



Nombre del estudiante: RB20

Asignación:

notas:

Nombre del Proyecto: RB20

Tipo de proyecto: C++

Date: Mon Jun 10 2024

```

1  #pragma region VEXcode Generated Robot Configuration
2  // Make sure all required headers are included.
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <stdbool.h>
6  #include <math.h>
7  #include <string.h>
8
9
10 #include "vex.h"
11
12 using namespace vex;
13
14 // Brain should be defined by default
15 brain Brain;
16
17
18 // START IQ MACROS
19 #define waitUntil(condition) \
20 do {
21     wait(5, msec); \
22     while (!(condition))
23
24 #define repeat(iterations) \
25 for (int iterator = 0; iterator < iterations; iterator++)
26 // END IQ MACROS
27
28
29 // Robot configuration code.
30 inertial BrainInertial = inertial();
31 distance SI = distance(PORT5);
32 distance SC = distance(PORT3);
33 distance SD = distance(PORT11);
34 motor LeftDriveSmart = motor(PORT1, 1, false);
35 motor RightDriveSmart = motor(PORT12, 1, true);
36 drivetrain Drivetrain = drivetrain(LeftDriveSmart, RightDriveSmart, 200, 173, 76, mm
, 1);
37 motor VOLANTE = motor(PORT6, true);
38 colorsensor COLOR = colorsensor(PORT4);
39
40 #pragma endregion VEXcode Generated Robot Configuration
41
42 // Include the IQ Library
43 #include "vex.h"
44
45 // Allows for easier use of the VEX Library
46 using namespace vex;
47
48 float myVariable, giro, grados;
49
50 int mathRandomInt(float a, float b) {
51     if (a > b) {
52         // Swap a and b to ensure a is smaller.

```

```

53 float c = a;
54 a = b;
55 b = c;
56     }
57 int tmpA = static_cast<int>(a);
58 int tmpB = static_cast<int>(b);
59 int r = tmpA + rand() / (RAND_MAX / (tmpB - tmpA + 1));
60 return r;
61 }
62
63 // "when started" hat block
64 int whenStarted1() {
65 while (true) {
66 Drivetrain.drive(forward);
67 Drivetrain.setDriveVelocity(100.0, percent);
68 if (SC.objectDistance(mm) < 280.0) {
69 giro = static_cast<float>(mathRandomInt(1.0, 1.0));
70 grados = static_cast<float>(mathRandomInt(25.0, 25.0));
71 if (giro == 1.0) {
72 VOLANTE.setVelocity(100.0, percent);
73 VOLANTE.spinFor(reverse, grados, degrees, true);
74 wait(0.5, seconds);
75 VOLANTE.spinFor(forward, 40.0, degrees, true);
76 wait(0.5, seconds);
77 VOLANTE.spinFor(reverse, grados, degrees, true);
78     }
79     }
80 if (SI.objectDistance(mm) < 100.0) {
81 giro = static_cast<float>(mathRandomInt(1.0, 2.0));
82 grados = static_cast<float>(mathRandomInt(25.0, 25.0));
83 if (giro == 1.0) {
84 VOLANTE.setVelocity(100.0, percent);
85 VOLANTE.spinFor(reverse, grados, degrees, true);
86 wait(0.5, seconds);
87 VOLANTE.spinFor(forward, 40.0, degrees, true);
88 wait(0.5, seconds);
89 VOLANTE.spinFor(reverse, grados, degrees, true);
90     }
91     }
92 if (SD.objectDistance(mm) < 100.0) {
93 giro = static_cast<float>(mathRandomInt(1.0, 1.0));
94 grados = static_cast<float>(mathRandomInt(25.0, 25.0));
95 if (giro == 1.0) {
96 VOLANTE.setVelocity(100.0, percent);
97 VOLANTE.spinFor(forward, grados, degrees, true);
98 wait(0.5, seconds);
99 VOLANTE.spinFor(reverse, 40.0, degrees, true);
100 wait(0.5, seconds);
101 VOLANTE.spinFor(forward, grados, degrees, true);
102     }
103     }
104 wait(20, msec);
105     }

```

```
106 return 0;
107 }
108
109
110 int main() {
111 // initialize the random number system
112 srand(Brain.Timer.system());
113
114 whenStarted1();
115 }
```