Project. Intelligent and Distributed Systems (SID)

A. Belbachir, J. Gustave, A. Ozturk Suri, O. Gurcan, IPSA February 14, 2022

Context

In this project the objective is to program a robot in order to perceive, reason and act in its environment. The robot will need to go from a known starting point to a known ending point using its reduced perception (Ultrasonic sensor and Inertial Measurement Unit).

The project will be divided in two main parts:

- An implementation on a real wheeled robot with its embedded sensors (see section A1),
- An implementation of this behaviour in a simulated environment (see section A2)

Requirement

Install Arduino software following this link.

Instructions

You will be divided into groups of 4 to 5 students.

Each student will have a group number.

Submission: You will need to submit a document (maximum 4 pages) + your programs with comments + slides on pdf format for 10min of talk. The files will be submitted on Moodle (one per group): deadline 3rd April.

Date of presentation: During the last session IN424(4/4).

Mark: The maximum mark is 20 points. Expose your solution in 10min (slides including flow charts to illustrate your implementations) + 5mn of questions. A competition will be held in order to evaluate in real situation the behaviour of your approach.

Be careful! the document + programs + slides should have just the group number (e.g. Group1.pdf, Group1.cpp), which will be given by the teacher.

A1. IPSA BOTS

PART I: Configuration of the motors and the embedded sensors

The objective of this section is to understand how the robot works and implement some functionalities in order to perceive its environment and control the motors.

Step1: General information

Arduino is an open source prototyping platform able to read inputs and activate outputs. Inputs can be « light », « button », « messages », etc. Outputs can be « motors », « LED », and others. Arduino can be programmed by a set of instructions using Arduino programming language. Arduino was born at the Ivrea Interaction Design Institute, to allow students without a background in electronics and programming to prototype and program fast.
The main advantages of Arduino are the following:

- It is open source,
- It is available on several operating systems,
- Arduino boards have several pins to plug sensors,
- Easy to program and several projects are nowadays using it.

Step2: Getting started

To program an Arduino board you need to :

1. Download the Arduino Platform here
2. Plug the Arduino board to your computer
3. Run the software (Arduino IDE)
4. Choose the right board type (install the board library if the card type you're looking for is not listed yet)
5. Choose the adequate port to upload your program to the Arduino card

Now, try to test the following program:

Listing 1: First steps with Arduino.

```
1 void setup(){
2 }
3
4 void loop(){
5     Serial.print("Just to print a sentence in the terminal ");
6 }
```

Q1) What can you see ? Do you need to modify something ?

.....

Step3 : Ultrasonic sensor, what is it ?

(a) HC SR04

(b) HY SRF05

Figure 1: Ultrasonic sensors

HC SR04 and *HY SR05* are ultrasonic sensors providing distance measurements from 2cm to 400cm. It has a focus of less than 15 degrees.
This sensor uses ultrasound waves to measure distance just like bats and dolphins do. Ultrasonic sound has such a high pitch that humans cannot hear it. This particular sensor sends out an ultrasonic sound that has a frequency of about 40 kHz.
The sensor has two main parts: a transducer that creates an ultrasonic sound and another that listens for its echo. To use this sensor to measure distance, the smart program must measure the time it takes for the ultrasound to travel. Sound travels at approximately 340 meters per second. First the sound travels away from the sensor, and then it bounces off a surface and returns back.
The sensor is already connected to the board, thus, **you do not need to modify the wiring!**
Pulse the trigger (Trig) pin high for at least $10\mu s$ (microseconds) and then wait for a high level on the echo (Echo) pin. The amount of time the echo pin stays high corresponds to the distance that the ultrasonic sound has travelled. The quicker the response, the closer your sensor is to an obstacle.

Q2) What is pulseIn() and what is it doing ?

.....

Q3) How can we use pulseIn() to get the propagation time of the wave ? Explain in details.

.....

Listing 2: Controlling the ultrasonic sensor

```
#define echoPin 29 // Echo Pin
#define trigPin 25 // Trigger Pin

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  /* The following trigPin/echoPin cycle is used to determine the
  distance of the nearest object by bouncing soundwaves off of it. */
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  /*.....TO DO 1.........
  Get and display the propagation time*/
  Serial.println("...propagationTime...");

  /*.....TO DO 2.........
  Calculate the distance (in cm) based on the speed of sound.*/

  //Print the distance to an obstacle
  Serial.println("...DISTANCE...");

  delay(200);
}
```

Q4) Modify the following code using the Arduino IDE to print the propagation time of the wave and the distance to the obstacle.

.....

Q5) Can you explain, line by line what are the following instructions doing ?

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

pulseIn(echoPin, HIGH);

. . . . . .

Q6) What would be the sensor error ?

. . . . . .

You may have noticed that the ultrasonic sensor is placed on a servomotor, a device composed of a continuous DC motor, control electronics, a potentiometer and gears for the range of rotation.

Figure 2: HS-65-HB (left) and SG90 (right) servomotors

Figure 2 shows the two types of servomotors used to rotate the ultrasonic sensor of your robot. They can move back and forth about 180 degrees.

Q7 What might be the advantage(s) of placing the ultrasonic sensor on top of such device ?

. . . . . .

Write the following program using Arduino IDE.
Remark:
Depending on the type of the servomotor that is embedded in your robot, you will use line 10 (uncomment line 10) of the code **or** line 11 (uncomment line 11) of the code to the Arduino board.

Listing 3: Manipulate the front servomotor

```
#include <Servo.h>

Servo motorFront;

void setup() {
  /*
    If you are using the sg90 servomotor, uncomment line 10
    Otherwise, uncomment line 11
  */
  // motorFront.attach(3, 530, 2500); //for the sg90 servomotor
  // motorFront.attach(3, 800, 2500);  //for the hs65hb servomotor
  motorFront.write(90);
}

void loop() {
}
```

Q8 Based on the official documentation, what does the instruction you uncommented do?

.....

Q9 Modify the previous code to smoothly turn left and right the ultrasonic sensor of ± 90 degrees. If the sensor cannot reach these positions, modify slightly the value of the minimum and maximum PWM [1] pulses assigned to the variable `motorFront`.

.....

Q10 Propose a solution that allows the robot to measure the distance to potential obstacles located on its left and its right.

.....

Optional questions:

Q11*) Can you make a real time 3D plot of the perceived distance in a graphic (x=timeClock, y=distance to obstacle, z=propagation time)?

.....

[1] https://www.arduino.cc/en/Tutorial/Foundations/PWM

Q12*) Can you, in a 2D graph, show any detected obstacle such a radar is doing ? (real time)

.....

Control the servomotors with the Arduino board

General information

There are several types of servomotors but for this part, you will focus on continuous rotation ones: they allow the user to rotate the motor shaft at a specific velocity, clockwise and counter-clockwise.

Figure 3: Servomotor used during this project

Figure 3 shows the parallax continuous rotation servomotor used to move the robot.
Both servomotors are connected to an Arduino shield mounted on the Arduino board.

Q1) According to the datasheet of this servomotor, what is the range of the PWM used to control it?

.....

Arduino programming

Listing 4: Manipulate servomotors

```
#include <Servo.h>
#define pwm_min ... //Add the minimum value of the PWM (in microseconds)
#define pwm_max ... //Add the maximum value of the PWM (in microseconds)

byte pin_motor_left = 5;
byte pin_motor_right = 6;
Servo motorL, motorR;

void setup() {
  motorL.attach(pin_motor_left, pwm_min, pwm_max);
  motorR.attach(pin_motor_left, pwm_min, pwm_max);
  motorL.writeMicroseconds(1700);
  motorR.writeMicroseconds(1300);
}

void loop() {
  delay(20);
}
```

Q2) Write the above program using Arduino IDE. Add the minimum and maximum values of the PWM related to this motor according to your answer to the previous question. Upload the program to the Arduino board. What can you see ?

.....

Q3) Based on *the documentation of the Servo library*, explain each line of the previous code.

.....

Q4) Modify the previous code to make the robot move forward for 4 seconds, and then move backwards for 10 seconds and finally stop to move (write your code here).

.....

Q5) Modify your code to make the robot move forward for 2 meters. Write the code and validate it with the professor. Then, test it on the ground.

.....

Optional questions:

Q6*) Show in a 2D graph the trajectory of the robot. Suppose that the initial point of the robot is (0,0). Show it to the professor.
What tool or software did you use to make the graph?

.....

Q7*) Same question as Q8, however this time the robot is not represented by a point, but by the shape of your robot.

.....

Get the orientation of the robot with an IMU

General information

The robot embeds an Inertial Measurement Unit (IMU) that will be used to determine its orientation while moving. The sensor is integrated into a shield as shown in Figure 4.

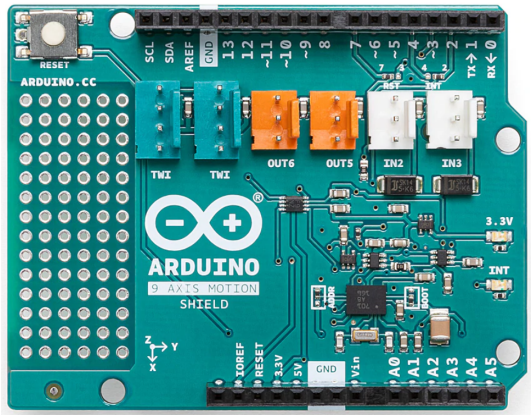


Figure 4: 9 axis motion shield

In order to communicate with the sensor, you need to install the required library as follow:

1. Open Arduino IDE
2. Click on Tools > Manage libraries...
3. After a few seconds, a new window appears. In the search zone, write **nine axes**. Download the library `Arduino_NineAxesMotion` as shown in Figure 5

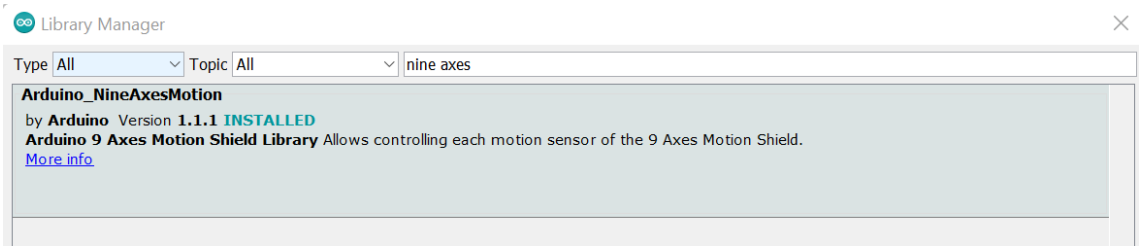


Figure 5: Library to install

Arduino programming

Make sure to download the required library before moving to this section!

Listing 5: Get the orientation of the robot

```
1  #include "Arduino_NineAxesMotion.h"         //Contains the bridge code
       between the API and the Arduino Environment
2  #include <Wire.h>
3
4  NineAxesMotion mySensor;          //Object that for the sensor
5  unsigned long lastStreamTime = 0;      //To store the last streamed time
       stamp
6  const int streamPeriod = 20;           //To stream at 50Hz without using
       additional timers (time period(ms) =1000/frequency(Hz))
7
8  void setup() //This code is executed once
9  {
10   //Peripheral Initialization
11   Serial.begin(9600);            //Initialize the Serial Port to view
       information on the Serial Monitor
12   Wire1.begin();                      //Initialize I2C communication to
       let the library communicate with the sensor (Wire1 for the Arduino
       DUE!!!)
13
14   mySensor.initSensor();  //Sensor Initialization
15   mySensor.setOperationMode(OPERATION_MODE_NDOF);   //9 degrees of
       Freedom Sensor Fusion
16   mySensor.setUpdateMode(MANUAL);  //The default is AUTO. Changing to
       MANUAL requires calling the relevant update functions prior to
       calling the read functions
17 }
18
19 void loop(){ //This code is looped forever
20   getRobotYaw();
21 }
22
```

```
23 void getRobotYaw(){
24   if ((millis() - lastStreamTime) >= streamPeriod){
25     lastStreamTime = millis();
26     mySensor.updateEuler();         //Update the Euler data into the
           structure of the object
27
28     Serial.print(" H: ");
29     Serial.print(mySensor.readEulerHeading()); //Heading data
30     Serial.println("deg ");
31   }
32 }
```

Q1) Write the following program using Arduino IDE. What can you see in the serial monitor ? Rotate manually the robot and observe the values displayed in the serial monitor. What can you conclude ?

.....

Q2) Modify the program to rotate the robot of 90 degrees (clockwise).

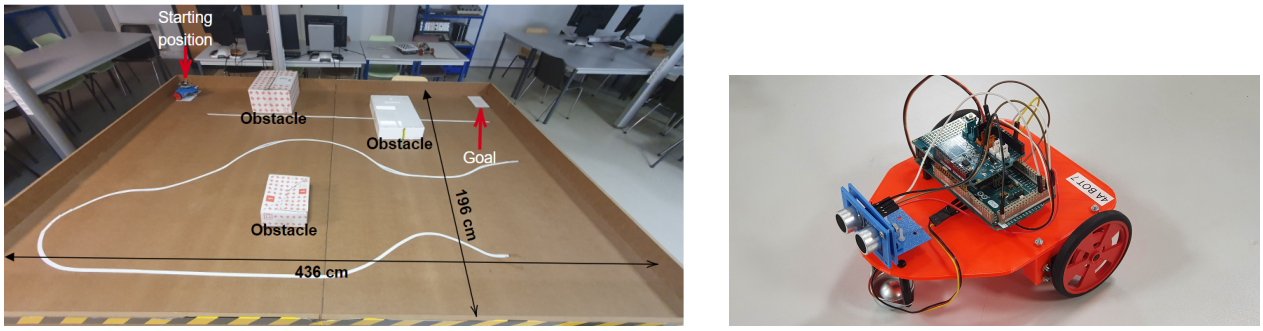
.....

PART II: Autonomous robot

In this part, you will implement your approach for the competition!

General information

In this section you will use the ultrasonic sensor to detect obstacles and program the adequate commands that allow the robot to avoid obstacles as a first step. You can draw inspiration from the previous sections. As a second step, you will need to program an algorithm to make the robot reach a destination « B » as shown in Figure 6.
The test environment is : 196 cm * 436.5 cm.



(a) Test environment (b) IPSA Bot

Figure 6: Test environment and IPSA Bot

Step1: Getting started

In this section, you will need to write the program that allows the robot to avoid obstacles. In your program, you will need to use information provided by the ultrasonic sensor and according to a specific distance to the obstacle, the robot will choose its move (backward, backward-left, backward-right). If there are no obstacles you can choose whether to move forward or make a random walk.
Write here the flowchart that describes your algorithm. Then, show the demonstration to the teacher.

Step2: Autonomous navigation

Now that you validated your approach, how can you modify your program to be able to reach destination « B » from a starting position « A » ? Write here your flowchart.

Good luck for the competition !

A2. SIMULATION USING ROBOT OPERATING SYSTEM (ROS)

Objective

The objective of this second part of the project is to move one or several simulated robots using ROS [2] from a starting position to a goal (represented by a green platform) while avoiding obstacles.
We define the robot as follow:

- An ultrasonic sensor is embedded, with a limited range (5 meters)
- 2 motorized wheels allow the robot to operate in the environment
- Its current pose in the environment is known (Position and orientation)

Background

Robot Operating System (ROS) is a set of software libraries and tools that help the programmer to build robot applications. In order to implement robot behaviours, ROS uses (1) nodes, (2) topics, (3) messages and (4) services.

1. nodes: A node is an executable that uses ROS to communicate with other nodes.
2. topics: Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.
3. messages: ROS data type used when subscribing or publishing to a topic or while using a service.
4. services: A client / server relationship between two nodes.

In this project, we use ROS1, where each node is linked with a master node. Therefore, it is a centralised structure.

Installation

The distribution used is ROS Noetic, based on Ubuntu 20.04. You will need to (1) Create a project using ROS Development Studio (RDS) and (2) download your project environment.

(1) Create your ROS project using ROS Development Studio (RDS)

Follow the specified instructions using the appendix A in order to start working with RDS.

(2) Download the project environment

Once you created your first ROS Project, you will need to download (clone) your environment using the instructions explained in `README.md` of our Github repository.

[2] `https://www.ros.org`

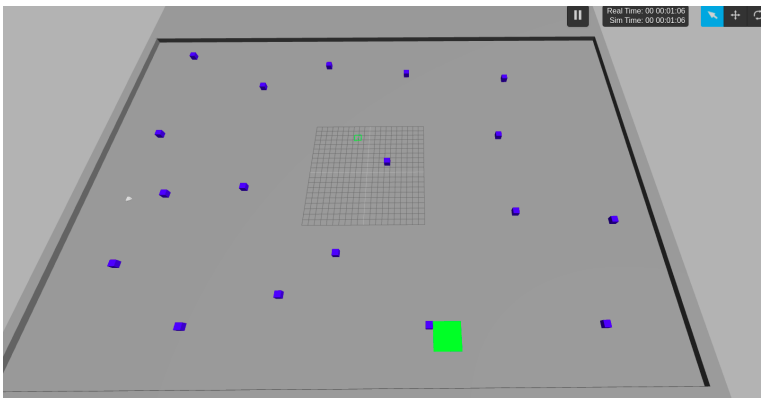


Figure 7: Simulated environment with one robot (white)

Getting started

The robot starts from a specific location and has to reach a green platform while avoiding obstacles from its environment.
The robot has a single input, its ultrasonic sensor, and two outputs (a linear and an angular velocity).

Below, you will follow 7 steps (**From step 1 to step 7**) in order to work with **one** robot.

Step 1: Run the simulation and the strategy node

To run the simulation, open a terminal and run the following instruction:

```
roslaunch project_in424_description simu.launch nbr_robot:=1
```

You do not see anything, it is normal. **To visualize the Gazebo environment, you need to click on the button *Open Gazebo*.**
The simulated environment will be then displayed on Gazebo as shown in figure 7.
Now run, in a second terminal, the python script that will contain your strategy:

```
roslaunch project_in424_navigation agent.launch name:=robot_1
```

In this second terminal, you should see data coming from the ultrasonic sensor displayed in the related terminal (SONAR VALUE = 5.00).

Now that you have seen the data coming from the ultrasonic sensor, you can stop the running process in the **second terminal** by pressing `CTRL + C`.

Step 2: Playing with ROS

In the second terminal, execute the following instructions (**the first terminal which is running the simulation under Gazebo should not be stopped!**)

1) List the topics :

To list the current topics, enter this instruction:

```
rostopic list
```

Q1) What is the result?

.....

2) Information about the sonar topic :

```
rostopic info /robot_1/sensor/sonar_front
```

Q2) Who is the publisher of this topic?

.....

Q3) What type of messages are published on this topic?

.....

3) Listening to a topic :

```
rostopic echo /robot_1/sensor/sonar_front
```

Q4) What is the result of the sonar?

.....

The next sections require you to edit the python script.

Step 3 : Move the robot

Your strategy will be implemented in the python script **my_agent.py** located at the following path:

`/catkin_ws/src/IN424/project_in424_navigation/scripts/my_agent.py`

In order to define your strategy, you need to edit **my_agent.py** and program using python. Figure 8 shows how to access the file **my_agent.py** through the `Code editor`.

my_agent.py is a python script that contains a set of instructions that allows you to get the data coming from the ultrasonic sensor. Moreover, you can publish velocity commands to the topic **/robot_1/cmd_vel** by modifying the values of the variables **linear_velocity** and **angular_velocity**. These velocities are respectively limited to [-2.0, 2.0] and [-1.0, 1.0].

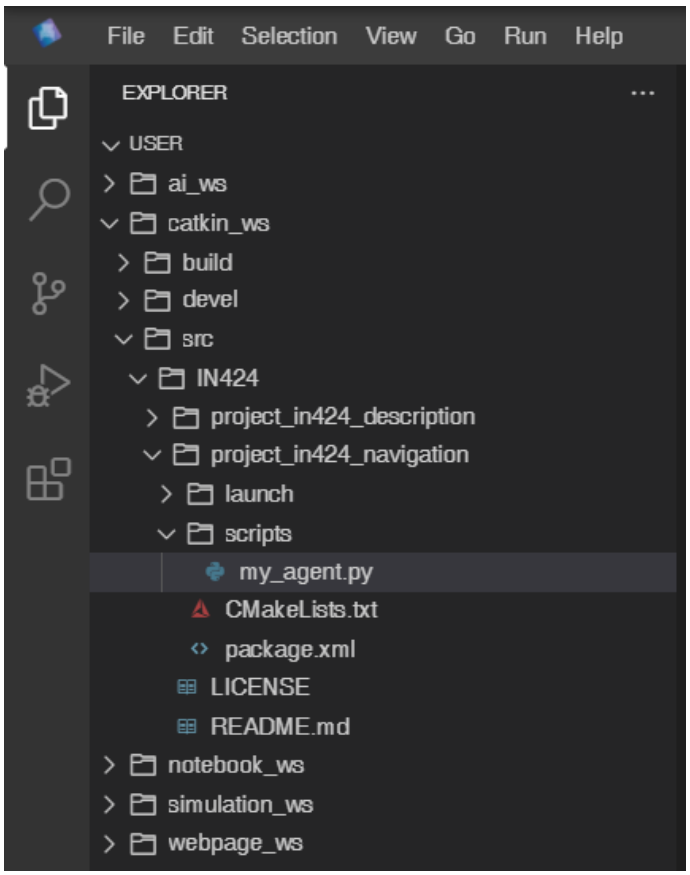


Figure 8: How to edit the file my_agent.py

Q5) Modify the python program in order to see the robot move. At line 75 change the value of the desired linear velocity.

.....

Q6) Launch the modified program using `roslaunch` as follow:

```
roslaunch project_in424_navigation agent.launch name:=robot_1
```

After seeing the robot displacement in the environment, you can stop the simulation. Stopping the simulation and the python script means pressing *CTRL + C* in the first and second terminals.

Now let's work by yourself !!

Step 4 : Getting the current position of the robot

1. In the first terminal, run the simulation:

```
roslaunch project_in424_description simu.launch nbr_robot:=1
```

2. In a second terminal, list the available topics and identify which one provides the position of the robot.

3. Listen to this topic and verify if it is the right one.
4. Identify the type of messages published on this topic.
5. To get the position of the robot in your program (**my_agent.py**), find out how to subscribe to the adequate topic. (**Tip:** you can check how the ultrasonic sensor message type has been imported and how the subscriber has been created).
6. Add a subscriber and a callback function to retrieve the current position and the orientation of the robot and store it into **new** attributes of the class *Robot*.

Step 5 : Avoid obstacles

Modify your initial program **my_agent.py** in order to avoid obstacles. **Tip:** By coupling the current position of the robot to the data coming from the ultrasonic sensor, implement an obstacle avoidance strategy.

Step 6: Get the coordinates of the goal position

In this section, you need to find out how you can get the coordinates of the goal position (green platform). In order to get its coordinates, you can find it in the list of the parameters using the following instruction.

1. List the available parameters:

```
rosparam list
```

2. Identify which ones provide the coordinates of the center of the green platform.
3. Add some instructions in the python program to automatically get these coordinates in the constructor of the class Robot (see this link for more details).

Step 7 : Autonomous robot

Based on the previous steps, the robot must fulfill its mission: move from its initial position to reach the green platform (goal position) while avoiding the obstacles. The goal should be reached as quick as possible.
Your solution should not depend on this fixed environment, which means you need to implement a robust solution. During the final evaluation, the robots will operate in a more complex environment to evaluate the robustness of your approach.

Step 8 : Multi-agent exploration

For this final step, you will have to implement a solution that moves 3 robots at the same time to the platform.

1. Open a terminal, and launch the simulation as follow:

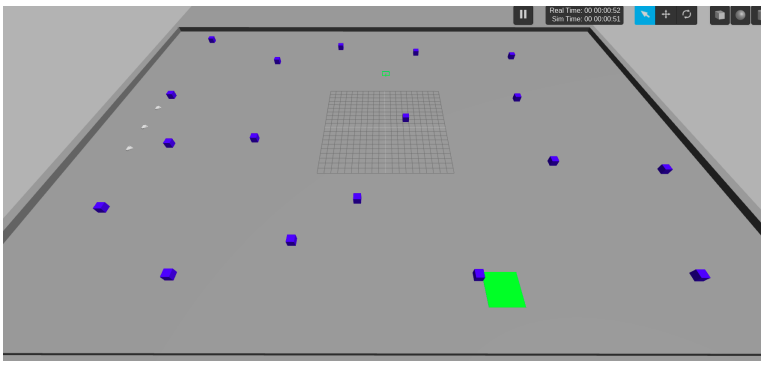


Figure 9: Simulated environment with three robots (white)

```
roslaunch project_in424_description simu.launch nbr_robot:=3
```

Three robots should appear in the environment as shown in Figure 9.

2. Open 3 other terminals and enter respectively:

   - `roslaunch project_in424_navigation agent.launch name:=robot_1`
   - `roslaunch project_in424_navigation agent.launch name:=robot_2`
   - `roslaunch project_in424_navigation agent.launch name:=robot_3`

3. Improve your solution at step 7 so that the robots reach the platform without colliding with each other.

Configuration of ROS Development Studio (RDS)

ROS Development Studio

JOHVANY GUSTAVE

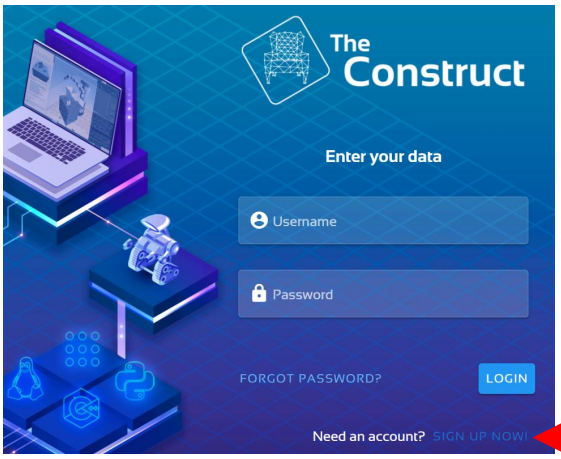
RDS

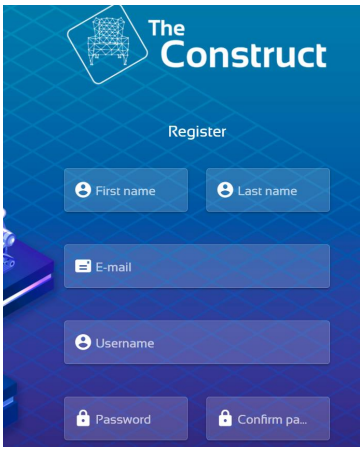
- Online tool allowing users to communicate with a powerful computer that runs ROS
- Easy to use, intuitive tool
- Accessible from any computer that has internet access (It is recommended to use Google Chrome)
- Free plan:
  - Creation of 2GB of projects (Rosjects)
  - You can work for **8 hours / day** on a Rosject

Create an account

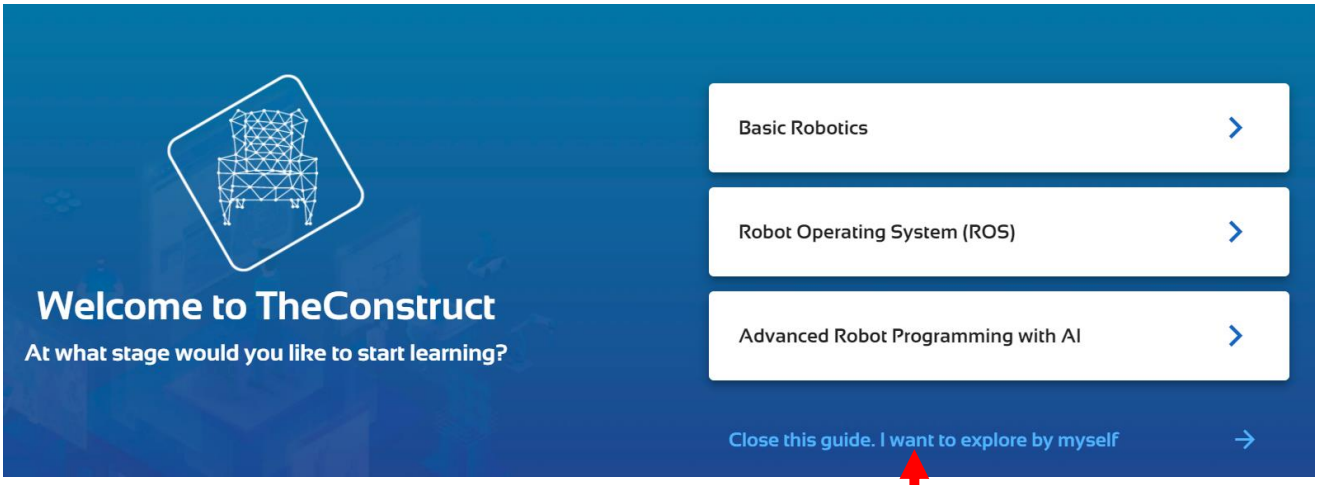
- Open your favorite web browser and go to the following address:

https://app.theconstructsim.com/#/

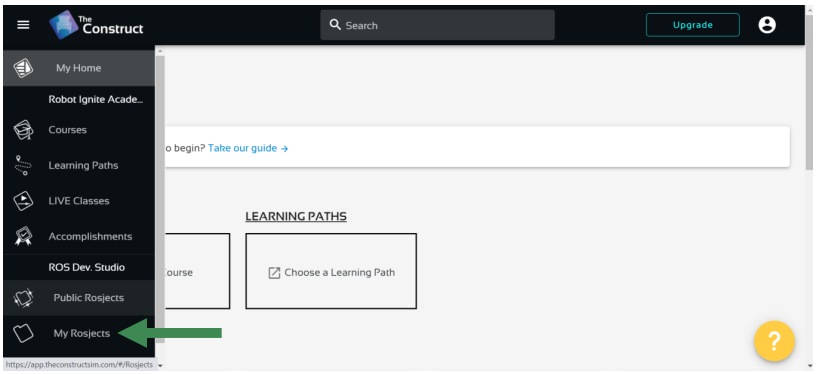


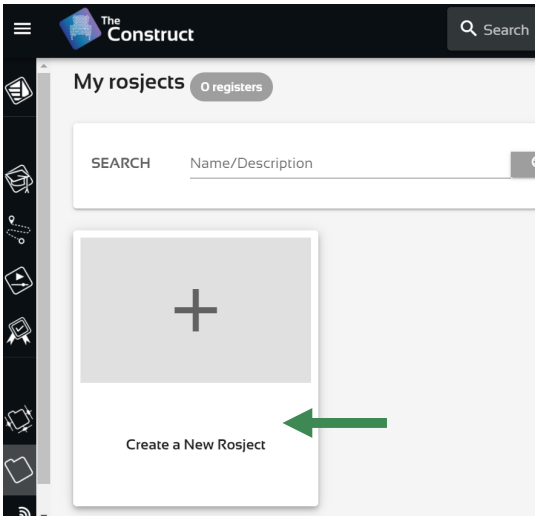


Complete the creation of your account

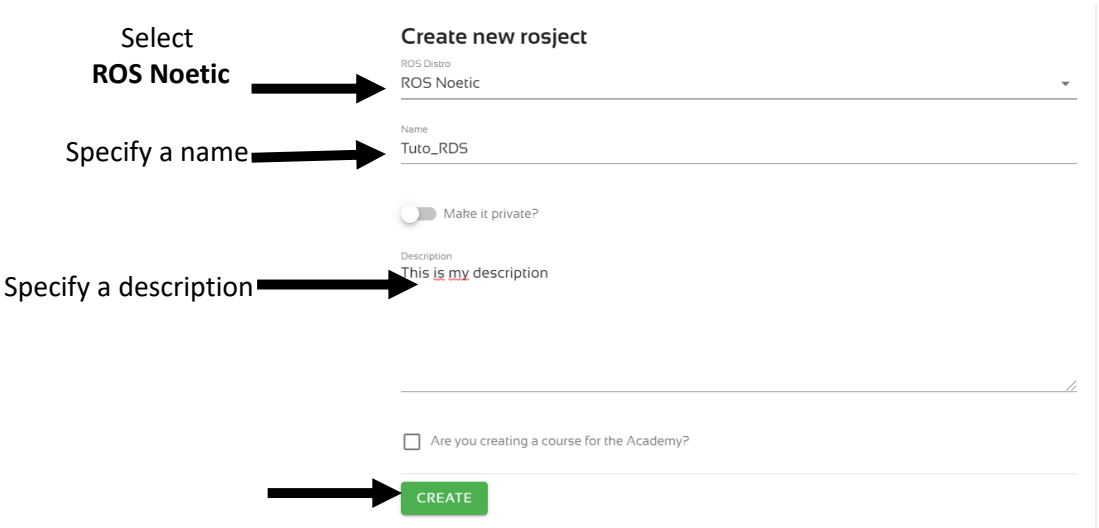


Create a Rosject

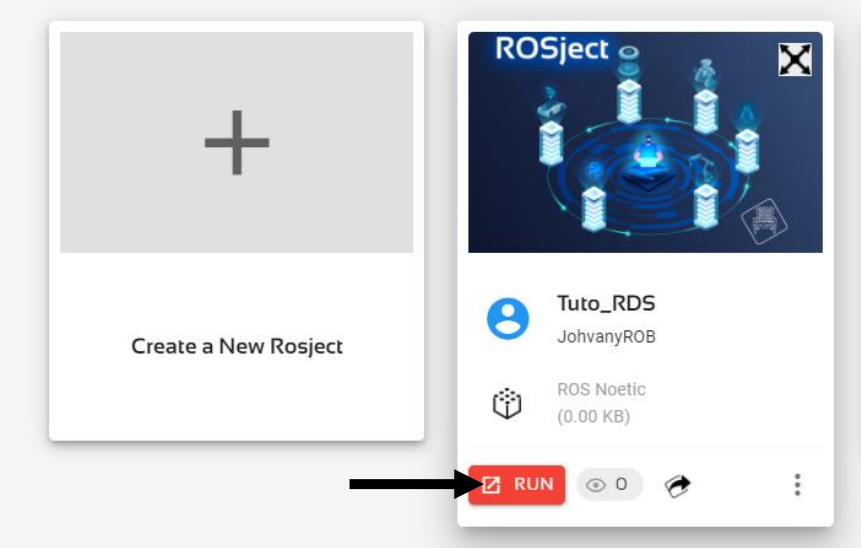




Configure the Rosject



Run the Rosject



Quick description of the tools