**Arduino Racing Game**

CONTENTS

1 Essence of the project 3

2 [Gameplay 3](#_Toc116889383)

[3 Components 3](#_Toc116889384)

[4 Building process 4](#_Toc116889384)

[5 Program 7](#_Toc116889384)

[6 Result 17](#_Toc116889384)

1. **Essence of the project**

To create an interesting and unique game for fun and improve your skills.

1. **Геймплей**

The player controls the car, obstacles are moving towards the car, which the player needs to go around. There are several different difficulty levels and the ability to view records. Gameplay video: <https://www.youtube.com/watch?v=zKWT0bUBfUk>

**3. Необходимые компоненты**

1. Arduino Nano.

2. Screen 16x2.

3. Joystick.

4. Battery compartment.

5. 8 connecting wires.

6. Phone tripod.

7. Anti-static bag.

8. 2 screws.

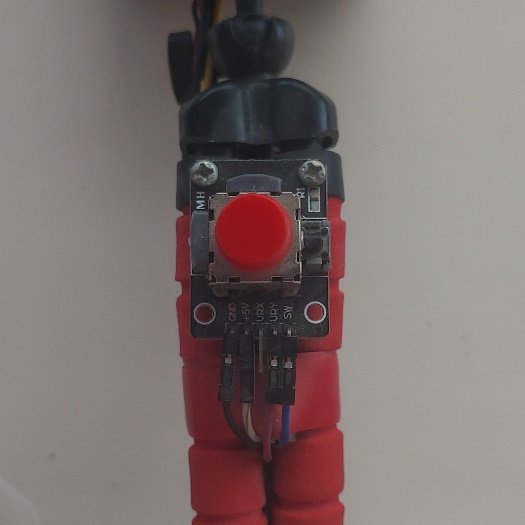
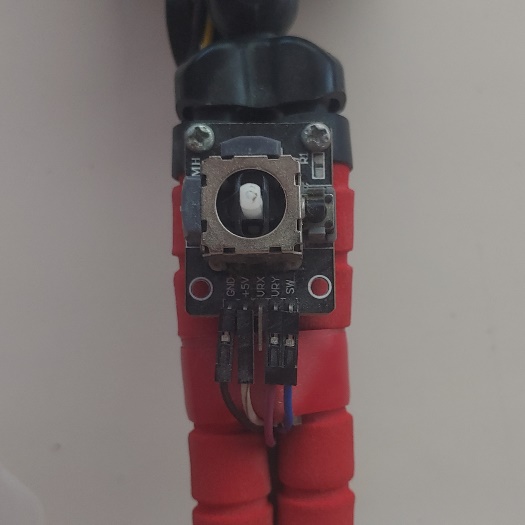
9. Sticky tape.

10. Glue gun with sticks

**4. Building process**

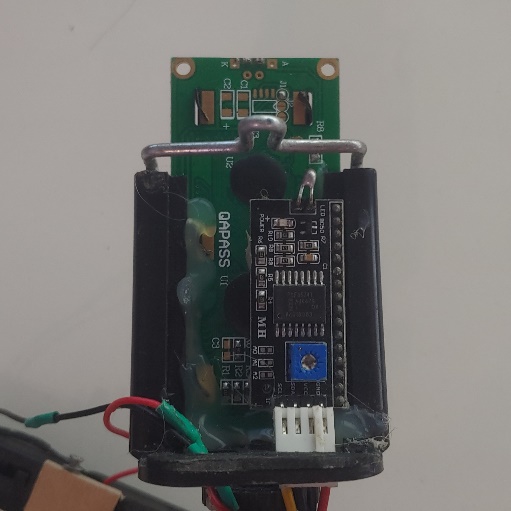
1. Joystick.

We take a tripod, and attach the joystick to it with the help of two screws or self-tapping screws. To make it easier, you can pre-drill holes in the plastic with a screwdriver.



1. Screen.

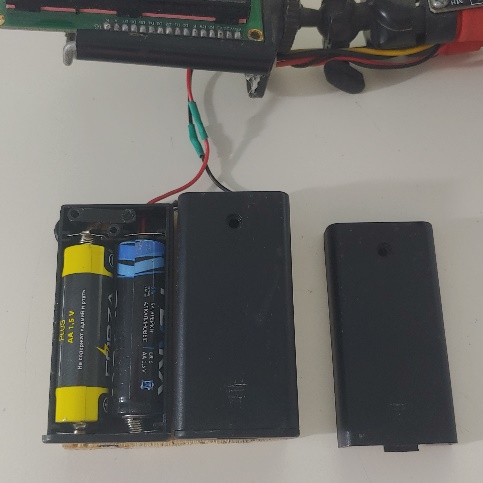
We unfold the clothespin of the tripod so that the back is looking at us. The screen here looks like a glove in size, except that it may be necessary to cut the plastic a little to fit the contacts of the i2c module. It is convenient to cut plastic with an engraver, but you can also use a saw, soldering iron, drill or screwdriver. We glue the screen on the back side with hot glue so that it does not fall off.



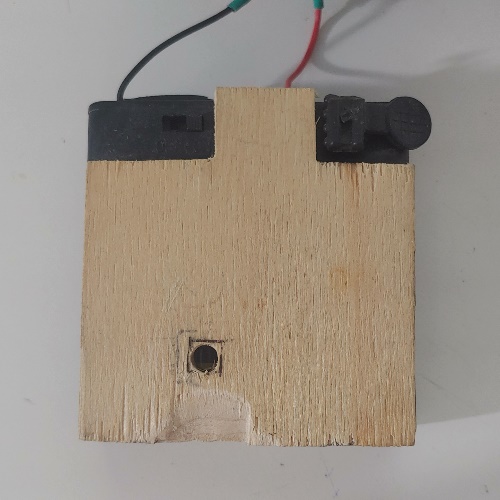
1. Battery compartment.

It is better to use a battery compartment with a built-in switch. If it is not there, it's okay, you can turn off the game by removing the battery. It is also better to use an AAA battery compartment rather than an AA one, as they are lighter, but they also need to be changed more often. You can also glue the battery compartment to the tripod body, but I did not see the need for this.

The store didn't have a 4-battery compartment, so I had to buy two 2-AA battery compartments and connect them. I glued them with universal glue to a piece of plywood of the right size and soldered the plus of one compartment to the minus of the other. Now they are like one compartment.

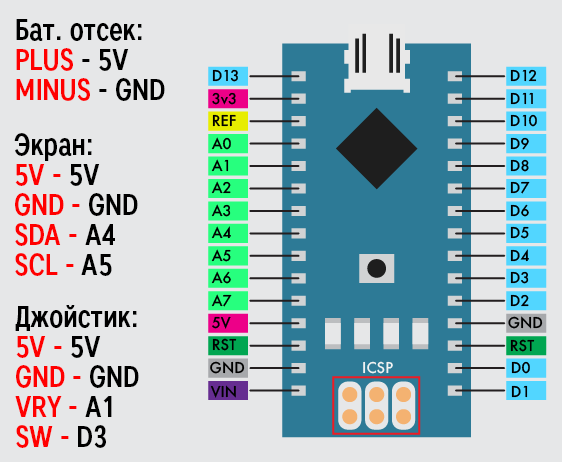


I also made two cutouts in the plywood for the switches. The tripod had a rubber “pimp” on top, pulling on which you can stretch the clothespin. We don't need to stretch the clothespin, so I found another use for it - I glued it to one of the switches to make it easier to turn the game on and off. I also made a hole in the plywood and the battery compartment so that you can adjust the brightness of the screen with a potentiometer without removing the battery compartment.



1. Arduino

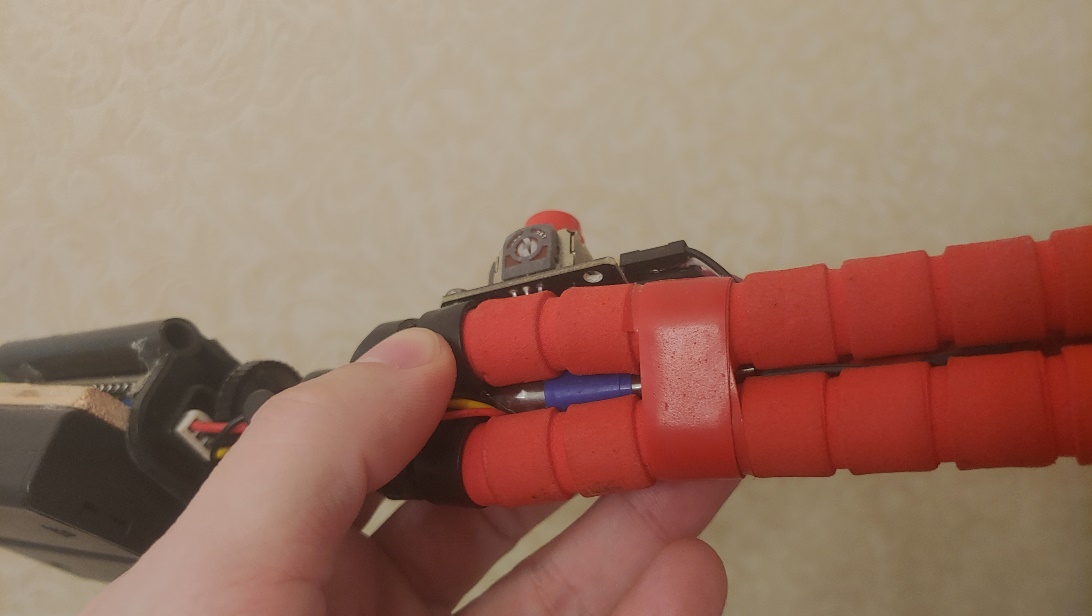
We take the Arduino and solder the wires to it according to the diagram. We flash the program in Arduino. (Бат. отсек – Battery, Экран – Screen, Джойстик - Joystick)



1. Finish

If everything works correctly, then we can finish the assembly and enjoy the result. We wrap the Arduino in a piece of an antistatic bag cut to fit its size and wrap it with electrical tape.

We place it between the legs of the tripod and wrap the legs with electrical tape so that they do not move apart. The legs come in different colors and it's best to use tape of the same color. The game is ready.



1. **Program**

Importing libraries and initializing global variables

1. #include <EEPROM.h> // Библиотека для работы с памятью
2. #include <Wire.h> // Библиотека для работы с i2c
3. #include <LiquidCrystal\_I2C.h> // Библиотека для работы с дисплеем
4. LiquidCrystal\_I2C lcd(0x27, 16, 2); // Инициализация дисплея
5. uint8\_t b[8]={0x1f, 0x19, 0x15, 0x13, 0x19, 0x15, 0x13, 0x1f}; // Графическая модель препятствия
6. uint8\_t Coords[16]; // В этом массиве хранится информация о местонахождении препятствий, 0 – нет препятствия, 1 – есть слева, 2 – есть справа
7. int last=0; // Последнее положение препятствия, переменная нужна чтобы предотвратить спавн препятствия сразу после следующего чтобы дать игроку возможность проехать между ними
8. int last2=0; // Эта переменная означает количество свободного места
9. unsigned long int points=0; // Счетчик очков игрока
10. int i,j,c; // В переменной с храниться информация о том где заспавнится препятствие, 0 – нигде, 1 – слева, 2 -
11. uint8\_t pos=0; // Текущая позиция машины, 0 – слева, 1 - справа
12. uint8\_t car1[8]={0x18,0x1e,0x5,0x19,0x19,0x5,0x1e,0x18}; // Графическая модель передней части
13. uint8\_t car2[8]={0xc,0xf,0x14,0x13,0x13,0x14,0xf,0xc}; // Графическая модель задней части машины
14. unsigned long int e; // Последнее сохраненное значение переменной millis()
15. int velocity, sectors; // Эти параметры зависят от сложности, velocity – скорость, sectors – минимальное расстояние между двумя препятствиями
16. int dif; // Сложность

Function that updates the position of the machine on the display

1. void printcar(){
2. if (pos==0){
3. lcd.setCursor(1,0);
4. lcd.write(1);
5. lcd.setCursor(0,0);
6. lcd.write(2);
7. lcd.setCursor(0,1);
8. lcd.print(" ");
9. lcd.setCursor(1,1);
10. lcd.print(" ");
11. }
12. else{
13. lcd.setCursor(1,1);
14. lcd.write(1);
15. lcd.setCursor(0,1);
16. lcd.write(2);
17. lcd.setCursor(0,0);
18. lcd.print(" ");
19. lcd.setCursor(1,0);
20. lcd.print(" ");
21. }
22. }

A function that is triggered when the position of the joystick is changed. It checks if there is an obstacle in the lane the player wants to change lanes to. If there is, it returns 0 and the game is over. If not, calls the printcar function.

A function that moves all objects 1 cell closer to the player

1. int turn(){
2. if (analogRead(A1)>640){
3. if (pos==0){
4. if (pos==0){
5. if ((Coords[0]==2)||(Coords[1]==2)){
6. return 0;
7. }
8. }
9. else{
10. if ((Coords[0]==1)||(Coords[1]==1)){
11. return 0;
12. }
13. }
14. pos=1;
15. printcar();
16. }
17. }
18. else if(analogRead(A1)<384){
19. if (pos==1){
20. if (pos==0){
21. if ((Coords[0]==2)||(Coords[1]==2)){
22. return 0;
23. }
24. }
25. else{
26. if ((Coords[0]==1)||(Coords[1]==1)){
27. return 0;
28. }
29. }
30. pos=0;
31. printcar();
32. }
33. }
34. return 1;
35. }

This function is called every tick and moves all obstacles by 1 cell

1. void moving(){
2. for (i=0;i<3;i++){
3. if (pos==0){
4. if (Coords[i]!=Coords[i+1]){
5. Coords[i]=Coords[i+1];
6. if (Coords[i]==1){
7. lcd.setCursor(i,1);
8. lcd.print(" ");
9. }
10. else if (Coords[i]==2){
11. lcd.setCursor(i,1);
12. lcd.write(0);
13. }
14. else{
15. lcd.setCursor(i,1);
16. lcd.print(" ");
17. }
18. }
20. }
21. else{
22. if (Coords[i]!=Coords[i+1]){
23. Coords[i]=Coords[i+1];
24. if (Coords[i]==1){
25. lcd.setCursor(i,0);
26. lcd.write(0);
27. }
28. else if (Coords[i]==2){
29. lcd.setCursor(i,0);
30. lcd.print(" ");
31. }
32. else{
33. lcd.setCursor(i,0);
34. lcd.print(" ");
35. }
36. }
37. }
38. }
39. for (i=2;i<15;i++){
40. if (Coords[i]!=Coords[i+1]){
41. Coords[i]=Coords[i+1];
42. if (Coords[i]==1){
43. lcd.setCursor(i,0);
44. lcd.write(0);
45. lcd.setCursor(i,1);
46. lcd.print(" ");
47. }
48. else if (Coords[i]==2){
49. lcd.setCursor(i,1);
50. lcd.write(0);
51. lcd.setCursor(i,0);
52. lcd.print(" ");
53. }
54. else{
55. lcd.setCursor(i,0);
56. lcd.print(" ");
57. lcd.setCursor(i,1);
58. lcd.print(" ");
59. }
60. }
61. }
62. }

If an obstacle spawns on the other side immediately after the previous obstacle, then a situation will result in which the player will be blocked from passing. In this situation, this function is called and moves all objects forward before displaying the next block.

1. int movingwithoutrand(){
2. for (j=0;j<sectors;j++){
3. if (pos+1==Coords[2]){
4. return 0;
5. }
6. moving();
7. Coords[15]=0;
8. lcd.setCursor(15,0);
9. lcd.print(" ");
10. lcd.setCursor(15,1);
11. lcd.print(" ");
12. if (turn()==0){
13. return 0;
14. }
15. points++;
16. e=millis();
17. while(millis()-e<velocity){
18. if (turn()==0){
19. return 0;
20. }
21. }
23. }
24. return 1;
25. }

The function of changing the difficulty level in the menu for displaying the maximum points scored

1. void dispscore(){
2. byte val1 = EEPROM.read((dif-1)\*4+1);
3. byte val2 = EEPROM.read((dif-1)\*4+2);
4. byte val3 = EEPROM.read((dif-1)\*4+3);
5. byte val4 = EEPROM.read((dif-1)\*4+4);
6. unsigned long int val = val1\*pow(2,24)+val2\*pow(2,16)+val3\*(pow(2,8))+val4;
7. switch (dif){
8. case 1:
9. lcd.clear();
10. lcd.setCursor(0,0);
11. lcd.print(" TripMode > ");
12. break;
13. case 2:
14. lcd.clear();
15. lcd.setCursor(0,0);
16. lcd.print(" < Easy > ");
17. break;
18. case 3:
19. lcd.clear();
20. lcd.setCursor(0,0);
21. lcd.print(" < Normal > ");
22. break;
23. case 4:
24. lcd.clear();
25. lcd.setCursor(0,0);
26. lcd.print(" < Hard > ");
27. break;
28. case 5:
29. lcd.clear();
30. lcd.setCursor(0,0);
31. lcd.print("< ImpossibleFast");
32. break;
33. }

The interface function of the maximum points view menu

1. void maxscores(){ // "difficulty" procedure calls it when user picks up "Max Score" option on the menu.
2. dif=3;
3. lcd.clear();
4. lcd.setCursor(3,0);
5. lcd.print("< Normal >");
6. lcd.setCursor(0,1);
7. byte val1 = EEPROM.read((dif-1)\*4+1);
8. byte val2 = EEPROM.read((dif-1)\*4+2);
9. byte val3 = EEPROM.read((dif-1)\*4+3);
10. byte val4 = EEPROM.read((dif-1)\*4+4);
11. unsigned long int val = val1\*pow(2,24)+val2\*pow(2,16)+val3\*(pow(2,8))+val4;
12. lcd.print(val);
13. while (1){
14. if ((analogRead(A1)>640) && (dif<5)){
15. dif++;
16. dispscore();
17. while(analogRead(A1)>640){
18. delay(5);
19. }
20. }
21. else if ((analogRead(A1)<384) && (dif>1)){
22. dif--;
23. dispscore();
24. while(analogRead(A1)<384){
25. delay(5);
26. }
27. }
28. else if (digitalRead(3)==0){
29. while(digitalRead(3)==0){
30. delay(5);
31. }
32. break;
33. }
35. }
36. difficulty();
37. }

The function changes the name of the difficulty level on the screen and sets the parameters for the selected level

1. void dispdif(){
2. switch (dif){
3. case 0:
4. lcd.setCursor(0,1);
5. lcd.print(" Max Scores > ");
6. break;
7. case 1:
8. lcd.setCursor(0,1);
9. lcd.print(" < TripMode > ");
10. velocity=200;
11. sectors=3;
12. break;
13. case 2:
14. lcd.setCursor(0,1);
15. lcd.print(" < Easy > ");
16. velocity=150;
17. sectors=3;
18. break;
19. case 3:
20. lcd.setCursor(0,1);
21. lcd.print(" < Normal > ");
22. velocity=100;
23. sectors=3;
24. break;
25. case 4:
26. lcd.setCursor(0,1);
27. lcd.print(" < Hard > ");
28. velocity=150;
29. sectors=2;
30. break;
31. case 5:
32. lcd.setCursor(0,1);
33. lcd.print("< ImpossibleFast");
34. velocity=100;
35. sectors=2;
36. break;
37. }
38. }

Difficulty selection menu interface function

1. void difficulty(){ // It is called before race is started to let user set the difficulty
2. dif=3;
3. velocity=100;
4. sectors=3;
5. lcd.clear();
6. lcd.setCursor(0,0);
7. lcd.print("Difficulty");
8. lcd.setCursor(3,1);
9. lcd.print("< Normal >");
10. while (1){
11. if ((analogRead(A1)>640) && (dif<5)){
12. dif++;
13. dispdif();
14. while(analogRead(A1)>640){
15. delay(5);
16. }
17. }
18. else if ((analogRead(A1)<384) && (dif>0)){
19. dif--;
20. dispdif();
21. while(analogRead(A1)<384){
22. delay(5);
23. }
24. }
25. else if (digitalRead(3)==0){
26. while(digitalRead(3)==0){
27. delay(5);
28. }
29. if (dif>0){
30. break;
31. }
32. else{
33. maxscores();
34. break;
35. }
36. }
38. }
39. }

Setup function

1. void setup() {
2. lcd.begin();
3. lcd.backlight();
4. Serial.begin(9600);
5. lcd.createChar(0, b);
6. lcd.createChar(1,car1);
7. lcd.createChar(2,car2);
8. //lcd.createChar(1, c);
10. lcd.setCursor(0,0);
11. lcd.print("Racing game");
12. lcd.setCursor(0,1);
13. lcd.print("Press the button");
14. pinMode(3,INPUT\_PULLUP);
15. }

Loop function. It runs in a circle and controls the entire process of the game

1. void loop() {
2. if (digitalRead(3)==0){
3. while (1){
4. if (digitalRead(3)==1) break;
5. delay(5);
6. }
7. points=0;
8. difficulty();
9. randomSeed(millis() % 1000);
10. lcd.clear();
11. for (i=0;i<15;i++){
12. Coords[i]=0;
13. }
14. printcar();
15. while(1){
17. c=random(3);
18. if (c!=0 & c!=last & last!=0 & last2<4){
19. if (movingwithoutrand()==0){
20. break;
21. }
22. }
23. if (pos+1==Coords[2]){
24. break;
25. }
26. moving();
27. Coords[15]=c;
28. if (Coords[15]==1){
29. lcd.setCursor(15,0);
30. lcd.write(0);
31. lcd.setCursor(15,1);
32. lcd.print(" ");
33. }
34. else if (Coords[15]==2){
35. lcd.setCursor(15,1);
36. lcd.write(0);
37. lcd.setCursor(15,0);
38. lcd.print(" ");
39. }
40. else{
41. lcd.setCursor(15,0);
42. lcd.print(" ");
43. lcd.setCursor(15,1);
44. lcd.print(" ");
45. }
46. if (Coords[15]!=0){
47. last=Coords[15];
48. last2=0;
49. }
50. else{
51. last2++;
52. }
53. if (turn()==0){
54. break;
55. }
56. points++;
57. e=millis();
58. while(millis()-e<velocity){
59. if (turn()==0){
60. break;
61. }
62. }
63. }
64. lcd.clear();
65. lcd.setCursor(0,0);
66. lcd.print("Game over");
67. if (points<10000){
68. lcd.setCursor(0,1);
69. lcd.print("Your points:");
70. lcd.setCursor(12,1);
71. lcd.print(points);
72. }
73. else if (points<1000000){
74. lcd.setCursor(0,1);
75. lcd.print("Ur points:");
76. lcd.setCursor(10,1);
77. lcd.print(points);
78. }
79. else if (points<1000000000){
80. lcd.setCursor(0,1);
81. lcd.print("Points:");
82. lcd.setCursor(7,1);
83. lcd.print(points);
84. }
85. else{
86. lcd.setCursor(0,1);
87. lcd.print("Find a girlfriend");
88. }
89. byte val1 = EEPROM.read((dif-1)\*4+1);
90. byte val2 = EEPROM.read((dif-1)\*4+2);
91. byte val3 = EEPROM.read((dif-1)\*4+3);
92. byte val4 = EEPROM.read((dif-1)\*4+4);
93. unsigned long int power = 2;
94. unsigned long int val = val1\*pow(2,24)+val2\*pow(2,16)+val3\*(pow(2,8))+val4;
95. if (points>val){
96. if (points>=1000000000){
97. points=999999999;
98. }
99. val1=0;
100. val2=0;
101. val3=0;
102. val4=0;
103. val4=val4+bitRead(points, 0);
104. for (i=1;i<8;i++){
105. val4=val4+power\*bitRead(points, i);
106. power\*=2;
107. }
108. val3=val3+bitRead(points, 8);
109. power=2;
110. for (i=9;i<16;i++){
111. val3=val3+power\*bitRead(points, i);
112. power\*=2;
113. }
114. power=2;
115. val2=val2+bitRead(points, 16);
116. for (i=17;i<24;i++){
117. val2=val2+power\*bitRead(points, i);
118. power\*=2;
119. }
120. power=2;
121. val1=val1+bitRead(points, 24);
122. for (i=25;i<29;i++){
123. val1=val1+power\*bitRead(points, i);
124. power\*=2;
125. }
126. EEPROM.write((dif-1)\*4+1, val1);
127. EEPROM.write((dif-1)\*4+2, val2);
128. EEPROM.write((dif-1)\*4+3, val3);
129. EEPROM.write((dif-1)\*4+4, val4);
130. }
131. while (1){
132. if (digitalRead(3)==0){
133. while(digitalRead(3)==0){
134. delay(5);
135. }
136. break;
137. }
138. }
139. lcd.setCursor(0,0);
140. lcd.print("Racing game");
141. lcd.setCursor(0,1);
142. lcd.print("Press the button");
144. }
145. }

**6. Result**

You've got the experience of interacting with the Arduino environment and an interesting, unique, homemade game. If something does not work, check the performance of each module separately and the quality of the soldering. You can check out my result here: [https://github.com/Mike-Kuznetsov/Arduino-Racing-Game/](https://github.com/Mike-Kuznetsov/Arduino-Racing-Game/%20)