REX

REX EVOLUTION SERIES
SUPER STAR TRANSFORMERS
8 IN 1

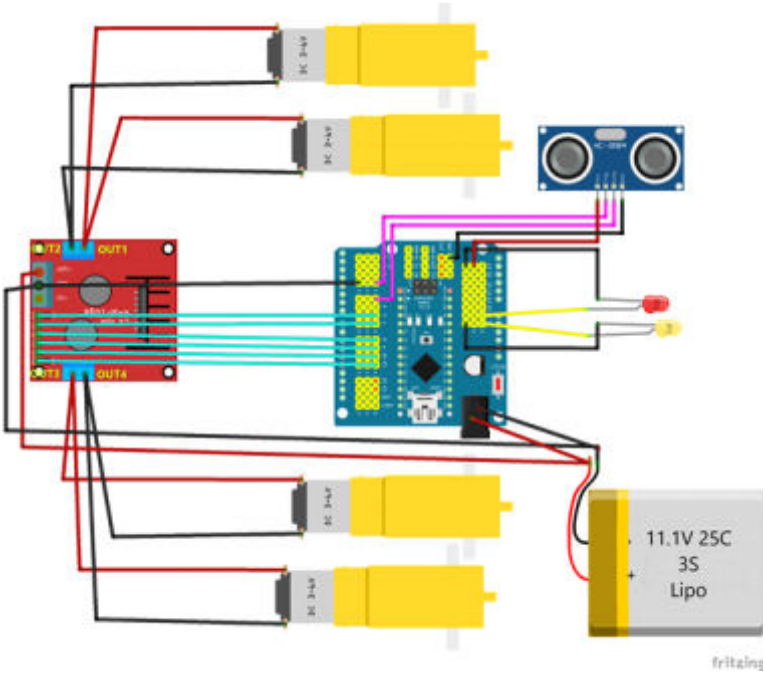
RoboSonic

Survivor Robot Avoiding Obstacle

Author: **Mustafa Kemal AVCI**

Connection Diagram

After assembling the parts of our Survivor Robot and assembling the electronic components, let's start coding by making the cable connections. Let's prepare the cable connections according to the connection diagram of the Survivor robot, which has overcome the following obstacles.



Attach the + poles of the motors on the right of the robot to the OUT1, the - poles to the OUT2 terminal and tighten the screws. Attach the + poles of the motors on the left of the robot to the OUT3 terminal and the - poles to the OUT4 terminal and tighten the screw.

OUT1 and OUT2 outputs on the motor driver board (L298N) are controlled from IN1 and IN2 pins. OUT3 and OUT4 outputs are controlled from IN3 and IN4 pins.

Let's connect the IN1 and IN2 pins, where the rotation direction of the right motor is set, to the digital pin 7 and 8 on the sensor shield. Let's connect the IN3 and IN4 pins, where the rotation direction of the left motor is set, to the digital pins 9 and 10.

The pins to be used for speed adjustment must have PWM support. The PWM pins on the Arduino Nano board are pins 3,5,6,9,10 and 11.

Let's connect the ENA pin, where the rotation speed of the right motor is set, to the digital pin 6, and the ENB pin, where the rotation speed of the left motor is adjusted, to the digital pin number 11.

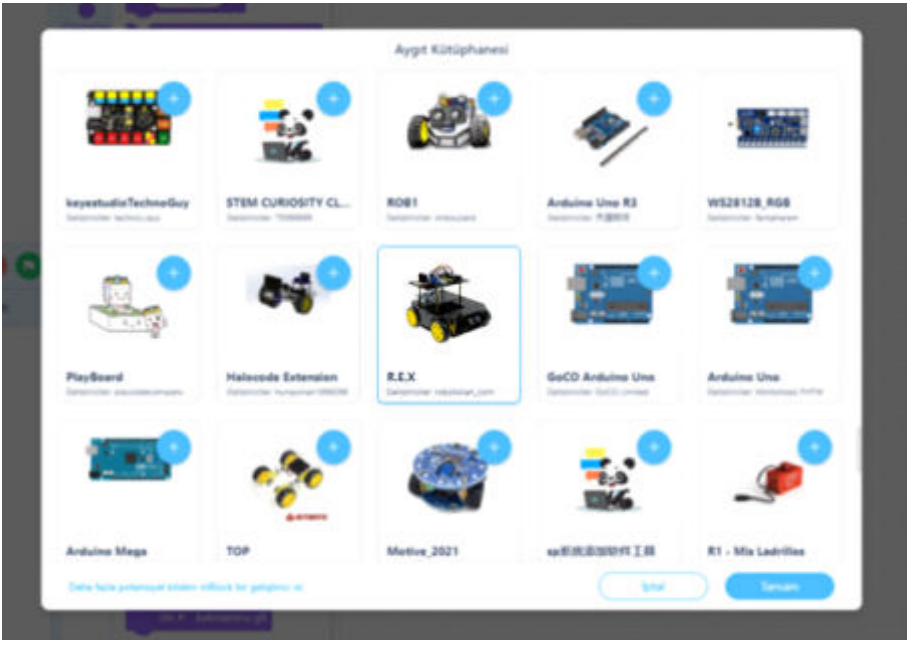
We have connected two LEDs to our robot in order to notify operations such as obstacle detection and reaction, and to provide convenience while checking the errors that may occur in the code. Let's connect the long leg of one to analog pin A0 and the long leg of the other to analog pin A1. You can connect the short legs to the appropriate one of the GND pins indicated with G on the sensor shield.

We will use an ultrasonic distance sensor so that the robot can detect the obstacles in front of it. Ultrasonic sensor emits sound waves and informs us by calculating the sound speed from the return time of the reflected sound wave. The name of the sensor is HC-SR 04. This sensor can measure between 4 cm and 4 meters. Let's connect the Trig pin to digital pin 3 and the echo pin to digital pin 4. Let's connect the GND pin to a pin indicated by G on the shield, and the VCC pin to one of the 5V pins indicated by V.

You should transfer your battery power (Lipo Battery or 6x AA battery) in the range of 7-12V to your motor driver and Arduino Nano sensor shield. For this, you have to double the power outputs of your battery. You must connect the + and - poles, one of each, to the terminals shown in the diagram on the motor driver. You should be able to plug the other one into the shield's power input as a jag. Finally, you must establish a common line with a cable from a G pin on the shield, that is, the GND pin, to the GND terminal on the motor driver.

Coding

After running the mBlock software, let's enter the device library and add the R.E.X device to our workspace.

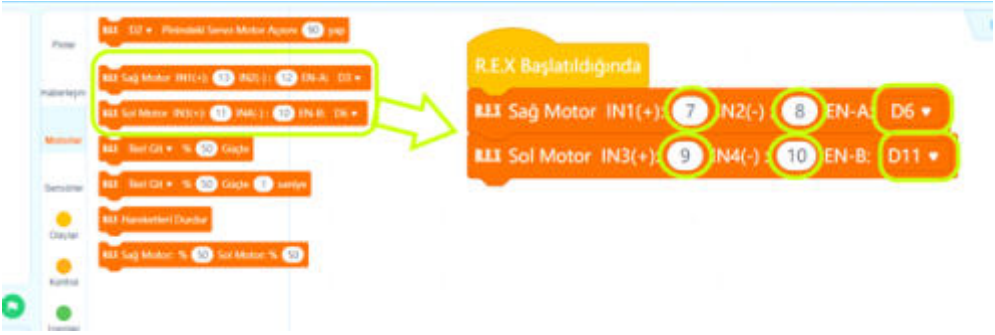


Our algorithm will be as follows.

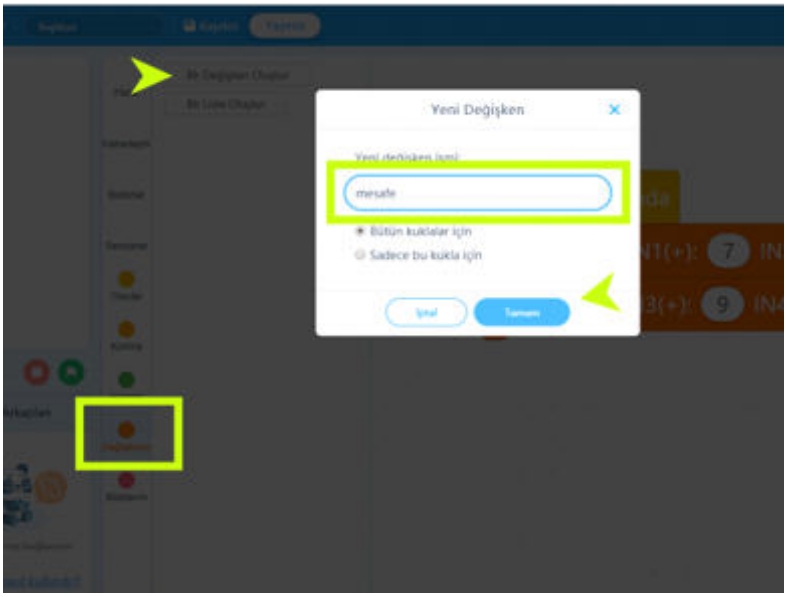
1. Start
2. Define motor pins
3. Define the distance variable
4. Read continuous ultrasonic distance sensor and assign its value to distance variable
5. If the value of the distance variable is less than 30 cm and more than 2 cm, turn right or left. (Right or left rotation will be randomly selected by the robot)
6. If the distance is not less than 30 cm, it moves forward.
7. Finish.

Drag and drop the "When REX Starts" block from the event blocks to the coding area.

Drag and drop the blocks in which we defined the Right and Left Motor control pins in the "Motors" category, under the "When REX Starts" block, respectively, as in the image. Write the pin definitions in the circuit diagram to the necessary places as in the image.

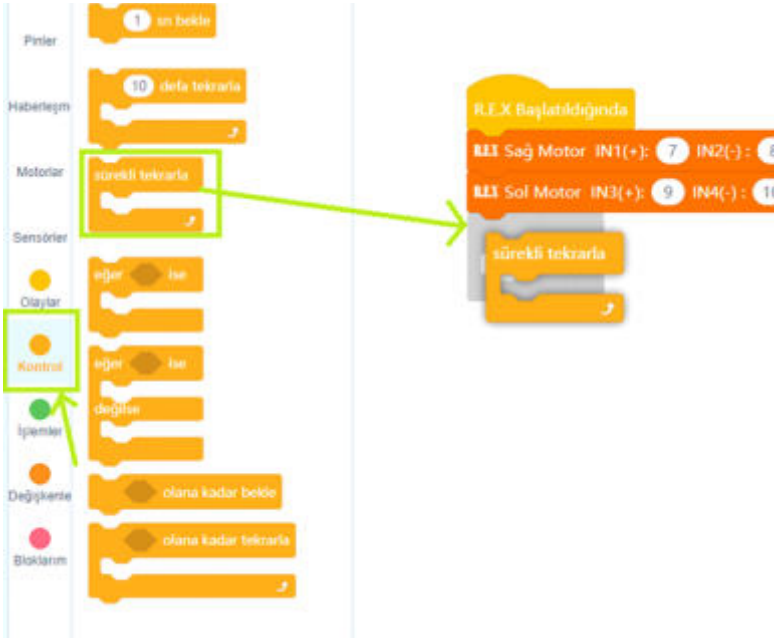


We will create a distance variable to store the distance value that the ultrasonic distance sensor will read. For this, we enter the "Variables" category and click the "Create a Variable" button. In the window that opens, we write the variable name as distance and press the OK button. Thus, we have defined our variable and set its initial value to 0.

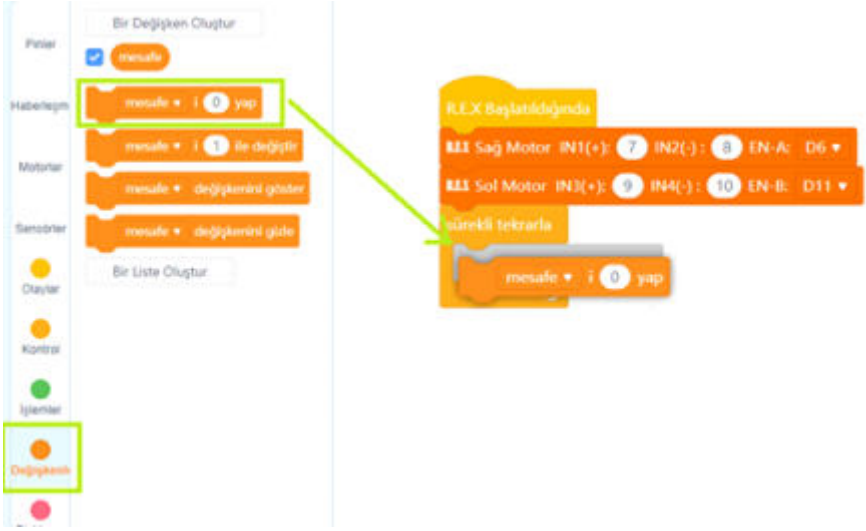


When we write the variable name as "distance" and press the OK button, the "Variables" category blocks should appear as follows.

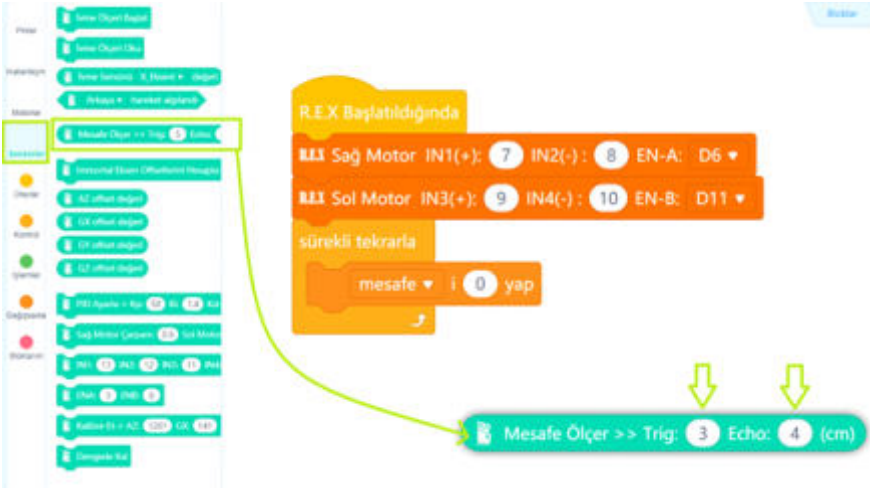
Since the distance will be read continuously and the motors will be given direction and speed according to the distance, we place the "Repeat continuously" cycle block from the category of "Control" blocks, just below the engine identification blocks.



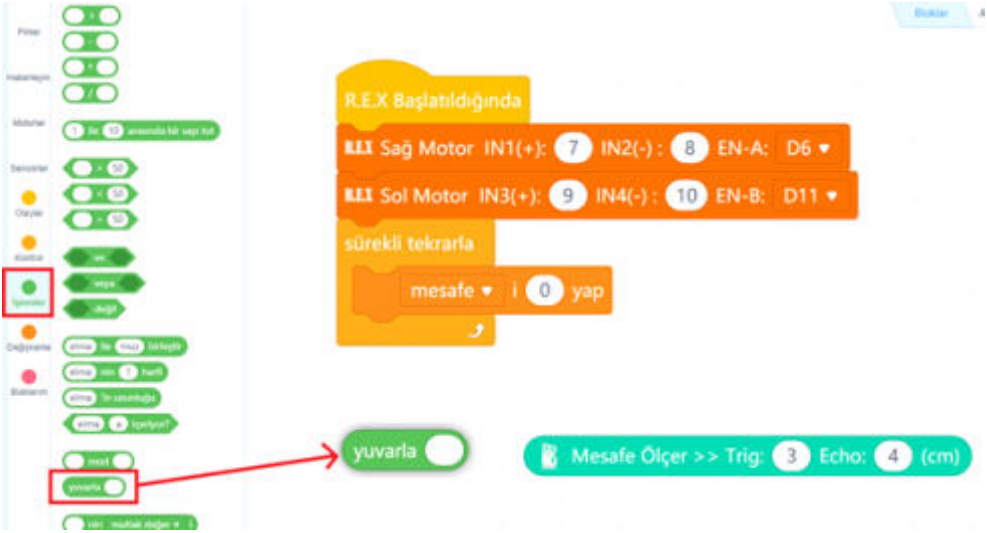
We will assign the value of the distance variable to the value read from the sensor continuously. Whenever we need the distance value in the program flow, we will use that variable. Let's place the "set distance to 0" block in the "variables" category inside the repeat constantly block as in the image.



The distance measuring sensor (HC-SR04) gives us the accuracy of one hundredth of a cm. For example, 36.42 cm. Let's move the distance meter block from the "Sensors" category to our work area as in the image below. As we have determined in our circuit diagram, let's change the pin number that we connect the Trig pin to 3, and the pin number to which we connect the Echo pin to 4.

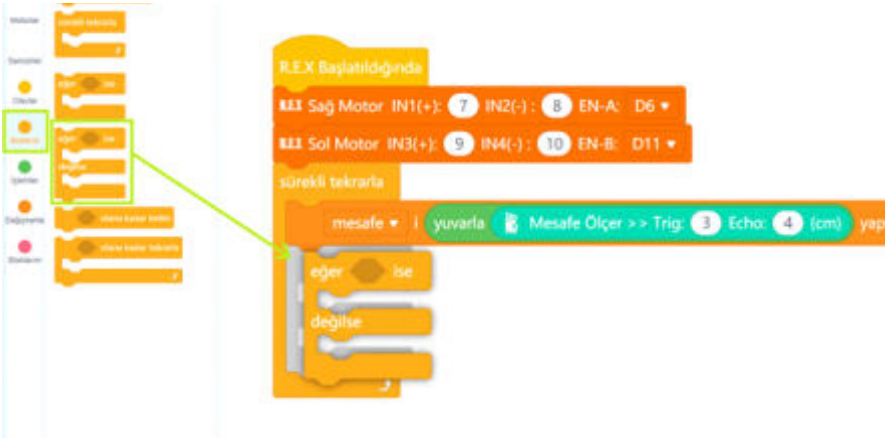


The fact that this number, which we express as a decimal, appears as an integer allows the robot to make more stable measurements for us. We will use the "Roll" block for this. We take the round block from the "Operations" category and move it to our workspace as follows.

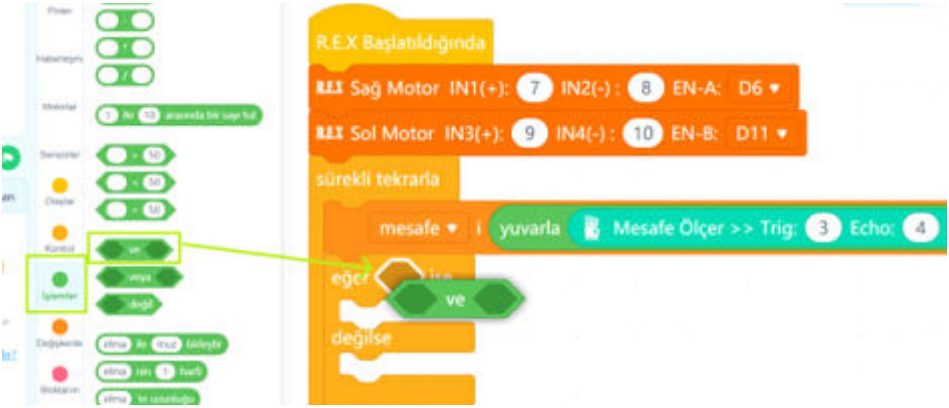


First of all, let's drag and drop the distance meter block into the round block as in the image. The Range Finder block will fit inside the roll block and become movable together. Then, let's hold the round block from the green part and drag and drop the distance to the value field of the make 0 block. Thus, we make the value of our distance variable rounded up from the value read from the distance sensor.

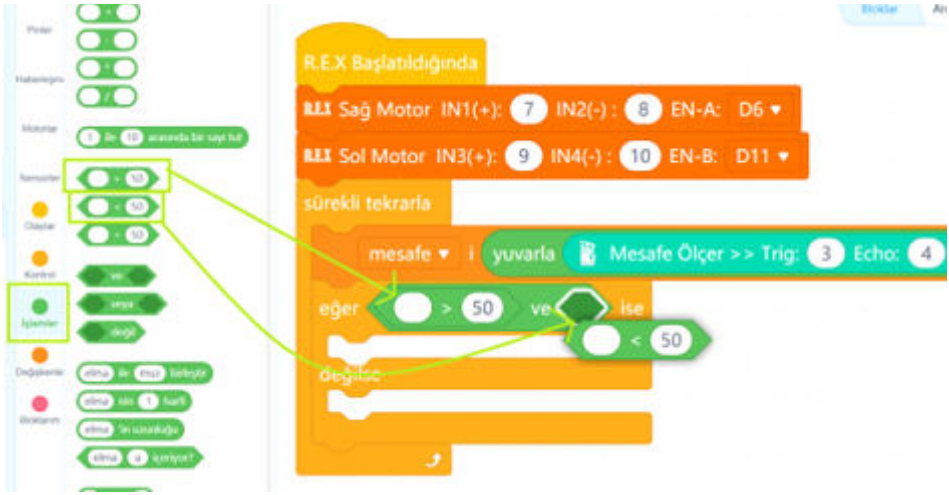
Now we can prepare the blocks on which we will perform the mathematical test. If the value of the distance variable is within a certain range, we want the motors to turn left or right to avoid the obstacle, and if it is not within a certain range, we want it to show the forward behavior. In other words, if the condition we are testing is true, a command will be executed, and if it is false, a different command will be executed, drag and drop the "If If Otherwise" block under the distance variable block in the "Control" blocks category.



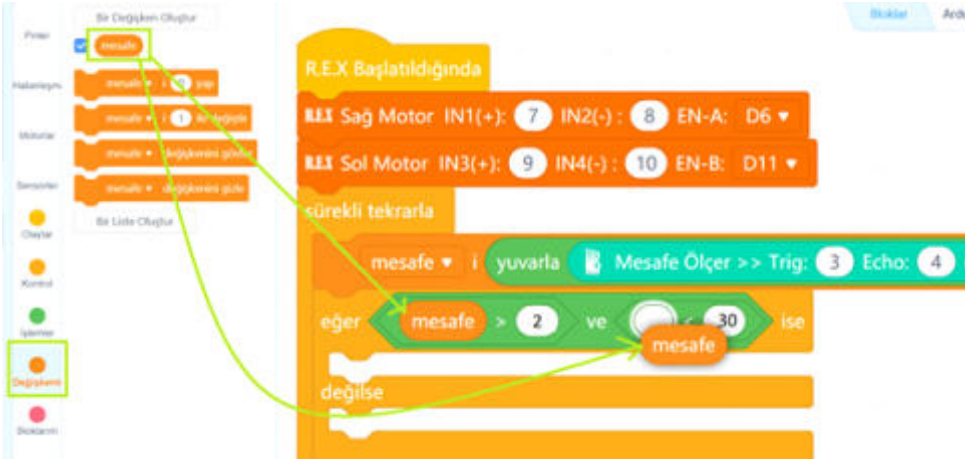
The mathematical test we will do will be as follows; if the distance variable value is greater than 2 cm and less than 30 cm. This test means to check if the distance value is between 2 and 30 cm. In order to express this with code blocks, let's move the "and" operator block from the "operations" category to the condition field of the if not, as in the image.



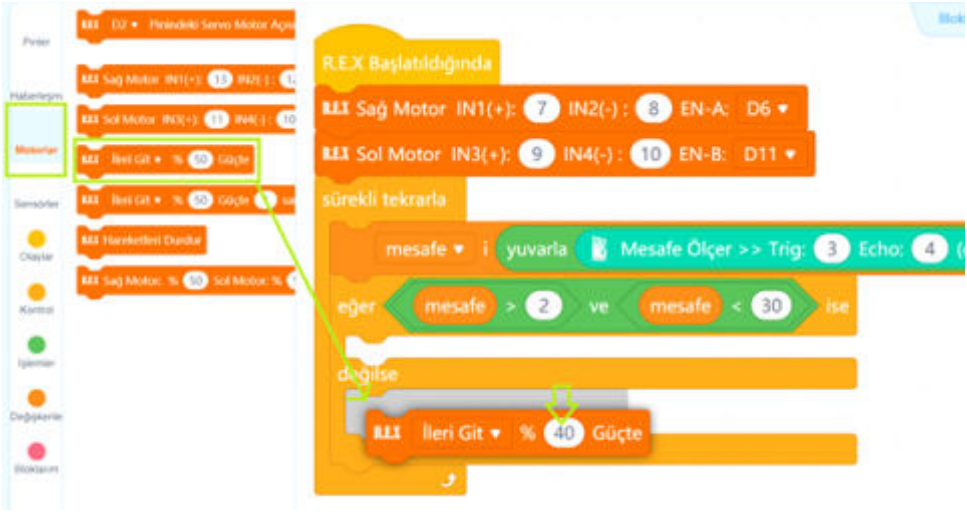
we had two separate conditions. the distance is greater than 2 and the distance is less than 30. Let's drag and drop our two conditions into the block and smaller than the comparison operators as follows.



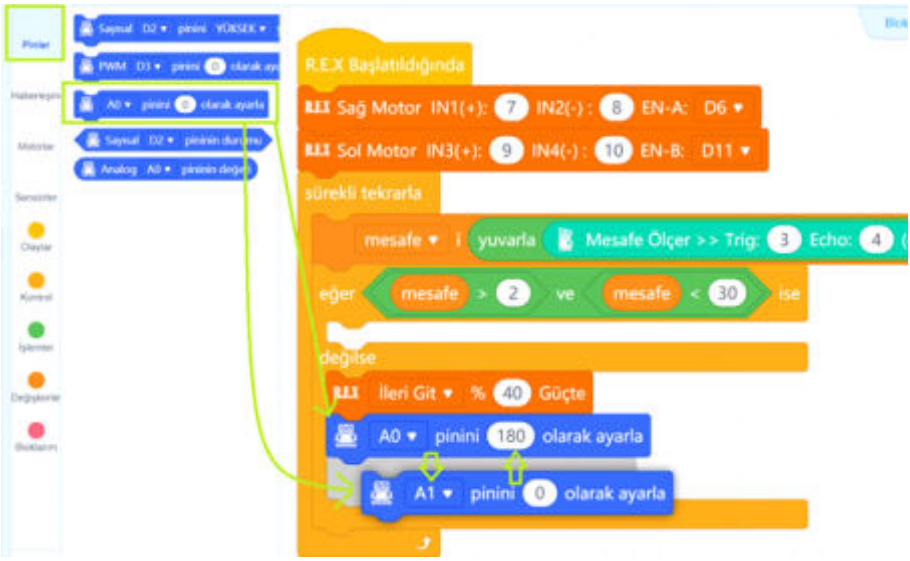
greater than and less than we will place our distance variable in the empty fields inside the comparison operator. For this, let's drag the distance variable value block in the variables category and drop it to the places shown in the image below. Let's replace the numbers 50 with 2 and 30.



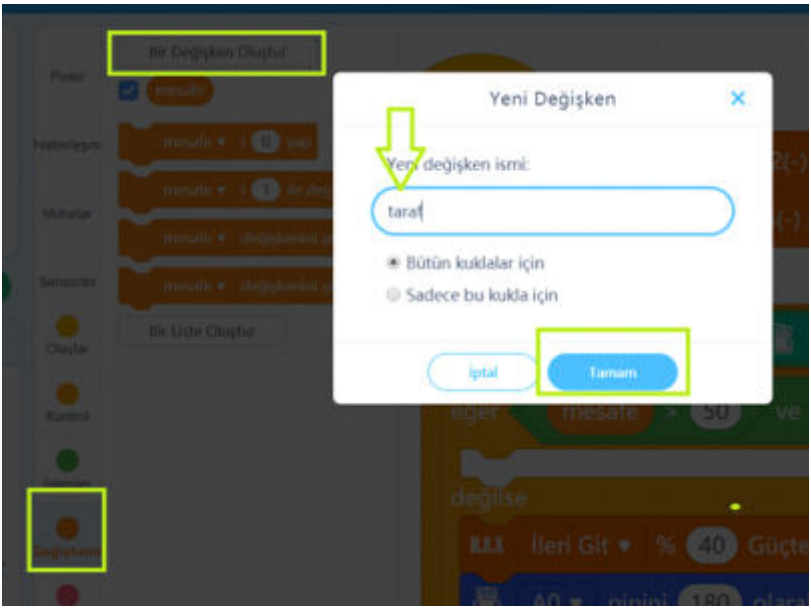
If our distance value is between 2 and 30, our robot will randomly turn to one side, otherwise it will continue to go forward. Until there is an obstacle standing less than 30 cm in front of him. Go forward Let's drag and drop the 40% power block to the "if not" command field as in the image.



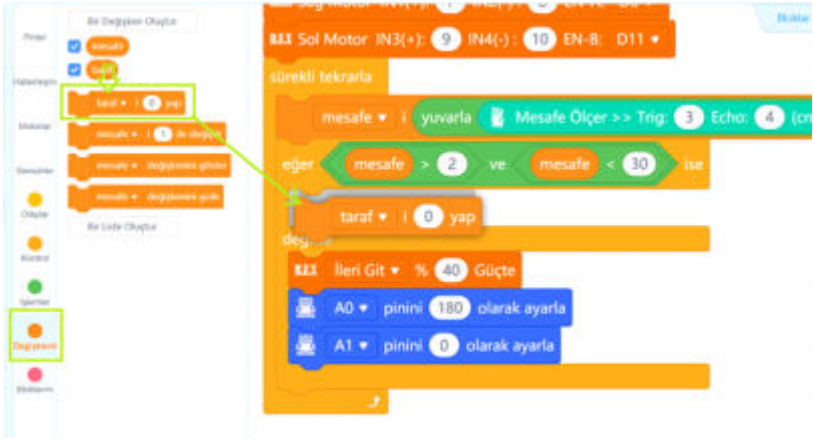
When our robot moves forward, let the front led turn on and turn off the rear led. Thus, we can understand in which range the value read from the distance sensor is in case of a problem with the motors. For this, let's drag and drop the "Make A0 pin 180" and "A1 pin 0" blocks in the "Pins" category, if they are not, to the command area as in the image below.



In the last part of our codes, we will code what our robot will do when an obstacle is detected. If not, we will add our necessary codes to the command field of our block, if not. First of all, we will determine the process of turning in a random direction, that is, right or left. Let's create a variable named "side" in it. We create the variable as in the image below.

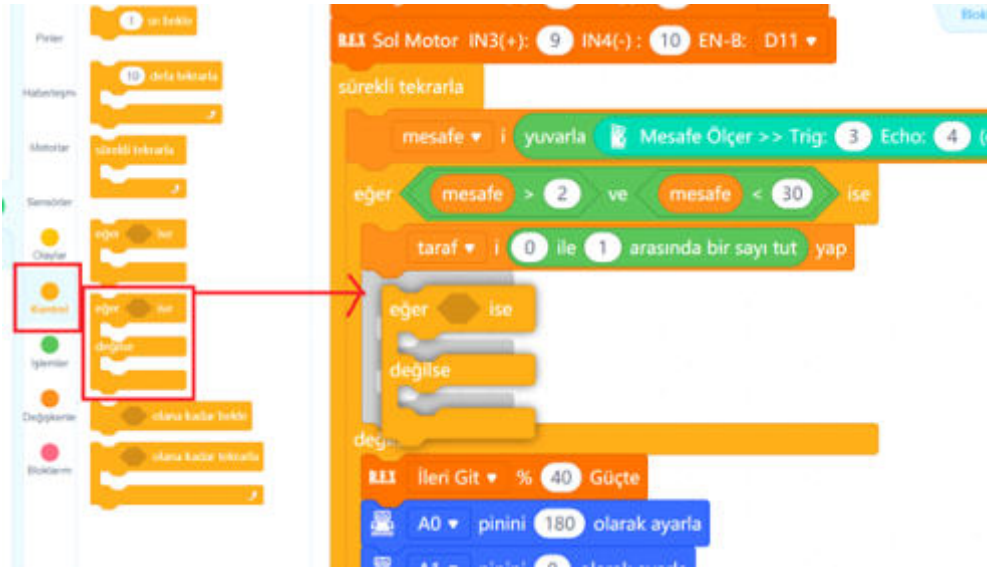


We will set the value of the side variable to 0 or 1 randomly each time an obstacle is detected (every time the distance value is measured between 2 and 30). If the side variable is 0, we will make our robot rotate 1.2 seconds to the right, otherwise 1.2 seconds to the left. In order for the side variable to take a random value of 0 or 1, first, let's drag and drop the make side 0 block to the command area as follows.

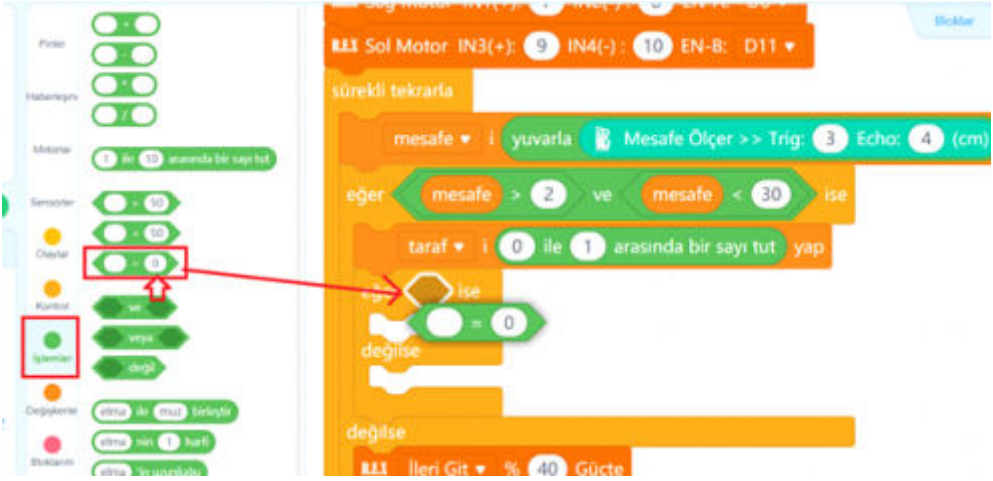


In order for the party variable to take a random value, let's change the "keep a number between 1 and 10" block in the "transactions" category to be between 0 and 1, and drag and drop it into the value field of the "set side to 0" block as follows.

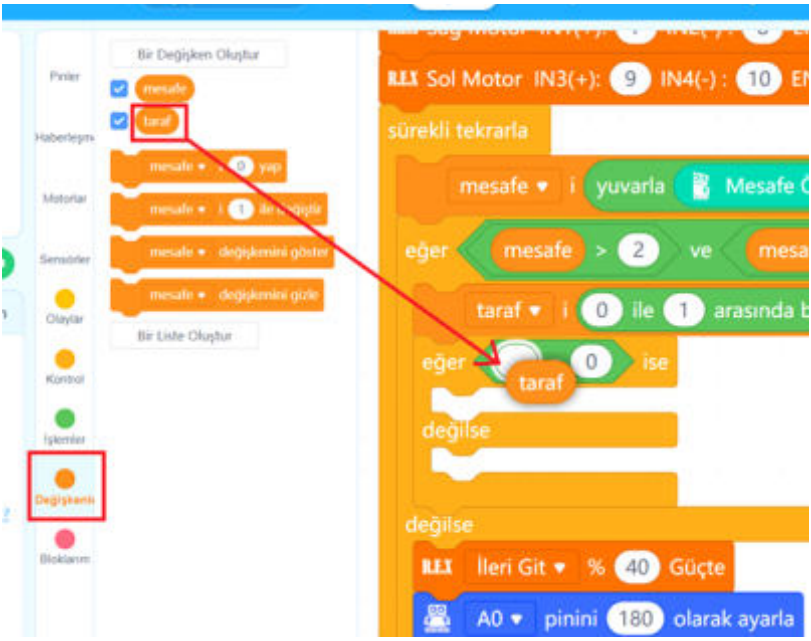
To perform the rollover, we need to query whether the value of the party variable is 0 or 1. If the side variable is 0 then we will issue a right turn command and not 0 (if 1) we will issue a left turn command. For this, let's place the "if not" block in the "control" category just below the block where the value of the side variable is set randomly, as in the image.



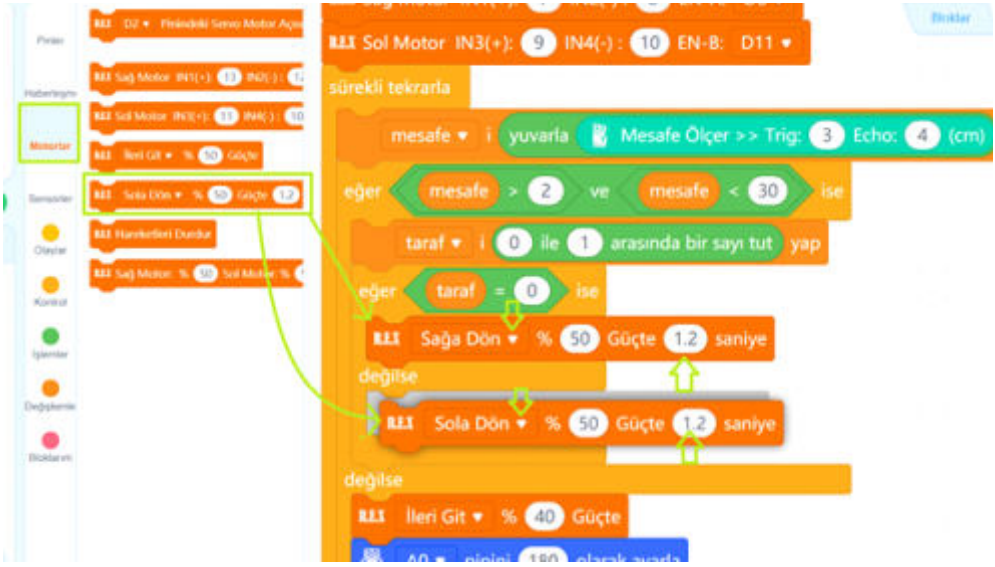
Now we have to settle the condition for the rotation of our robot. If not, let's drag and drop the "equal" operator block in the "operations" category as in the image below to express the condition if the side variable is equal to 0 in the condition field of the block.



Değişkenler kategorisinden taraf değişkeni değer bloğunun eşittir operatör bloğunun değişken alanına sürükleyip bırakalım.



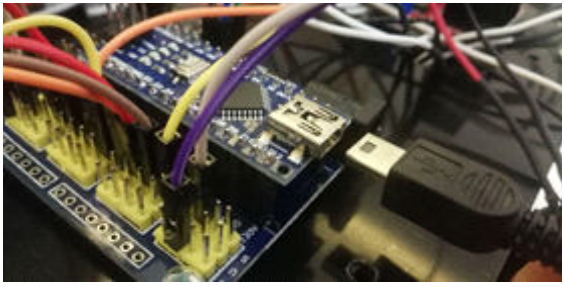
In order for our robot to turn right for 1.2 seconds if the side variable value is 0, or to turn left for 1.2 seconds (if the side variable value is 1), let's drag the blocks and place them in the "If not" block, as in the image in the Motors category.



Finally, let's turn on the rear led and turn off the front led while the rotation process is taking place. For this, let's drag and drop the A0 pin in the Pins category just before the If not block, using the Make 0 block.

A0 and A1 pins are analog pins, they can also send voltage to the pins in the 0-255 value range, such as PWM. Since most of the LEDs reach the maximum brightness around 3V-2V, we have not shortened the life of the LEDs by providing a voltage of around 3V with a value of 180 instead of giving the A0 and A1 pins a value of 255, which means 5V.

The finished version of our code should look like the image below.



We can now upload the codes we have prepared to our robot. We connect one end of the usb cable to Arduino nano and the other end to our computer.

Click on the connect button in the installation mode and tick the show all connectable devices option. The mBlock software will automatically insert the COM port number to which our CH340 chip card is connected. If connection is not possible, you can select other COM port numbers from the drop-down list.

Once connected, “disconnect”, “settings” and “Install” buttons will be active. Let’s upload the code to our robot by clicking the upload button.

Execution of Codes
As soon as the code is loaded, our robot will start to move by reading the data from the sensor. After loading the codes, you can make the robot run the codes by removing the usb cable and connecting the cables of the battery providing power to the sensor shield and the motor driver, or by turning on the key, if any. You can achieve the precision you want by adjusting the rotation times of your robot according to the battery power, the forward speed and the rotation speed.

Possible Problems and Solutions
If your robot is only going back and forth, not turning left or right or vice versa,
Make sure to check the locations of IN1,IN2,IN3 and IN4 pins on the shield and on the motor driver. They may be confused.

If your robot is going backwards in the Forward code, and turning left in the turn right code; Make sure that OUT1 and OUT 3 on the motor driver are connected to the (+) pole of the motors, and OUT 2 and OUT 4 are connected to the (-) pole of the motors. Also, make sure that the motors on the right of the robot are connected to OUT1 and OUT2 on the driver, and the motors on the left are connected to OUT3 and OUT4.

If your robot encounters an obstacle, the back led does not light up, and it always continues to go forward; Make sure that the jumper cables connected to the Trig and Echo pins of the HC-SR04 (distance sensor) module are more stable. With regard to the stance angle of the sensor, a very small movement in the jumper cables on these two pins causes the sensor to read 0 (si) value, which causes the obstacle detection system to not work.

YouTube

youtube.com/robotistan

FORUM

robotistan

forum.robotistan.com

BLOG

robotistan

maker.robotistan.com

Robotistan Electronics Company

Mustafa Kemal AVCI (Content) - Fadıl PALA - Mehmet AKÇALI (Editor) - (Mehmet Nasır KARAER (Graphics) - Mert ALTUNTAS (Translation)
info@robotistan.com - www.robotistan.com
Phone: 0850 766 0 425