Task 4 - Maze Solving using the Robot

Objective:

The objective of this task is to make your Robot rescue victims from **Firezones** based on the number of vacancies in one of the **Hospitals** and drop them there.

Given:

1. **Maze Image with 4 Firezones**
2. **Number of Vacancies in Hospital (X): 4**
3. **Python List of all Digits in the image:**

   digits_list = [8, 0, 2, 2]

   This information is already being used in the **task_4.py** script. There is no need for your script to identify the digits from the maze image.

4. **Python Dictionary having combination of digits for sum with their locations:**

   combination = {2: (6, 9), 2: (8, 6)}

   This information is already being used in the **task_4.py** script. Since, such kind of dictionary cannot be declared in Python where we have duplicate entries of keys. Hence, we have created a function in **task_4.py** named **create_combination_dict()**. Refer to Description_task_4_py.pdf to know more about the script.

Arena Preparation:

In order to start with the task, first arrange the walls in the following manner, starting from **NEW_START (4, 4)** and ending at **EXIT (9, 9)** as shown in Figure 1.

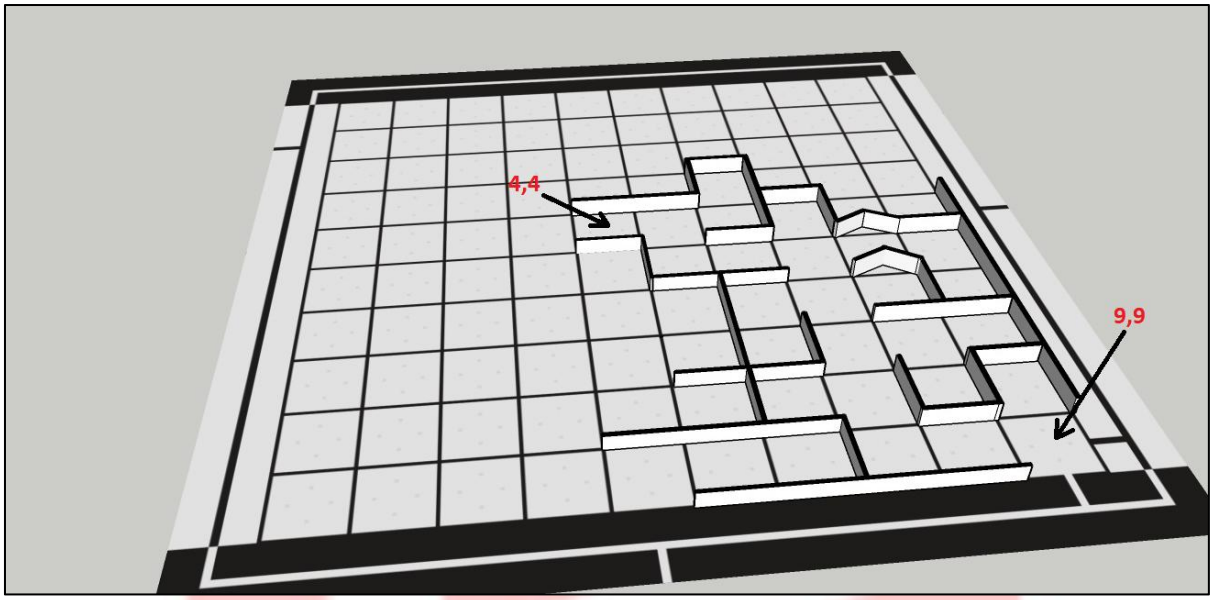

Figure 1: Walls placement for Task 4

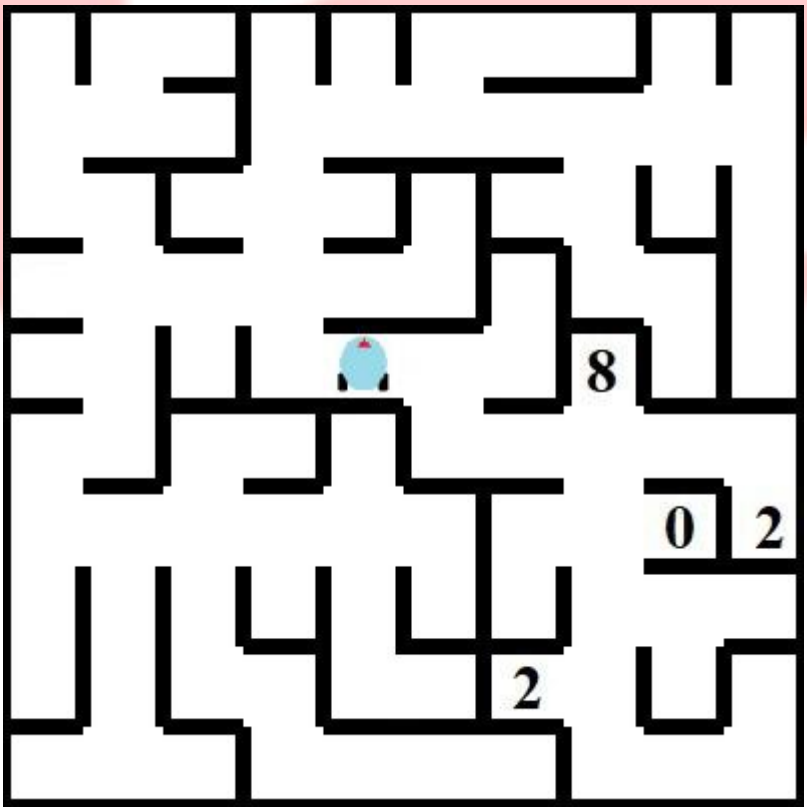

Figure 2: Maze image with Firezones and Robot

Source Code Preparation:

For Task 4, you need to modify/write following four programs,

1. Python Client script (**task_4.py**)
2. **task_1a.py** (same as Task 1A of Stage 1)
3. ESP32 Robot Server Application (**robot-t4-esp32.c**)
4. ATmega2560 Robot Application (**robot-t4-atmega2560.c**)

The Robot is supposed to,

- Start Wireless Access Point.
- Receive the **digits_list** and **combination** information from the Python Client script.
- Receive coordinates of shortest path to traverse the arena as per the Problem Statement described below.

The Python Client script is supposed to,

- Establish socket communication with Robot.
- First communicate the **digits_list** and **combination** to the Robot.
- Take the maze image as input.
- Find the shortest path between **NEW_START** and the respective Firezones to be visited and **EXIT** and each time communicate all the coordinates to the Robot wirelessly using socket.
- The template of the script is already given as **task_4.py**. Refer to Description_task_4_py.pdf to know more about the script.

Problem Statement:

In this task, your Python Client script (**task_4.py**) and your Robot should exchange necessary information to make the Robot traverse through the maze visiting desired **Firezones**, rescuing trapped victims from there and then dropping them at the nearest **Hospital**.

Following steps need to be completed in order to successfully complete Task 4:

- The robot should be kept at **NEW_START** position at center within the cell **(4, 4)** as shown in Figure 2. Direction of the robot can be decided by the respective team.
- Turn **ON** the Robot and Connect your PC/laptop to the Wireless Access Point of your Robot.
- Start **Task4_GUI_App** by running the command:

  " **sudo python3 Task4_GUI_App.py** "

  which will monitor the communication between the Robot and your Python Client script, while simultaneously grading the run. Refer Task4_GUI_Description.pdf for more details. Click **Start** button on GUI.
- Once your PC/laptop is connected to the Robot, start your Python Client script for Task 4 by entering the following command in the Terminal,

  python task_4.py

  NOTE: Make sure that your Conda environment is activated.
- Teams must communicate this information: **digits_list** and **combination** (which is also already given for this Task) wirelessly from the Python script to the robot using socket.
- The Python Client should take the provided maze image as input and find the shortest path between the **NEW_START** and first Firezone planned to be visited.
- Make sure that while sending the above information from Python Client to the robot, it is sent in proper format: **#<data from client to server>#**.
- Similarly, while receiving the data from ESP32's server on robot to Python Client, it is received in proper format: **@<data from server to client>@**.
- To get the shortest path, **task_1a.solveMaze()** should be used. The **task_1a.py** present inside the **Codes** folder is the skeleton file provided in Task 1A of Stage 1. Team is expected to either replace the **task_1a.py** file with the one submitted during Stage 1 or complete the **readImage()** and **solveMaze()** functions in the one provided now.

- On pressing **USER_SW** switch on the robot, **@started@** should be communicated by it to the Python Client (**Note:** this is not mentioned in the Rulebook but has to be followed for all the future Tasks and for the Finals as well) to start the traversal of the robot. The **Task4_GUI_App** will start the Timer at its backend. This will be considered as the start of a run. Once the run in started, human intervention is NOT allowed.

- Note that the robot waits at the **NEW_START** position until the **USER_SW** switch is pressed.

- Robot should follow the path from **NEW_START** to respective Firezone/s and then to **EXIT**.

- Robot has to visit all the **Firezones** depending on the combination which is already given for the Task. On reaching each Firezone, robot **MUST** face towards the Firezone (refer Figure 12a and 12b in Rulebook), glow **Red** LED for 1sec and communicate that "**n**" victims are evacuated from **Firezone** with cell co-ordinates **(x,y)** in the format as specified below.

On reaching a **Firezone**, communication from robot to client should be done in following format:

Cell co-ordinates of **Firezone** and number of victims evacuated = **@$|n|(x,y)@**

("**$**" indicates that **(x,y)** are the co-ordinates of the **respective Firezone** and "**n**" indicates the digit in that Firezone)

- Once done, repeat the process till all the planned **Firezones** are visited.

- Once the robot reaches **EXIT**, it should enter into the Outer area and visit **one** Hospital. Based on the **EXIT** position chosen by the robot (refer Figure 10b in the Rulebook), the Hospital reached by the robot will be named as **HA**.

- On reaching the hospital, robot MUST face towards the hospital, glow **Green** LED for 1sec and communicate the following to client, respectively:

  @HA reached, Task accomplished!@

- Once the **@HA reached, Task accomplished!@** is communicated, the **Task4_GUI_App** will stop its Timer and note down the Total Time of the run at its backend. This indicates the end of run.

- Now press **Calculate** button and then **Exit** button. On pressing the **Exit** button on the GUI, an encoded text file: **task_4_output.txt** file will be generated with all the parameters including the Total Time of Run and Team ID.

Video:

- Once you are done practicing and are ready with complete run of Task 4 as mentioned in above steps, you have to record a demonstration video.
- Following steps should be taken to record the demonstration video:
  - The robot should be kept at **NEW_START** position at center within the **(4, 4)** cell as shown in Figure 2. Direction of the robot can be decided by the respective team. Turn **ON** the Robot and Connect your PC/laptop to the Wireless Access Point of your Robot.
  - Run the **Task4_GUI_App** and focus your camera on your laptop screen so that GUI is clearly visible. **There is no need to use any screen-recoder app for the demonstration video.**
  - While your camera is focused on your laptop screen enter your **Team ID** in the GUI. After entering your Team ID, Screen 2 (Figure 3 of Task4_GUI_Description.pdf) will open. Here you have to press the **Start** button to start the grading process.
  - Run Python Client script for Task 4, using command: **python task_4.py** (as stated above). The GUI will get updated when **digits_list** and **combination** are sent from Python Client to the ESP32's server. You have to then click **Calculate** button on the GUI. The complete GUI should be visible while you do this, hold it for few seconds.
  - Then you have to focus your camera on the arena. Make sure that all the **Firezones** and the **Hospital** are visible in your frame. Once this is done, start the run as described in the **Problem Statement** section above.
  - After your run is complete, again focus your camera on the laptop screen where only the GUI is visible. Now, after the run is completed, according to whether your robot was able to visit the **planned Firezones** and reach the **Hospital**, you have to move the respective **Slider** towards **right**. You should do this only if your robot after reaching the **Firezone/s** and **Hospital** was able to glow the respective colored LED and appropriate communication was done to the Python Client script.
  - Now, press the **Calculate** button. (**Note:** No unnecessary inputs should be added. If the glowing of LED is not visible and still Team moves the Slider, your run will not be considered and no marks will be awarded. Also, if you communicate **@HA reached, Task accomplished!@** without reaching to the Hospital, Maximum Time will be considered for evaluation).
- Make sure you use camera of resolution **more than 5 MP** to record the video.

- While recording the video, one team member may explain the entire implementation of Task 4. **Maximum time limit of video: 4 minutes**.
- Upload the video on YouTube as **Unlisted** and provide the link on the portal under the **Task 4** page.
- Instructions for submitting the video link are mentioned on the portal.

For this Task 4, you have to submit the following:

A. **task_4.py** and **task_1a.py** script with all the necessary modifications.

B. Complete project folder of ATmega2560 application created in eY-IDE with the source file named: **robot-t4-atmega2560.c** and its compiled binary files inside the ***build*** folder. The name of the project folder will be same as the source file named as: **robot-t4-atmega2560**.

C. The source file of ESP32 Robot Server application created in eY-IDE named as: **robot-t4-esp32.c**.

Note: You are not supposed to provide any compiled binary files of ESP32 Robot Server application, **only the source file** needs to be submitted.

D. An encoded text file (**task_4_output.txt**) generated after running the **Task4_GUI_App**.

E. An Unlisted YouTube video

Kindly refer to the Task_Submission_Instructions.pdf before uploading the Task on portal.

After completing all the steps successfully, Task 4 will be considered as complete.

NOTE: Hard coding in any of the steps described in the Problem Statement is not acceptable.