Task 4.1 - Survey & Rescue

1 Problem Statement

1. Aim of this task is to produce a script, that should be able to detect & locate colours present in any cell and at any time after the run starts as per rules mentioned in Rulebook (refer section 7 of Rulebook for more information).

2. This script should also create and publish the detected colours [type of event (refer section 2.1.4 of Rulebook)] and it's location on the topic *detection_info* with specific msg type and format (refer 1st point from Points To Remember to know more formatting).

2 Procedure

1. For detecting the colours, as hints a boilerplate script has been added to the survey_and_rescue package on GitHub. You will need to fetch and merge it. Follow the steps below to do so.

   NOTE: After following the below steps, there can be a merge conflict, if so, you will have to resolve them, outcome have some additional lines in the merged scripts. Please go through these files using an editor which has a diff tool functionality, i.e. which shows you which lines have been changed in the versions. Another way to do this is to open the survey_and_rescue folder using sublime-text editor before pulling the new content. After the merge command, the changed files will be highlighted in the editor's window. Remove the conflict markers if there is a conflict to bring back the file to a fully operational status.

   (a) Open a terminal and change directory to the survey_and_rescue package by typing the following command.

   ```
   cd ~/catkin_ws/src/survey_and_rescue
   ```

   (b) If you have not already committed your local changes yet, follow this link to do so. In summary you have to run the following commands:

      i. To know the files you have changed and added

      ```
      git status
      ```

      ii. To add all these files to be tracked.

      ```
      git add .
      ```

      iii. To commit (log) these changes with an optional message.

      ```
      git commit -m "My local changes in Task 3"
      ```

      iv. Update the survey_and_rescue package by typing the following line. This will both update the package and resolve the merge conflicts if they exist.

      ```
      git pull origin master
      ```

2. After updating the package, Points To Remember) in the 6x6 grid of the Arena. To do so, fill in the boilerplate script. To make this process robust, make use to the RoIs saved from from Task 3.3's output(if utilised).

3. Make sure that roscore and usb_cam_SR.launch has been initialized before proceeding.

4. Run the completed *Beacon Detector* python script (please change the name of the completed script from beacon_detector_boilerplate.py to beacon_detector.py). This can be done by either running.

```
rosrun survey_and_rescue beacon_detector.py
```

5. Now to detect the Beacons, next step would be glow them. This is taken care by the ***monitor.pyc*** script (refer 3rd point of Points To Remember to know more about this important script).

6. Follow the step below to glow the LEDs.

   (a) Make sure that you have burned the Hex file provided, into the Arduino Nano. Please refer Installation Instruction and Tips PDF of Task 4 for help regarding burning code and installing neccessary packages.

   (b) Make connection between Arduino Nano, LED modules and your system.

   (c) Start rosserial node by running:

   ```
   rosrun rosserial_python serial_node.py _port:=/dev/ttyUSB0 _baud:=115200
   ```

   (d) Run the following command.

   ```
   rosrun survey_and_rescue monitor.pyc
   ```

7. We have provided an alternative as well, instead of running 3 nodes separately. All of these nodes can be initialized by detect_beacons.launch file.

   ```
   roslaunch survey_and_rescue detect_beacons.launch record:=False
   ```

   This launch file (detect_beacons) will cover communication (rosserial node), monitor.pyc script and if the *record* param is set as *True*, it will also initalize the recording with following functionality...

   - This will record a bag file (titled SR_detection.bag by default) for a duration of 25 seconds of wall-clock time.
   - Recording will be done of /stats_sr and /detection_info topics for the mention duration.
   - This bag will be stored in the survey_and_rescue/bag_files folder.

3 Expected Output

- Refer to this video to know more regarding *Beacon Detections* and expected results.

4 Points to Remember

1. Name of type of Services:

Sr.No	Colour Detected	Service Type	Message string
1	Red	Rescue Event	RESCUE
2	Blue	Medicine Event	MEDICINE
3	Green	Food Event	FOOD

2. By locations, we mean the name of the cells. Example location can be A1, A6, F4, D2, etc.

3. The script monitor.pyc is a compiled python script which has the following usage:

   (a) To glow LEDs using the rosserial node which enables communication with Arduino Nano.

   (b) To provide feedback whether Beacon detected is correct or not.

   (c) To keep a track of the Rescue Drone's services(refer section 7.5 from Rulebook) and provide feedback on topic *serviced_info* about the successful or failed service status. Teams should use this info as feedback and make decision in scripts produce in Tasks.

   (d) Please refer video for better understanding of *monitor.pyc* script.

   (e) To keep track of the score as per the Rulebook.

5 Submission Instructions

Follow the instructions below to submit your Task.

5.1 Bag File:

- Next, after completing the *Beacons Detection* python script, when you want to record the bag file for submission, run the following launch file.

```
roslaunch survey_and_rescue detect_beacons.launch record:=True
```

  WARNING:

  - For recording/creating the bag file, use ONLY detect_beacons launch file.
  - Do not change the *LED_Timing.tsv* file.

- Before submitting make sure to verify your bag file, by Check the number of messages, which should be non-zero. By running the following commands.

```
rosbag info SR_detection.bag
```

- Further verification can be done by playing the rosbag itself.

```
rosbag play SR_detection.bag
```

  While echoing those 2 topics in other terminal. One could manually check the messages being published.

  NOTE:
  Repeated running of this launch file will overwrite the SR_detection.bag file, please make sure you have a backup if you so wish. Alternatively, you can use the argument rec_name param have a custom name while executing the detect_beacons launch file.

5.2 Python Script:

- You must submit your *Beacon Detector* script that you developed.
- Rename the python script as SR_<team_id>_4_1.py

Store both the files (bag as well as python script) mentioned above into .zip file and rename the zip as <team_id>

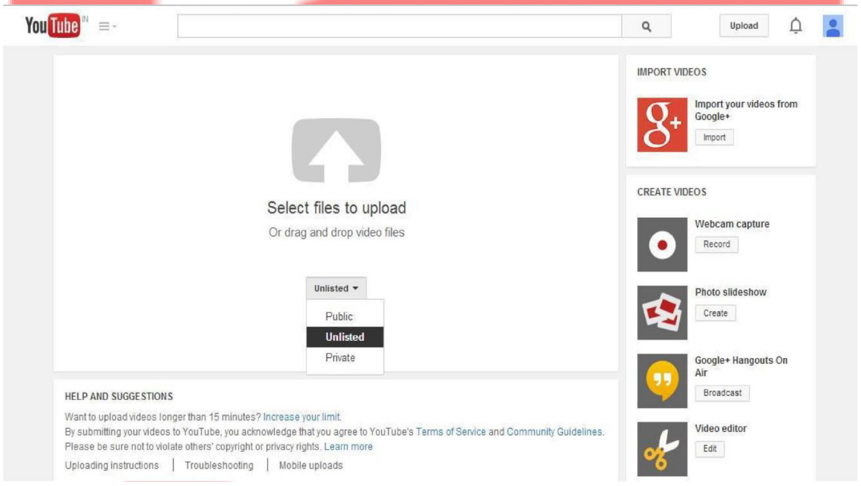
5.3 Video:

- Upon verifying that your task is complete, record a maximum 2-minute video using a screen recorder like simplescreen recorder or kazam.
- The video must be as follows:
  1. Screen Recording starts
  2. Team Slide –All member's details in a slide, included in this folder.
  3. Any One member of the team, running the necessary scripts or launch files in terminal.
  4. After the script begins, make fullscreen the */whycon/image_out* window.
- Please refer this video a sample recorded video.

The video should not be edited in any manner, except as mention in the instructions. Teams uploading an edited video will be disqualified from the competition. **e-Yantra reserves the rights to disqualify any team if any foul play is suspected.**

5.3.1 Uploading video/s on YouTube:

- Upload a one-shot continuous video with the title eYRC#SR#Task4_1#<TeamID>
  (For example: If your team ID is 1234 then, save it as eYRC#SR#Task4_1#1234)
- Please note that while uploading the video on YouTube select the privacy setting option as Unlisted as shown in Figure 1. You need to upload the video as instructed on the portal.



Want to upload videos longer than 15 minutes? Increase your limit.
By submitting your videos to YouTube, you acknowledge that you agree to YouTube's Terms of Service and Community Guidelines.
Please be sure not to violate others' copyright or privacy rights. Learn more

Figure 1: Uploading unlisted video on YouTube

5.4 Overview:

You must upload the following files:

- bag file
- Python code

Please place these files inside a .zip file before uploading. You must upload the video to YouTube as mentioned in the above instructions. You will have to submit the video link on the portal.
Please follow the naming convention strictly as specified in each step. Failure to do so may lead to repercussions. The deadline for Task_4.1 is till midnight of 30th January, 2020.
Your final .zip output must be of the following structure:
<team_id>_4_1.zip

- SR_<team_id>_4_1.bag
- SR_<team_id>_4_1.py

Instructions for uploading the folder will be provided on portal.
Good Luck!!!