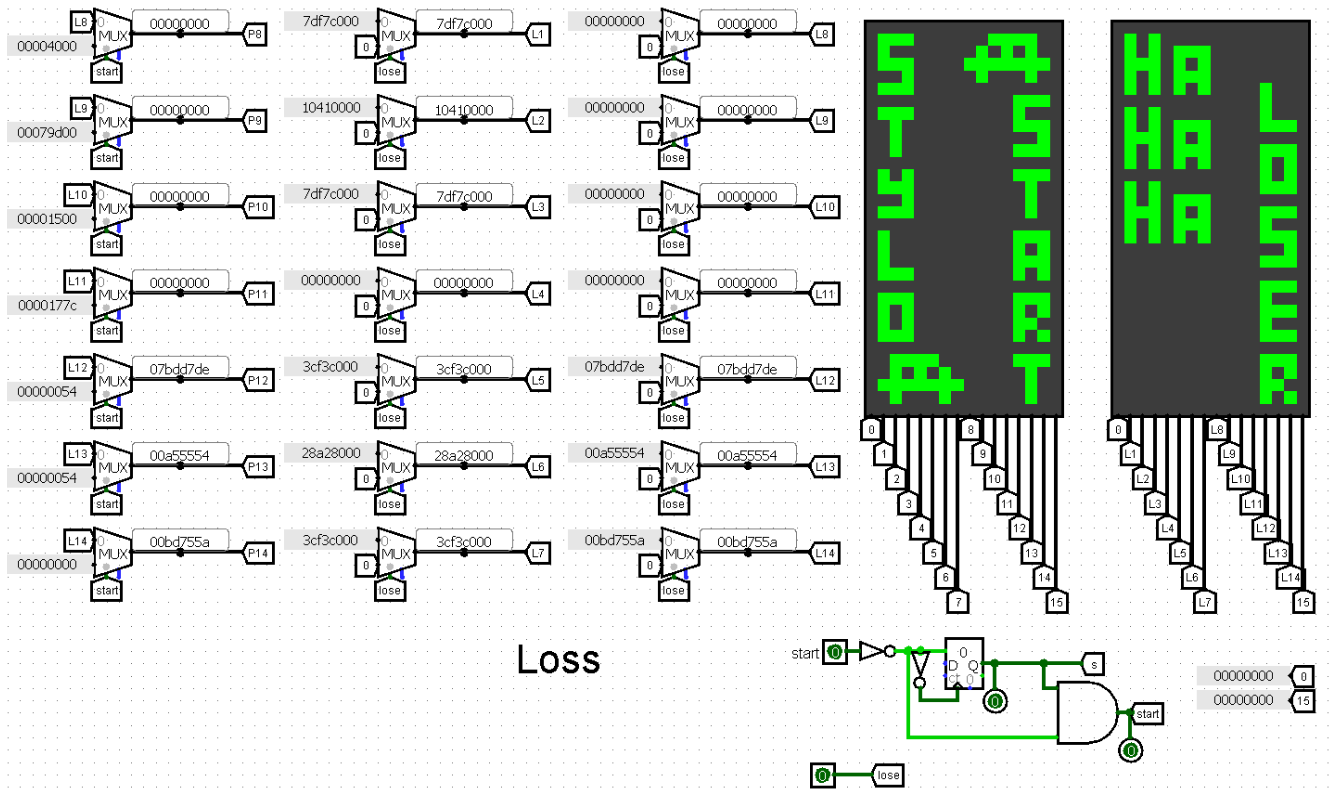


Figure 7. Part of the subcircuit "Title"

The values for this subcircuit are taken from the circuit "Main", namely labels "Start", "Lose", and the label "isStarted" is output in turn. If "Start" is zero, the Started screen is output to the "Main" circuit. This is realized with the help of a multiplexer. The multiplexers select the value of either "s" or "p1", "p2", ..., "p14". Since "s" initially has a low signal, the value of the constants responsible for displaying the start screen are output to the matrix. As soon as the signal "space" was sent from the keyboard, the counter "s" became equal to one and multiplexers take the value from "p1", "p2", ..., "p14", which, in turn, are taken from other multiplexers, which take either "0" or "L1", "L2", ..., "L14", if the player lost, there was a collision, then "Lose" = 0 and the screen is displayed with a message about the loss, if the game continues, nothing is printed. The start screen is displayed only once, before the game starts. If the "Space" button is pressed further, a screen with a pause message will be displayed. This is implemented using the 'AND' element, when the 's' counter has reached one and 'space' is pressed, then the 'start' label is sent to the multiplexers.

Score

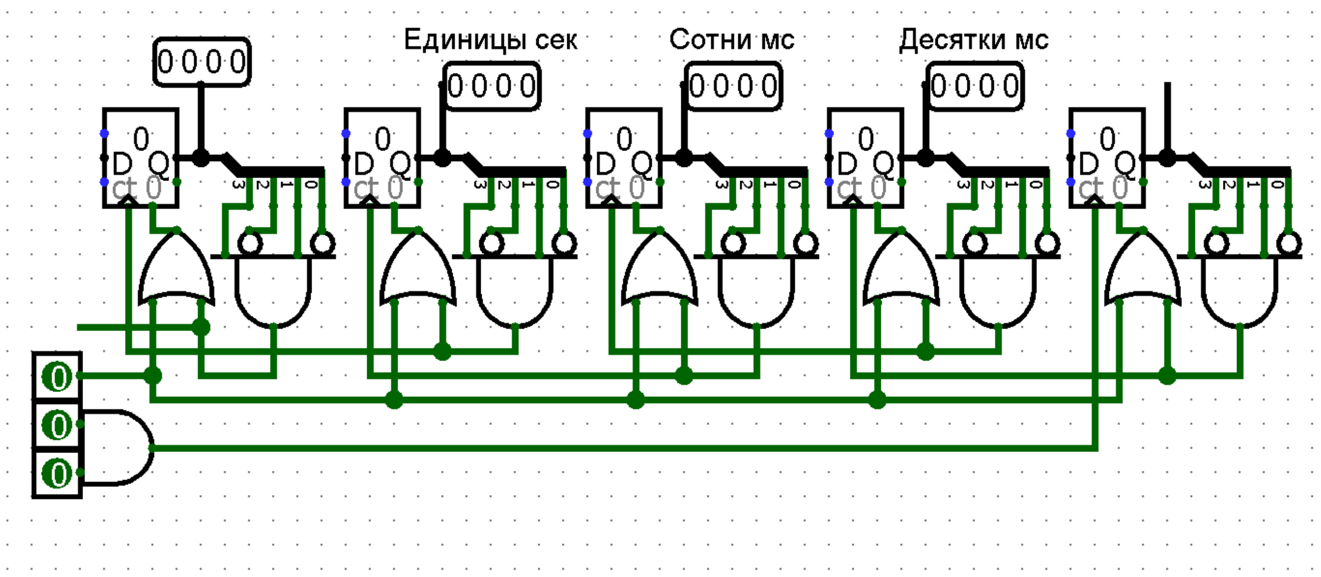


Figure 8. ”Score” subcircuit

On the main screen, above the matrix, the time counter is displayed, hexadecimal indicators display tenths of a second, as well as units, tens and hundreds of seconds.

Faster-slower

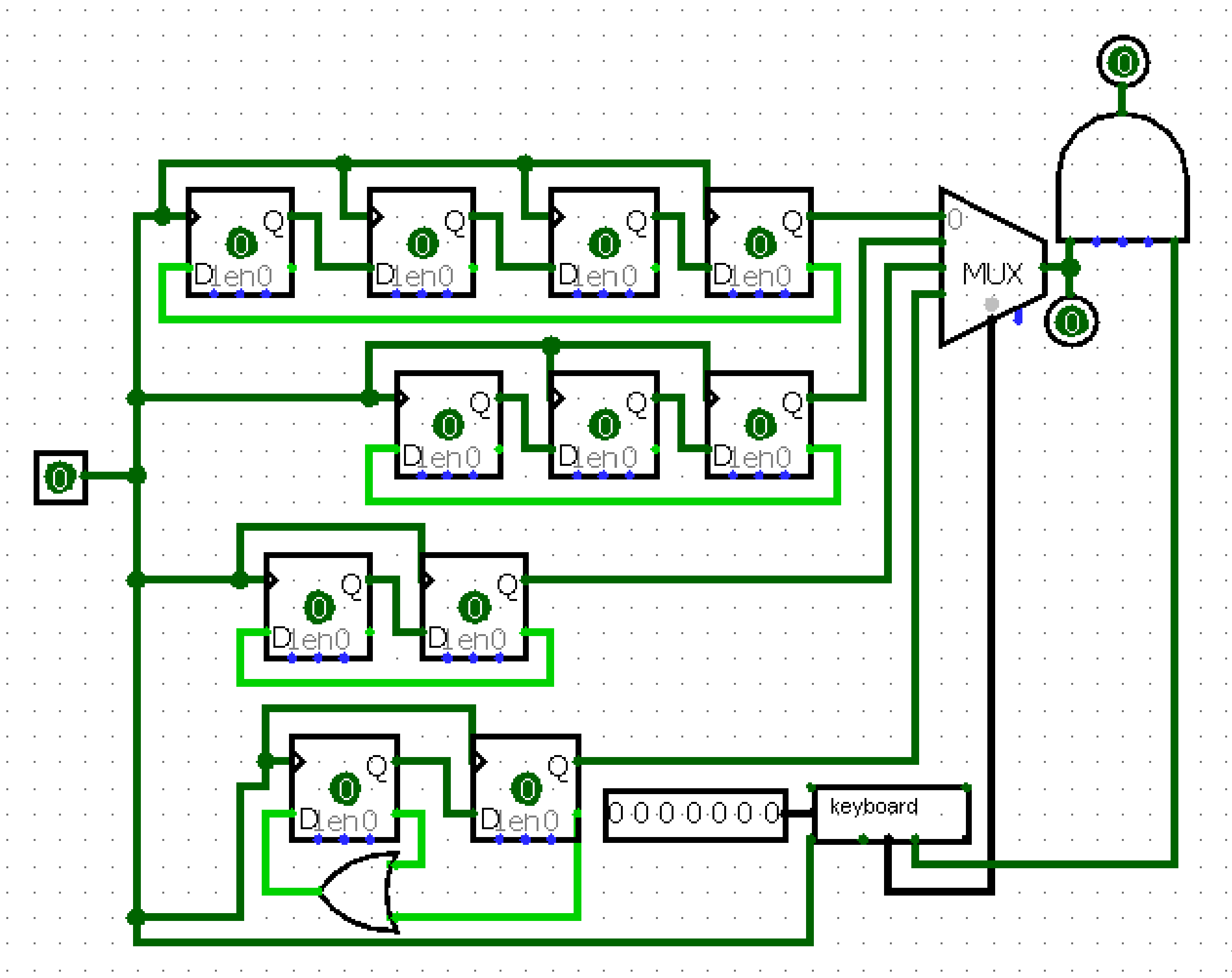


Рис.

Figure 9. Subcircuit “Faster-slower”

Pressing the button “w” changes the speed from 1 to 4 with the counter, and also the number of cycles, if “0” - then the obstacles appear every fourth cycle, if “1”, then every third cycle and so on.

Up-down

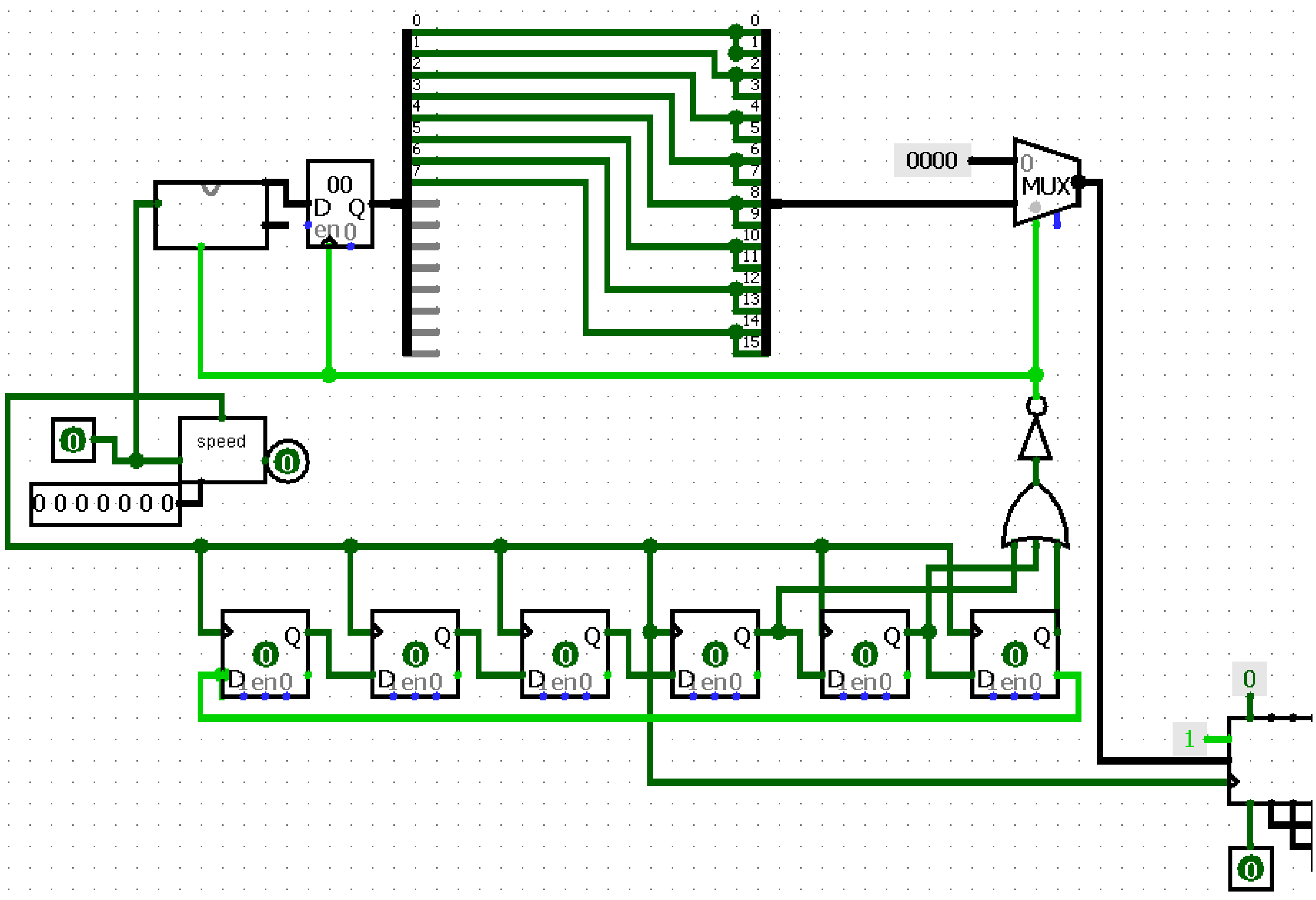


Figure 10. Part of the "Up-down" subcircuit

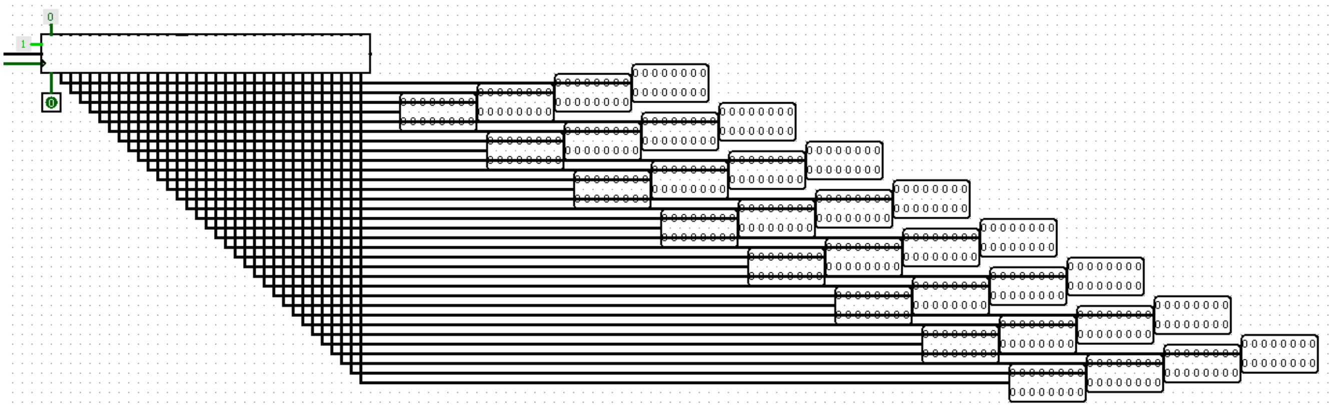


Figure 11. Part of the "Up-down" subcircuit

One of the main subcircuits of the project. It receives values from the keyboard and labels "tg" from the clock, "r" for restart and the subcircuit "Faster-slower" with the label "speed" with the generation of clock cycles to change the speed. The value is fed into the multiplexer from the "Harvard" subcircuit where random generation of numbers is done with the help of "CdM 8" and also with the help of splitters with "Check" subcircuits, which is responsible for the width of obstacles. The value from there is written to a register. This is to ensure that the value is not written every clock cycle, but every few clock cycles. D Flip-Flops are responsible for the same obstacle length. After all the manipulations, the value is sent to the shift register, which simply sends the value from the previous register to the next register. This is how the movement occurs along the vertical axis.

Right-left

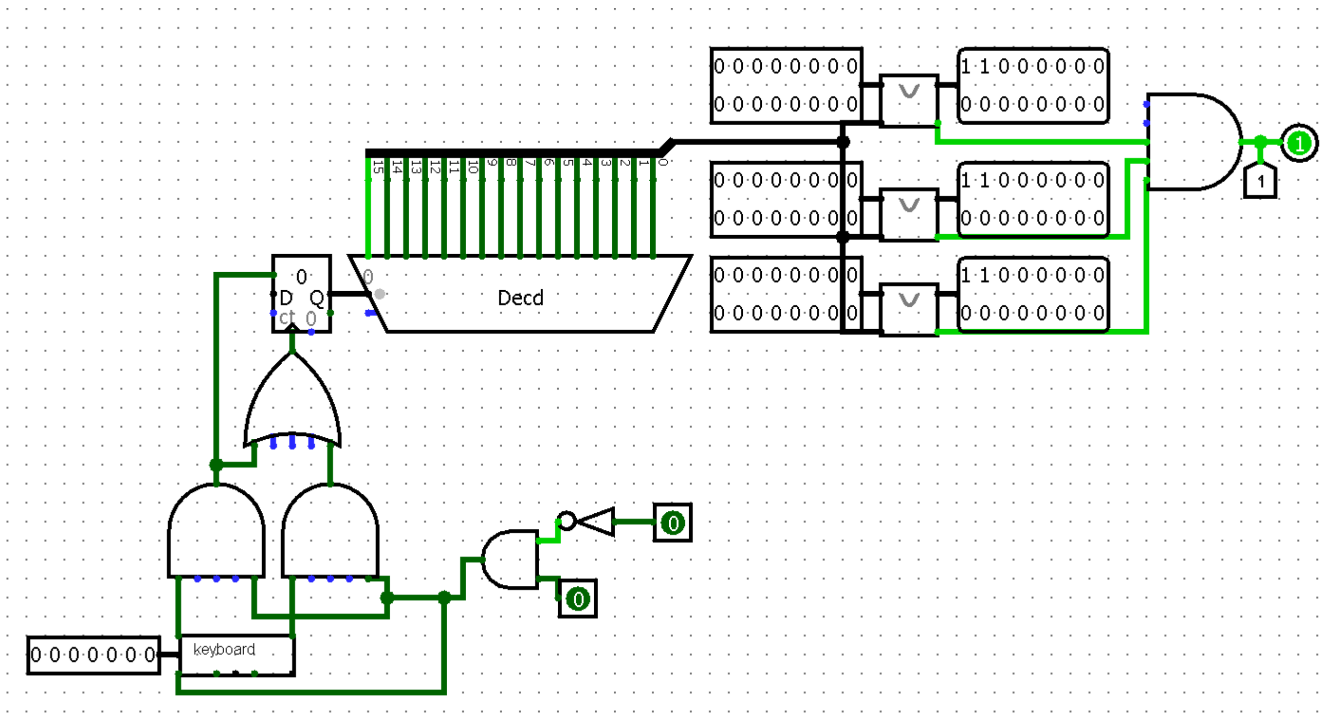


Figure 12. Right-left" subcircuit

"Keyboard" takes a value and outputs "1" if the "a" or "d" keys are pressed. This is to increase or decrease the counter, the decoder changes the position of the car, then this value is split into three lines. Each line has two values, one from the decoder and the other from the up-down circuit. These values are processed by the "Collision" subcircuit.

Collision

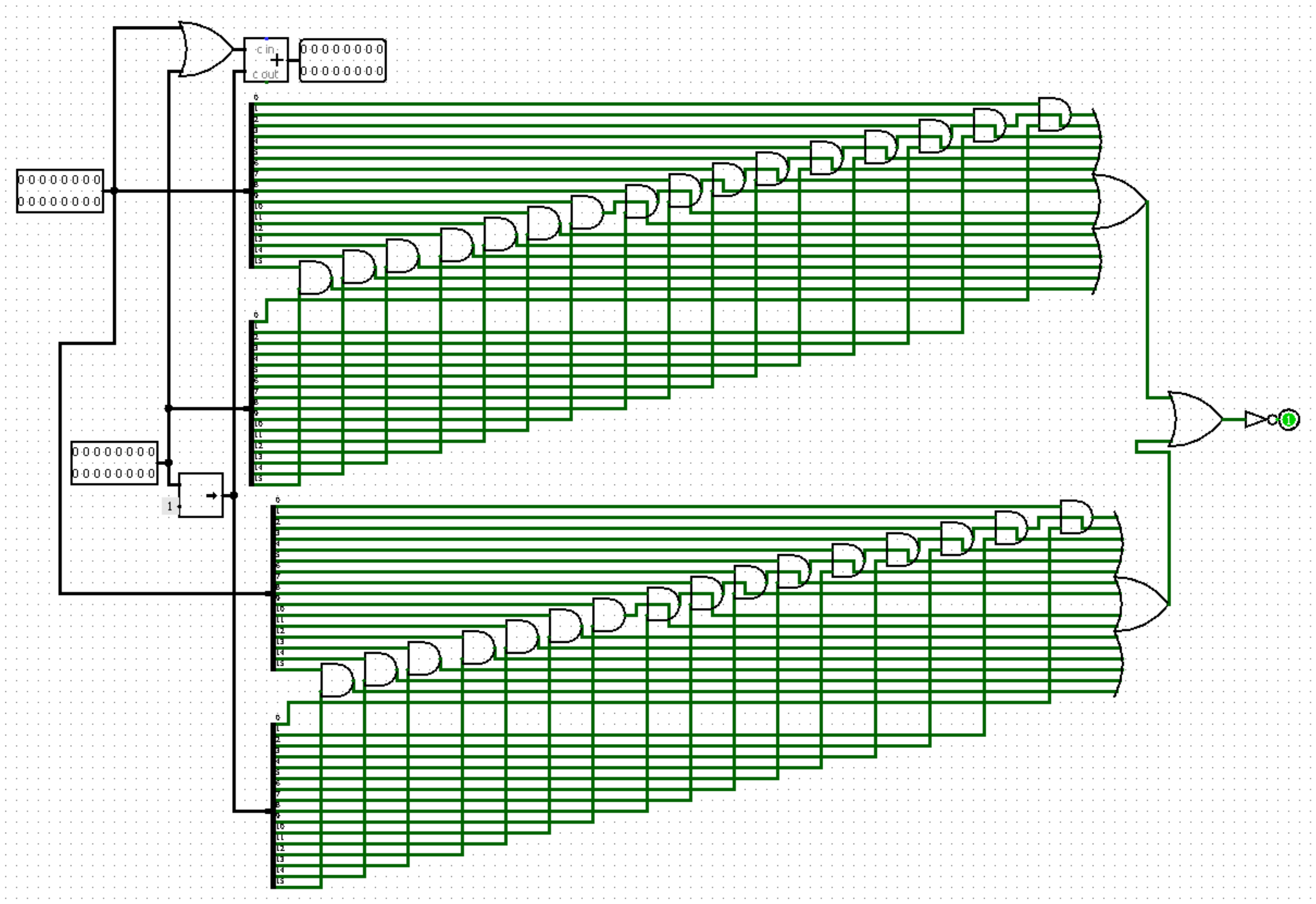


Figure 13. Subcircuit "Collision"

Two values are input: the position of the car and the position of the obstacles, if any bits match, then we send a loss message. As long as nothing is crossed, a high signal is output.

Tacktdel

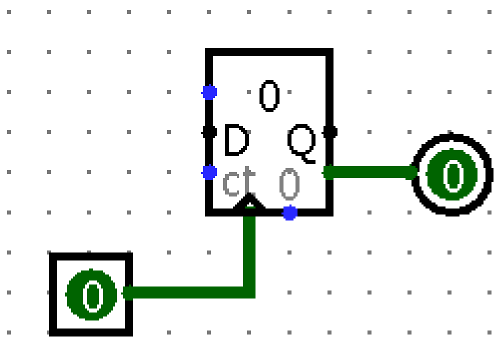


Fig. 14. Tacktdek subcircuit

Here the clock is divided 16 times by using the overflow.

Harvard

This subcircuit uses a "CdM 8" processor for connection. Harvard architecture is used.

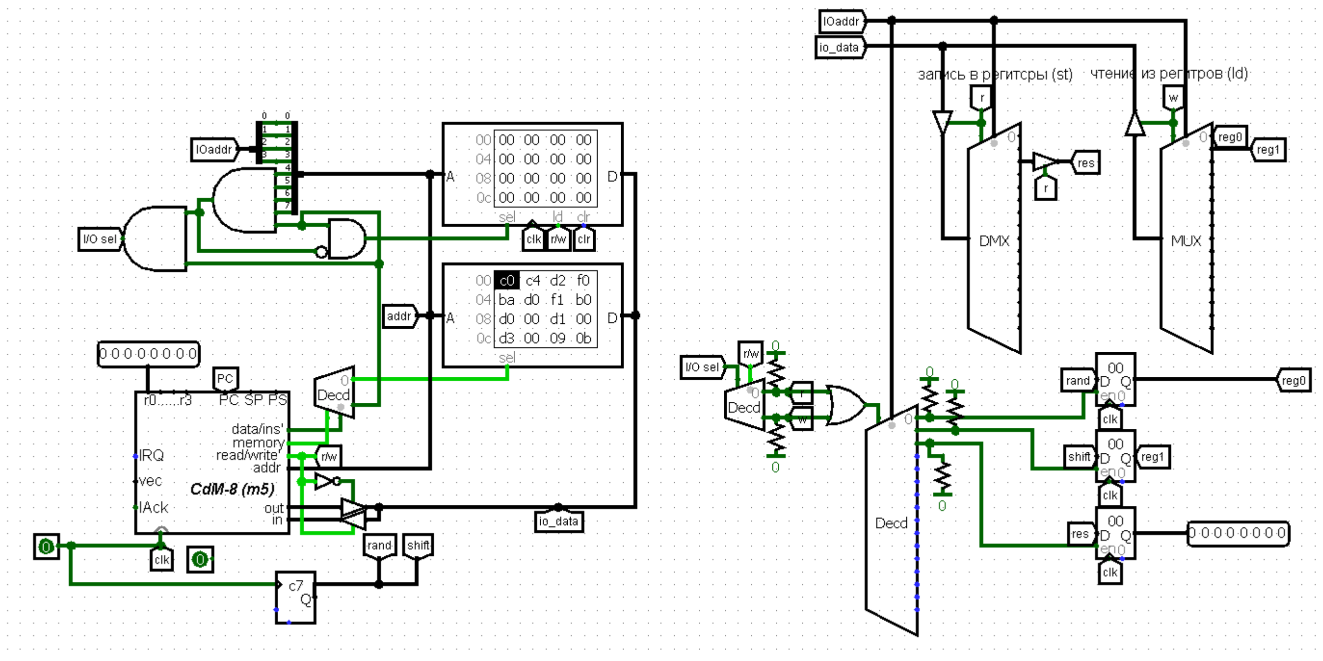
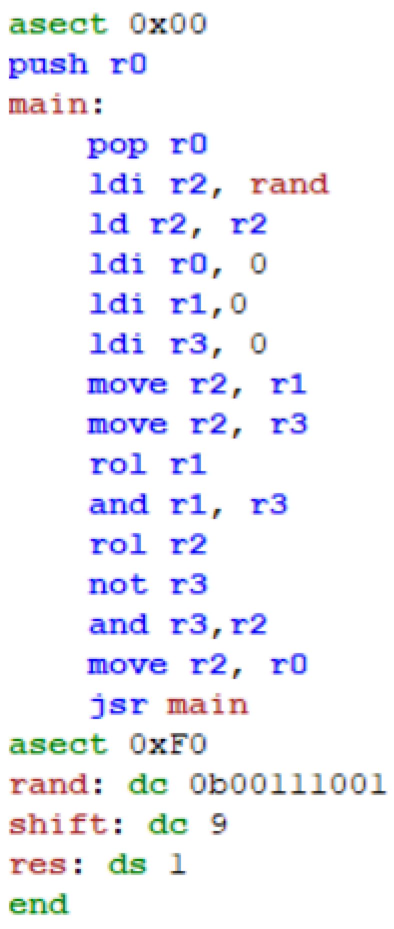


Figure 15: Harvard subcircuit

This circuit is used to generate obstacles using Cdm 8.

5. SOFTWARE



```
asect 0x00
push r0
main:
    pop r0
    ldi r2, rand
    ld r2, r2
    ldi r0, 0
    ldi r1,0
    ldi r3, 0
    move r2, r1
    move r2, r3
    rol r1
    and r1, r3
    rol r2
    not r3
    and r3,r2
    move r2, r0
    jsr main
asect 0xF0
rand: dc 0b00111001
shift: dc 9
res: ds 1
end
```

Figure 16. Screenshot of “Cdm 8” Assembly

This code gets a number, shifts it, performs XOR on the same number before the shift, inverts the result and performs the “AND” operation on the original number. In this way we get away from situations where a number has many units in a row.

6. CONCLUSION

During the development of this project all the set tasks were achieved. The game “Stylo” was created with the help of logic circuits and processor “CdM 8” executing the required program. All the requirements stated were successfully realized. When creating this project, knowledge in the field of circuitry and working with the processor, namely the Harvard architecture, writing documentation for the project and experience of working in a team were gained. A user manual is provided in appendix A.

7. SOURCES USED IN DEVELOPING

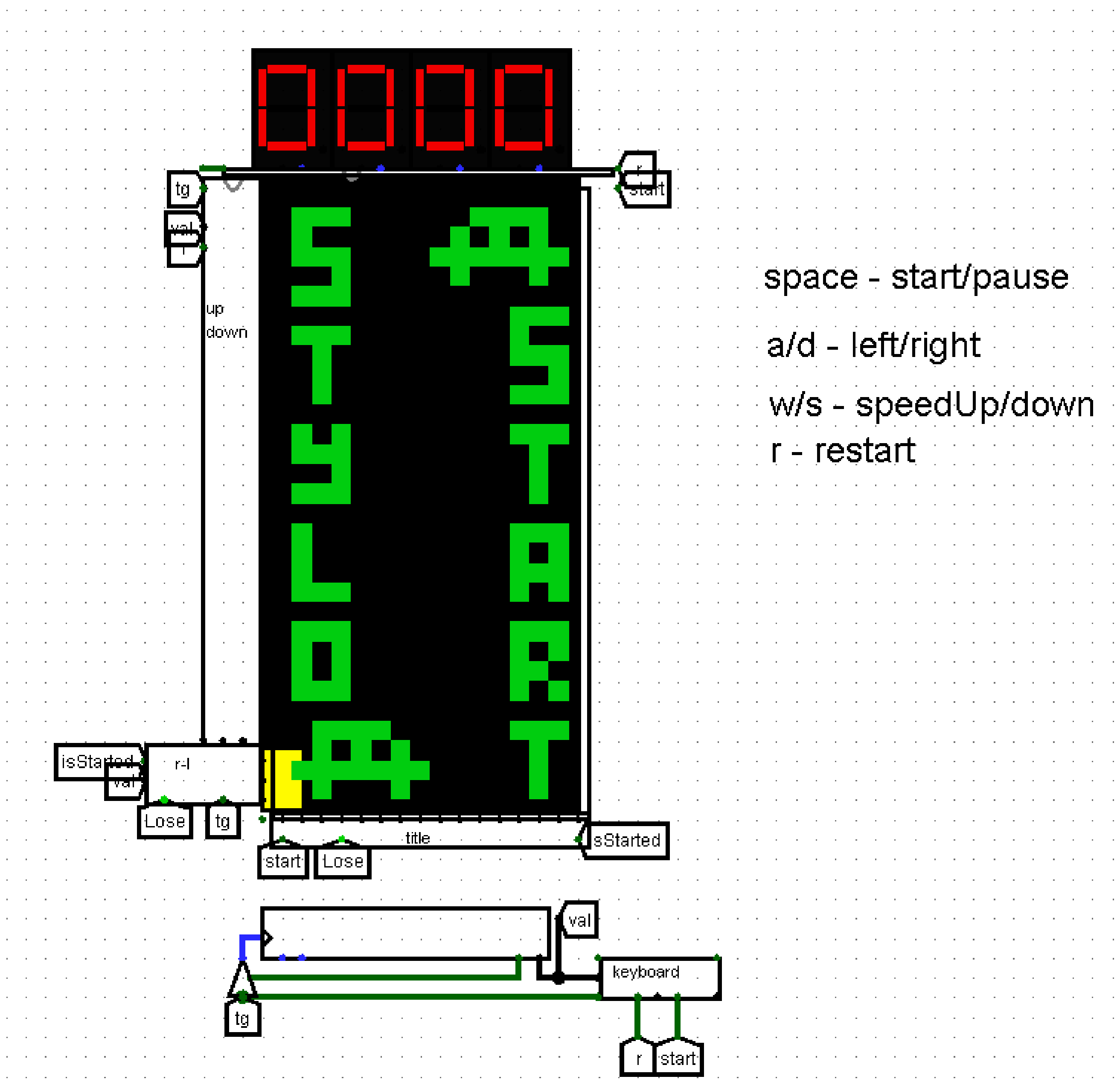
1. Computing platforms: учебник / A.Shafarenko, S.P.Hunt. – 2015.

APPENDIX

APPENDIX A

User manual

When first opening, the player is greeted by a start screen. To start the game, you need to press the "Space" key.



space - start/pause
a/d - left/right
w/s - speedUp/down
r - restart

Figure A.1 – the screenshot of the game's initial screen

After pressing the spacebar, the game started. It is required to control the car with the keys "a" and "d" in order not to crash into oncoming obstacles.

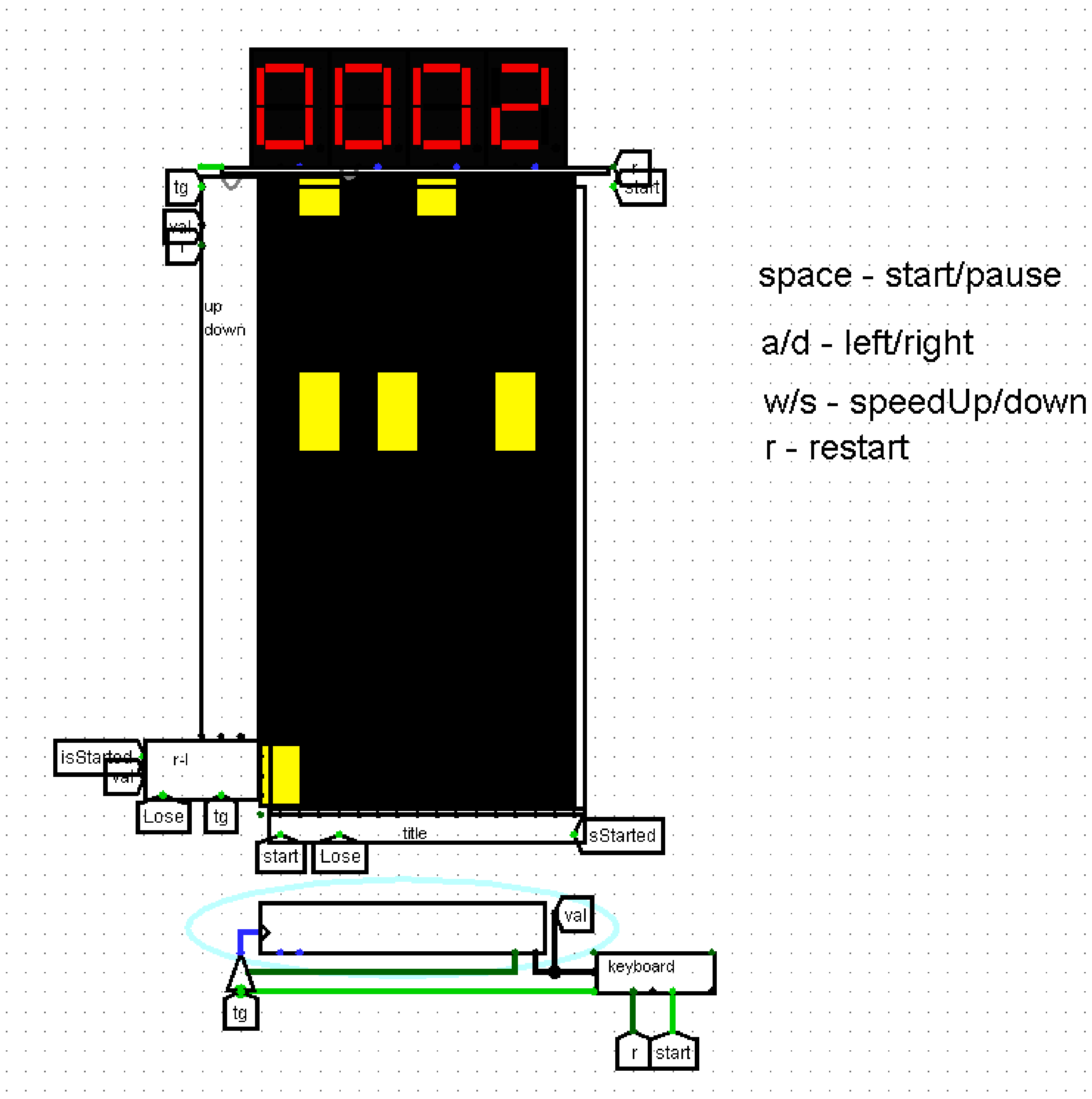


Figure A.2 – gameplay screenshot

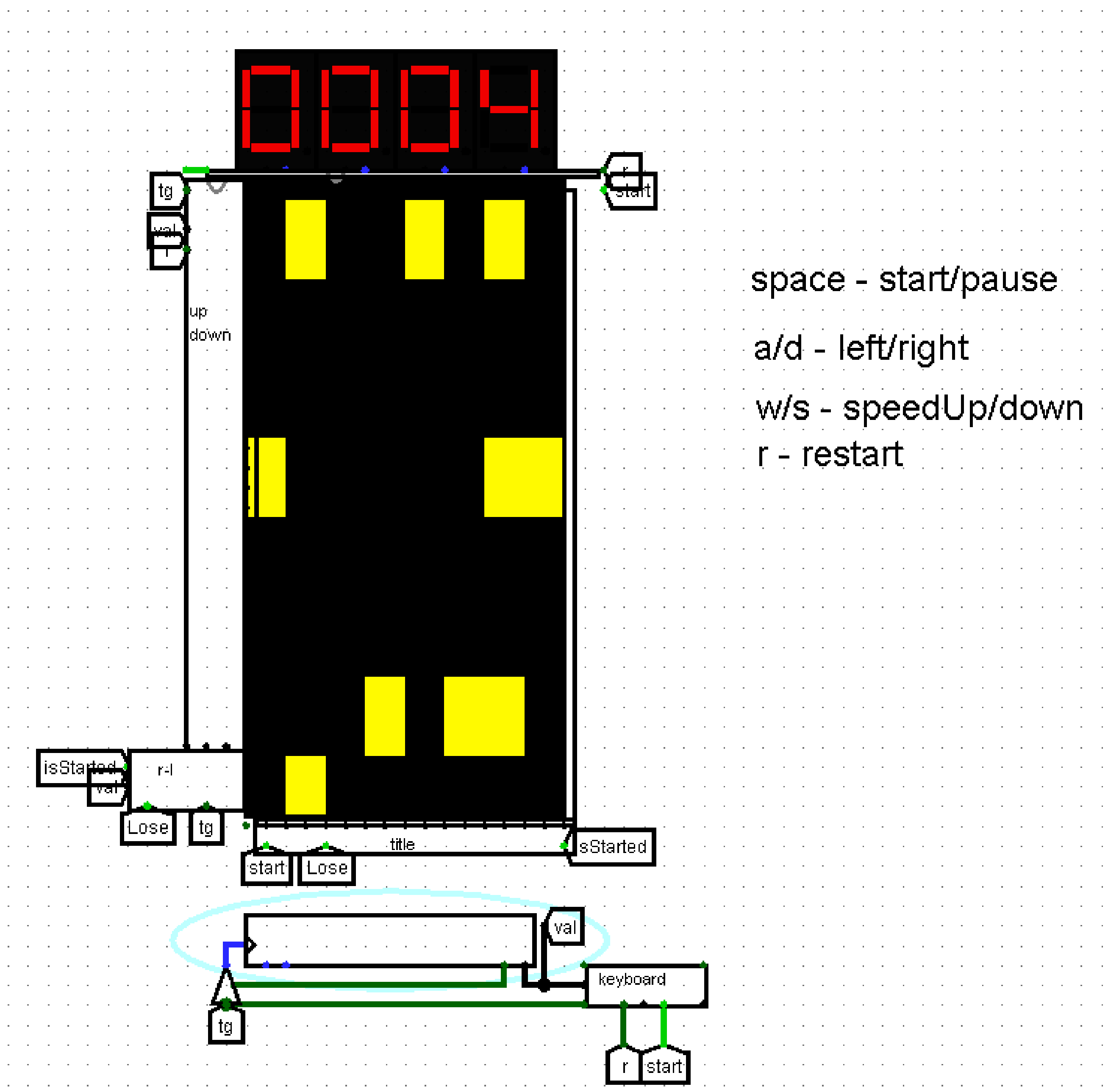


Figure A.3 - the car has moved and will be able to pass through the obstacles

If the player needs to leave or think about the best move, pressing the spacebar allows the player to pause the game.

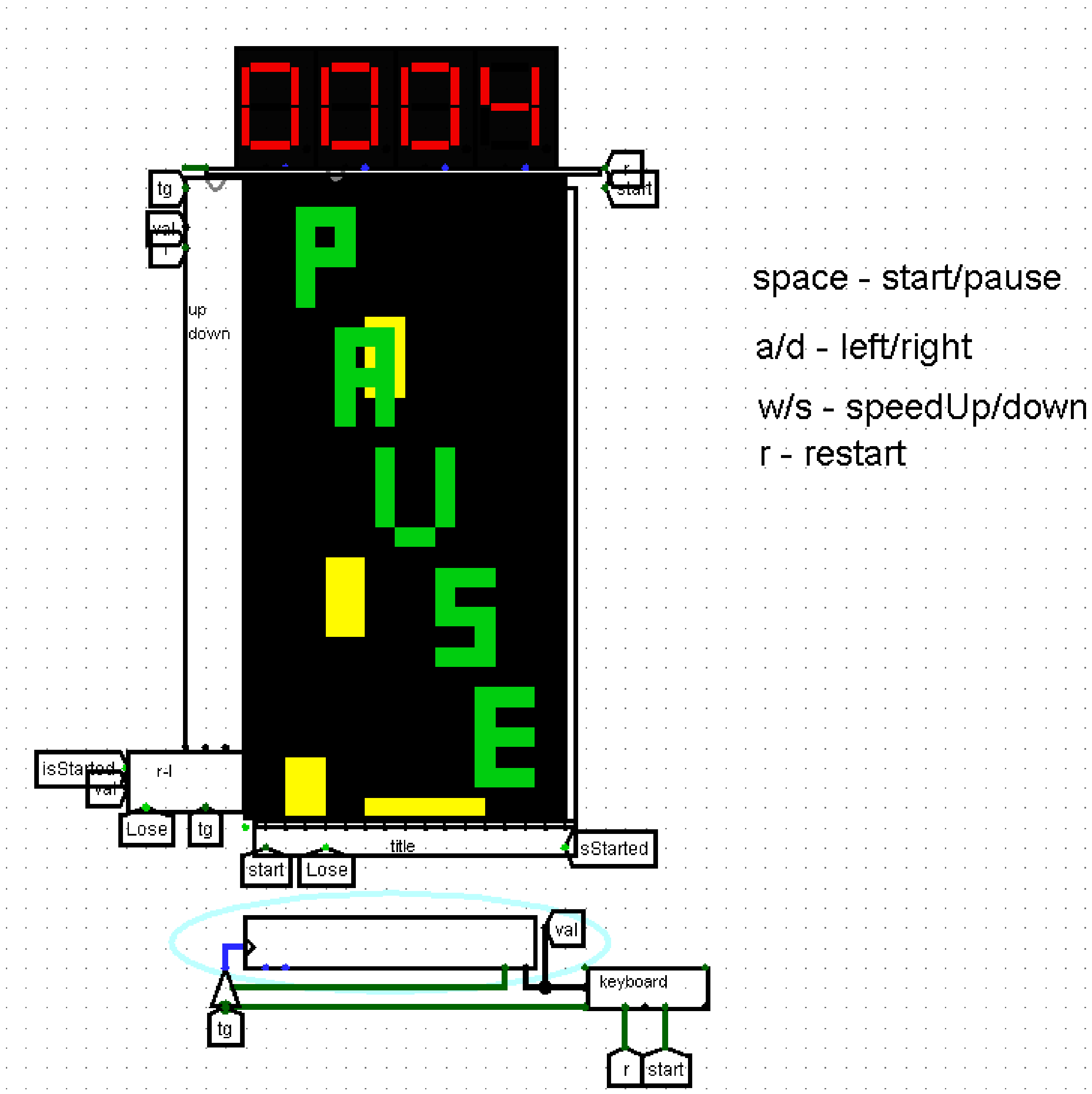
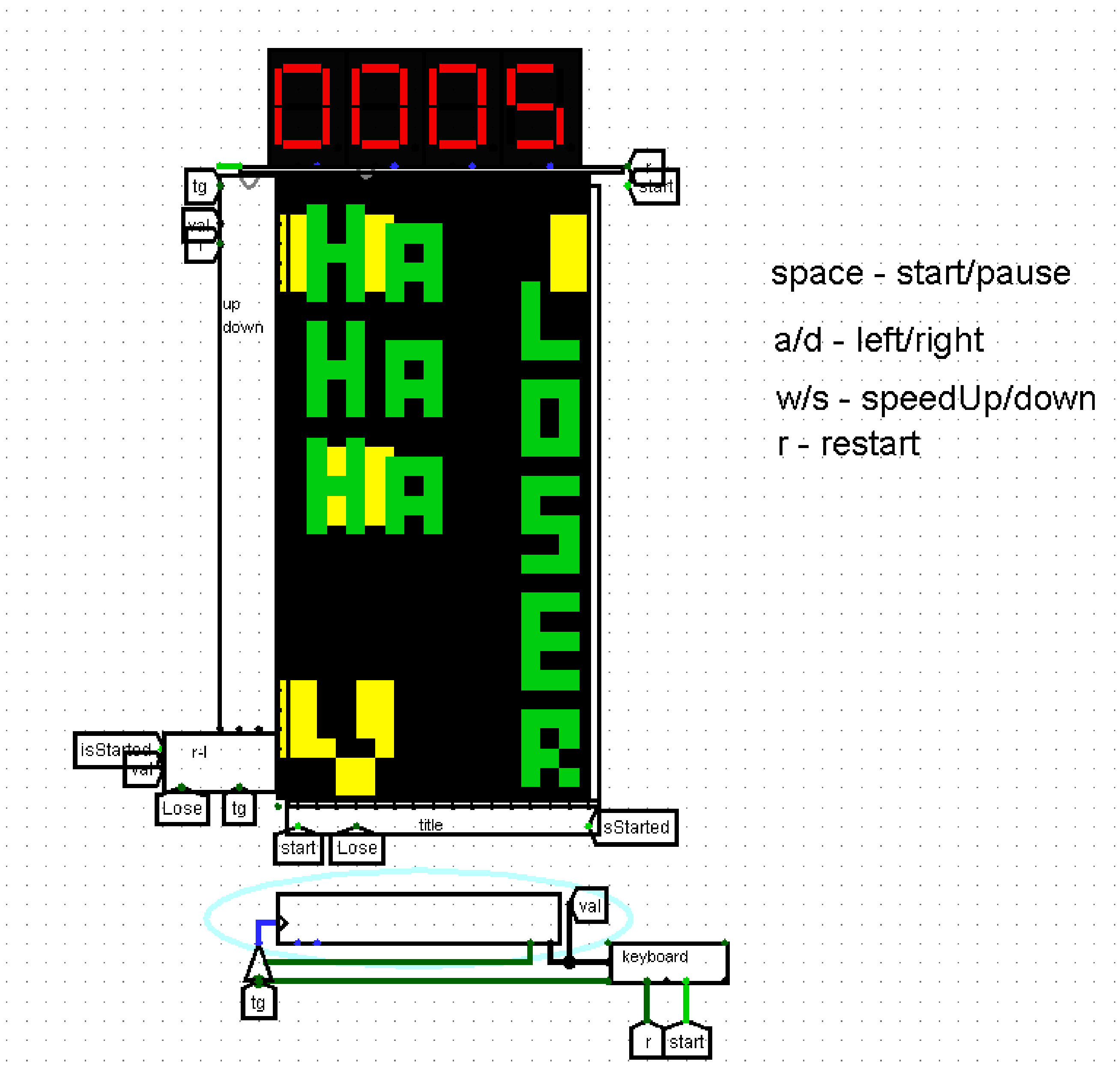


Figure A.4 – the screenshot of the game's pause screen

If the player could not manage to control the car and crashed into an obstacle, the game stops and a loss message is displayed.



space - start/pause
a/d - left/right
w/s - speedUp/down
r - restart

Figure A.5 – the screenshot of the game's lose screen

If the player thinks that the game is too slow and easy, he can increase the speed by pressing the “w” key or slow down if necessary, using the “s” key. If the game has failed from the beginning or the player has failed, pressing the “r” key will restart the game.