EXAMINATION PERIOD:	Coursework, first semester, ay 2022/23
MODULE CODE:	5CCS2ITR
TITLE OF EXAMINATION:	Introduction to Robotics – second coursework
FORMAT OF EXAMINATION:	Online submission
SUBMISSION DEADLINE:	Friday 9/12/2022 at 4pm

IMPLEMENT THE SPECIFICATION IN ALL SECTIONS

SUBMISSION PROCESS: Your work must be submitted as a zip file containing the full package second_coursework with your code.

Ensure you upload the correct file to the submission folder

Background Story

Our robot, a TurtleBot3, is working in a store doing the inventory of items overnight. It must go around the building noting the items it sees. The owner is interested in all items, but in one above all: **cake**. The inventory robot must navigate on a specific pattern while reporting the objects that it sees, until it finds **cake**.

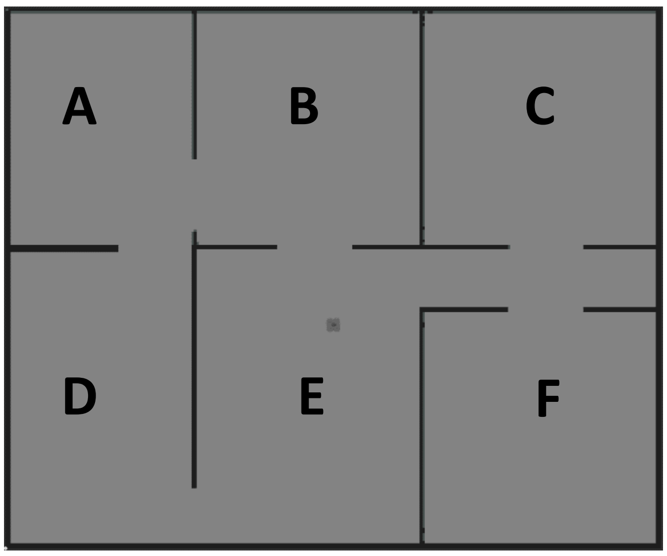
Preparation

- Create a package called "second_coursework"

Total of Section: 2 marks

Ros Fundamentals

Create a node that implements a service that given a *string* with a room name returns a coordinate in that room. Every time the service is called, it returns another coordinate in the same room.



- Create a service called GetRoomCoord that, given a room name, as a *string*, returns a point inside the room. You can use any representation for the point (a predefined message, such as those in geometry_msgs, or your own definition). **(10 marks)**
- Create a node called roomservice that implements the service specified above. Every time the service is called with a room name, it must return the next coordinate inside the room from a list of 4 coordinates. The service must have 4 coordinates of your choice per room, and return them in order, starting again from the first one after having returned the fourth. More precisely: if at the previous call the service returned the n-th coordinate for a room, in the next call with the same room it must return the (n+1)%4 coordinate in the list. **(15 marks)**

Total of Section: 25 marks

ROS Action

Create a node that implements an *action server* to run the search behaviour.

- Define an action message called *Search*, with fields:
    - Request: a *string* containing the name of a room for which the robot has to make the inventory.
    - Feedback: a list of *strings* and a list of *integers*.
    - Result: a list of *strings*, a list of *integers*, and a time stamp (obtained with `rospy.Time.now()`).
- Create a node called main_node.
- The main_node should contain a class implementing the server of the action above. The behaviour of the action must be represented as a state machine, as specified in the next section.

Total of Section: 13 marks

Robot Search

The robot behaviour must be represented as a *state machine* using SMACH. The state machine must be run inside the action server (so a call to the action will start the state machine). The state machine must implement the following behaviour:

- The robot starts to move within the room specified in the action request. The robot should go to all the coordinates of the room returned by the GetRoomCoord service you implemented **(10 marks)**.
- In a SMACH state class: **While** moving around the room, the robot must process the camera feed to recognise objects. Use a concurrence container to make the robot move while recognising objects. Use a variant of the YOLO detector that subscribes to the camera and keeps running YOLO every 20 frames to recognise objects. The images must be obtained from the provided ROS bags (see Test Videos) **(20 marks)**.
- The action must send feedback at a rate of 5 Hertz. The feedback should include the list of objects recognised until that point, and how many items of each object (for instance, there may be three bottles) **(5 marks)**.
- When the robot sees a **cake**, the robot should stop moving and the action server return, as a result, the final list of recognised objects, the count of each object, and the time when it stopped (saw the **cake**). You can get the current time as `timestamp = rospy.Time.now()` **(5 marks)**.

Total of Section: 40 marks

Desiderata

- Structuring the code in different State classes (ideally in different files and importing them into the main node) **(5 marks)**.
- Using SMACH states such as SimpleActionState or CBState **(5 marks)**.

Total of Section: 10 marks
Grand Total: 90 marks

Test videos

To facilitate the development of the coursework without access to the real robots or a webcam, we provide videos in the form of ROS bags. The videos were taken from a real robot's camera, showing the different objects. Objects are displayed in the video for at least 3 seconds.

Download the ROS bags from KEATS at the same section as this brief. There are three bags for your development. We have a fourth bag that we will use for testing in addition to the three provided, in order to avoid implementations hardcoded to the particular videos.

Note that the camera topic in the robot (and therefore, in the ROS bags) on which the images are published is "`/camera/image`" instead of the "`/usb_cam/image_raw`" used in the lectures for the webcam!