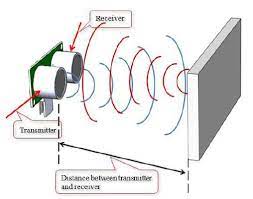
NITRObot Maze solving

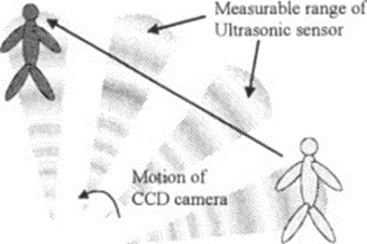
Ultrasonic sensors are sensors that use sound waves to detect nearby objects. Ultrasonic sensors are a non-contact sensor meaning they can operate without touching the object that needs to be detected.

Ultrasonic sensors work similarly to the echolocation used by dolphins and bats. They emit high-frequency sound waves to the target to be detected. When an object is present within the sensor’s detection range, it will reflect a sound wave back to its source. Distance is measured by how long it took the soundwave to return (using the known speed of sound in the calculation).



Ultrasonic sensors can be used in environments with high levels of dust or humidity and perform better than infrared sensors when there is a lot of smoke. Ultrasonic sensors can measure the distance to a wide range of objects regardless of shape, color or surface texture. They are also able to measure an approaching or receding object. Certain variables, such as target surface angle, changes in temperature and humidity, and reflective surface roughness, can affect the operation of the ultrasonic sensors.

Ultrasonic Sensors are widely used in the industry and have application in mobile robotics. The next figure shows an example of obstacle and human detection in front of the mobile robot used for collision avoidance and human safety when in proximity of the robot.

******

***NOTE:*** *In robotics, engineers usually do not rely on single type of sensor for a particular task, as each type of sensor has its own downsides and limitations. Combining data from different types of sensors in a single system is called* ***Sensor fusion****.*



*Sensor fusion gives a more accurate or complete perception of the environment. The benefits of sensor fusion are numerous. For example:*

*Mirrors and glasses are quite common objects that appear in our daily lives. While laser scanners (LiDAR /Light Detection and Ranging/ sensors) play an important role in the field of robotics, they have problems with mirror reflection and glass transparency. The information collected by LiDAR sensors can be combined with ultrasonic sensor data to detect the potential obstacles not seen by laser scanners and improve obstacle detection and localization of objects in difficult-to make out environments like foggy areas.*

**NITRObot** is equipped with different sensors and designed to let students simulate tasks from the industrial robotics world.

**Maze solving Robot** exercise allows beginners and students to get experience using ultrasonic sensors.

In this program, we will use **NITRObot**'s ultrasonic sensor to perform a wall following task similarly to the way it is done in the big industrial robots.

**WHAT IS A MAZE?**



Maze is a network of paths or tunnels designed as a puzzle through which one has to find a way from the entrance to the exit.

There are a number of different maze-solving algorithms, that is, automated methods for the solving of mazes. The random mouse, wall follower, Pledge, and Trémaux's algorithms are designed to be used inside the maze by a traveller with no prior knowledge of the maze.

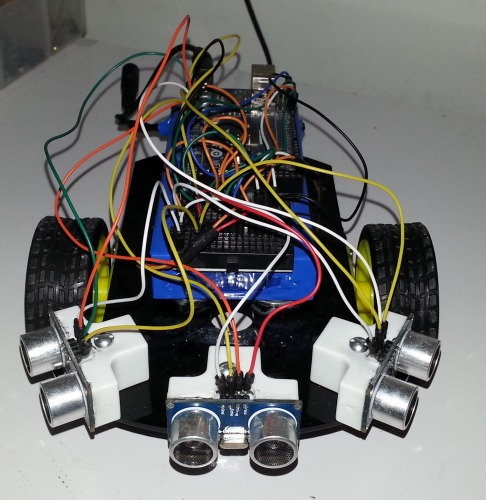
**Wall Follower Algorithm**

The most basic technique to solve a maze is the “right hand rule” (alternatively “left hand rule”). Simply touch the wall to the right and keep your hand glued to it as you wander along. When you hit a junction, pick the option that keeps your hand connected to the wall. This way you’ll find the exit.

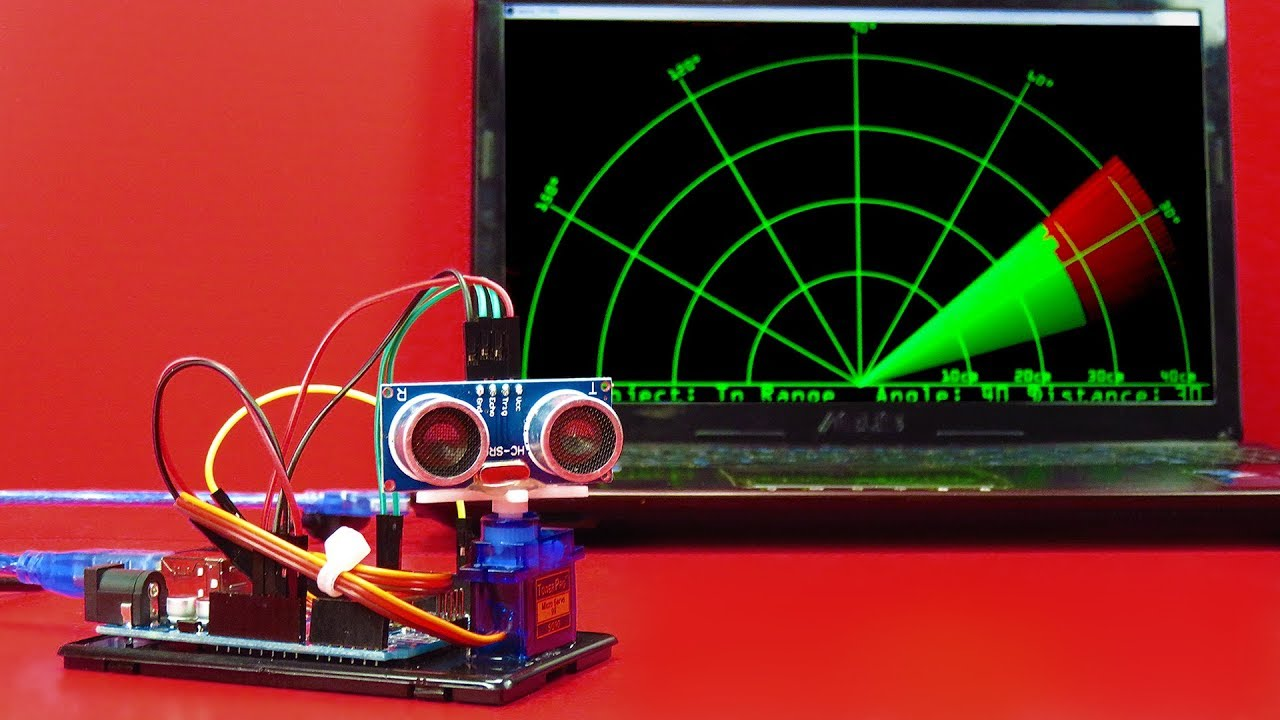
If the maze’s inner walls are all connected, you can picture them as a single piece of string looping back and forth, occasionally doubling over itself. Unravelling the string creates a circle. So, when you follow the wall with your hand it may feel like a strange squiggly route, but it turns out that you’re heading in a straight line. Would the “wall follower” technique also work with your left hand? Why might you choose one direction over the other?

What’s more, you need to put your hand on the wall the moment you enter the maze. The right-hand rule can fail if you start in the centre, or the maze has bridges and crossovers. The biggest danger is getting stuck on an island: an isolated section of wall disconnected from the rest of the labyrinth. To deal with these features, we need a more advanced solution.

**Ultrasound sensor mounted on a hobby servo motor – DIY radar**

While it is possible to use a mobile robot with two or three permanently mounted Ultrasonic sensors to perform the Maze solving task, in order to learn more with every new task, we will make the things a bit more complicated…

In **NITRObot maze solver program** we will use a single Ultrasonic sensor, mounted on top of a hobby servo, creating a something similar to a radar.



While moving the robot we will check the distance to the wall in front of NITRObot, then turn the servo 90 degrees to the right (or left in the case of “left hand algorithm”) and read the distance to the side wall of the maze. Then turn the servo back to straight ahead position and start the sequence over.

1. **The First step is to design and build the maze.**

**NITRObot dimentions:** length 25.0 cm., width 16.7 cm.

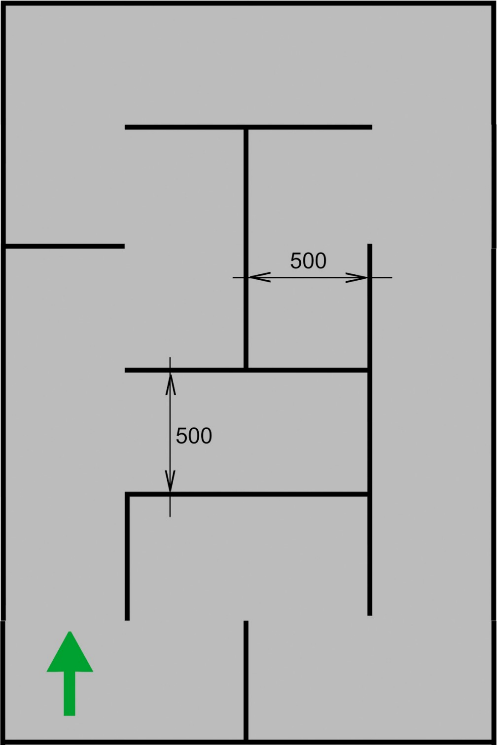
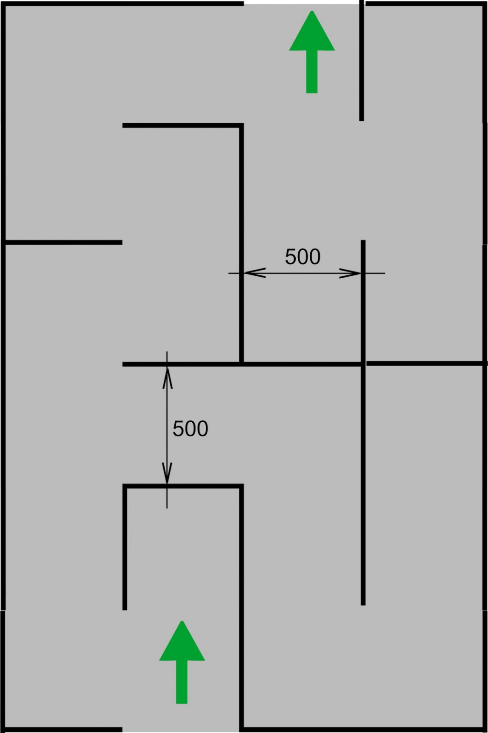
**Maze parameters:** In order for the robot to be able to safely make an U turn, the maze corridor width needs to be 3 times the robot width, which is equal to 50.1cm, we will approximate this value to 50 cm.

The height of the walls should be 200 mm (20 cm) or more in order to ensure that it is tall enough for the ultrasonic sensor to detect the walls from distance.

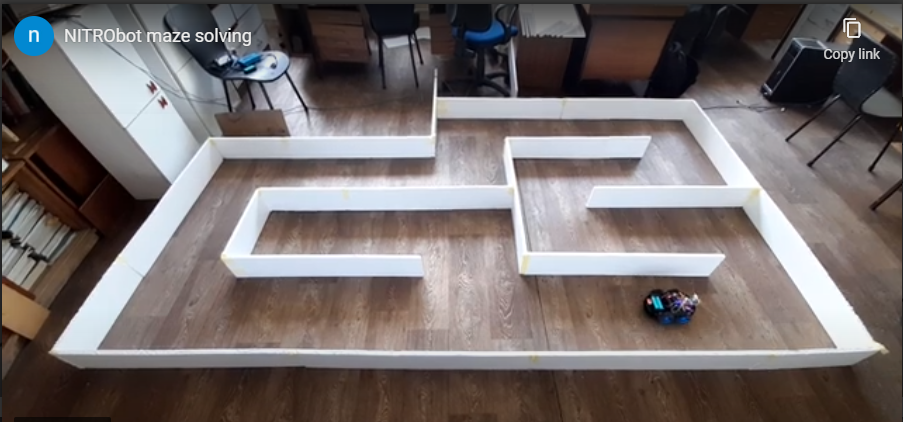
Using cardboard, thin (2-3 mm.) Styrofoam insulation panels or some kind of thin plastic, cut pieces to size and stick them together and to the ground using sticky tape.



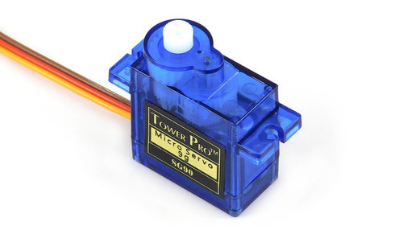
The reference design for open and closed maze types are depicted in the drawings below.



Below is a real-life example of a closed NITRObot maze:



1. **The next step is to refresh your knowledge on how to read distance using the RUS-04 Ultrasonic sensor mounted on the NITRObot from the previous NITROclubs lessons and exercises.**
2. **Then, from the previous NITROclubs lessons and exercises refresh your knowledge on how to control the SG90 – small hobby servo using the Arduino servo library.**

****

1. **Download the source code from GitHub**

The link to download the sketch is: <https://github.com/nitroclubs/NITRObot_maze_solver>

1. **The implementation of the algorithm for “right hand rule” maze solving**

Each iteration of the main loop starts with two consecutive calls to the getDistance function, which moves the servo to position (90° /front/ at the first call and at the second call 0 degrees /90° to the right/) and then reads the value from the Ultrasonic sensor.

Using the distance readings from the ultrasonic sensor at the two predefined servo positions (straight ahead and 90° to the right), by using a set of if…else statements, the robot position relative to the maze is determined and a corresponding value is set to the robotPosition variable.

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/ifStatementConditional>

If there is an obstacle in front of the robot closer than 20 cm, the robot should make a 90 degree turn to the left:

robotPosition = 1;

The possible positions of the **NITRObot** inside the corridor are as follows:

|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_ROBOT\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|\_\_\_\_\_\_\_\_\_|\_\_ROBOT\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|\_\_ROBOT\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_ROBOT\_\_|\_\_\_\_\_\_\_\_\_\_|

|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_ROBOT\_\_|

50cm 35cm 29cm 21cm 15cm 0cm

When the robot sensor is between 21 and 29 cm from the right-hand wall, we can assume the NITRObot is on the imaginary centre of the corridor:

robotPosition = 7;

If the robot is to the left or to the right of the centre, the robot needs to make a shallow or more aggressive turn (depending on the distance from the wall) in order to go back in the centre of the corridor.

The robot motors control is realized in the switch…case based on the value of the robotPosition variable. <https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/>

Each iteration of the main loop starts with two calls to the getDistance which moves the servo to position and then reads the value from the Ultrasonic sensor.

float getDistance(int servoAngle, int delayAfterServoMovement){

…

}

1. **Try NITRObot\_Maze\_solver.ino program (sketch).**

* **Replace the default left and right speed values (of 100) with the ones you obtained with the one obtained from the calibration of your robot using** **NITRObot\_motor\_calibration.ino:**

****

* **Upload NITRObot\_Maze\_solver.ino to the robot**
* **Place the NITRObot in starting position**

***NOTE:***

*There is a delay set to 1 second inside void setup() {….}, which will delay the start of the robot in order for the robot not to jump out of your hands when you turn on the switch. You can change the delay time or remove the delay if you wish.*

* **Switch on the robot**
* **ENJOY!**
* **If the maze type is an open one wait for the NITRObot to get to the exit and switch it off. If the NITRObot is palced inside a closed maze, it will go back to the starting position and then go trough the maze again and again until you switch it off.**

1. **T** **urn timing tunning**

In the more advanced robots, a combination of wheel encoders, inertial measurement units (IMU) and other sensors are used to insure, that the robot make turns to the exact heading (turn angle).

**NITRObot** is a differential drive steering type of vehicle – you accelerate one side (and eventually slow down the other side) in order to make a turn. The wheels on both sides are slipping while making the turn.

As a robot for beginners, **NITRObot** does not use wheel encoders (sensors counting pulses on each revolution of the wheel shaft) and other motion sensors, in order to ensure that the NITRObot turns to a predefined angle, we need to find the proper timing value at a given (different) speed of rotation of the wheels on both sides, which will result in a turn at the desired heading.

If your **NITRObot** does not make a complete 90 degree turn, or the robot overturns, you need to need to use the trial-and-error method and tune it by changing the delay values in the switch…case control structure’s case 1 and case 2 where the turn to the right and to the left are implemented.

***Do the*** ***turn timing tunning until NITRObot starts to successfully solve the maze.***