

Abhiyaan

Software Application

Srikar Babu Gadipudi, EE21B138

April 20, 2022

# Contents

<b>1</b>	<b>Details</b>	<b>1</b>
<b>2</b>	<b>Section A</b>	<b>1</b>
2.1	Task 1 . . . . .	1
2.2	Task 2 . . . . .	1
2.3	Task 3 . . . . .	3
<b>3</b>	<b>Section B</b>	<b>4</b>
3.1	Subsection B 1 . . . . .	4
3.2	Subsection B 2 . . . . .	5
3.3	Subsection B 3 . . . . .	5
<b>4</b>	<b>Acknowledgements</b>	<b>7</b>

# 1 Details

**Name:**

Srikar Babu Gadipudi

**Roll no**

EE21B138

**Previous Experience**

Have a fair bit of learning on coding. Very into competitive programming (even though not very good at it).

**Current PORs:**

Part of Avanthi team

**Why I want to work in the team:**

The idea of driver-less cars is very fascinating. This is the area of technology which is still under incubation and needs time to get into large scale. I want learn as much as I can and contribute my views on this fresh concept.

**Relevant Courses:**In Institute

EE1103 Numerical Methods

S

Outside

CS50 Harvard's computer science course

Mostly learned coding out of curiosity

## 2 Section A

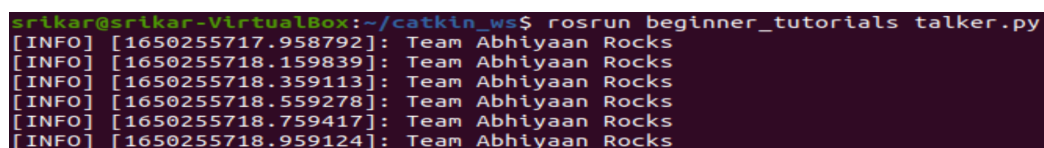
### 2.1 Task 1

Installed virtual box and ROS Noetic.

### 2.2 Task 2

Created two nodes (publisher and subscriber) over the topic `/team_abhiyaan`.

To run these codes use the command `roslaunch beginner_tutorials _____.py`.



```
srikar@srikar-VirtualBox:~/catkin_ws$ roslaunch beginner_tutorials talker.py
[INFO] [1650255717.958792]: Team Abhiyaan Rocks
[INFO] [1650255718.159839]: Team Abhiyaan Rocks
[INFO] [1650255718.359113]: Team Abhiyaan Rocks
[INFO] [1650255718.559278]: Team Abhiyaan Rocks
[INFO] [1650255718.759417]: Team Abhiyaan Rocks
[INFO] [1650255718.959124]: Team Abhiyaan Rocks
```

Figure 1: Output for the publisher

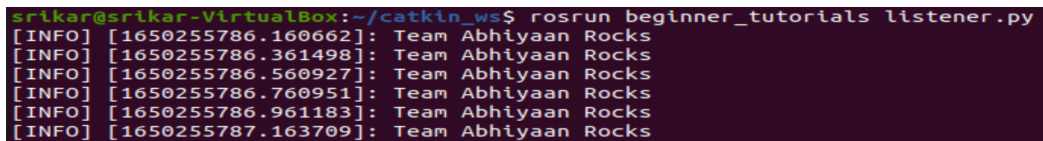
Listing 1: Publisher node

```
1 #!/usr/bin/env python
2
3 import rospy
```

```

4 from std_msgs.msg import String
5
6 def talker():
7     pub = rospy.Publisher('team_abhiyaan', String, queue_size=10)
8     rospy.init_node('talker', anonymous=True)
9     rate = rospy.Rate(5) # 10hz
10    while not rospy.is_shutdown():
11        hello_str = "Team Abhiyaan Rocks"
12        rospy.loginfo(hello_str)
13        pub.publish(hello_str)
14        rate.sleep()
15
16 if __name__ == '__main__':
17     try:
18         talker()
19     except rospy.ROSInterruptException:
20         pass

```



```

srikar@srikar-VirtualBox:~/catkin_ws$ rosrn beginner_tutorials listener.py
[INFO] [1650255786.160662]: Team Abhiyaan Rocks
[INFO] [1650255786.361498]: Team Abhiyaan Rocks
[INFO] [1650255786.560927]: Team Abhiyaan Rocks
[INFO] [1650255786.760951]: Team Abhiyaan Rocks
[INFO] [1650255786.961183]: Team Abhiyaan Rocks
[INFO] [1650255787.163709]: Team Abhiyaan Rocks

```

Figure 2: Output for the subscriber

Listing 2: Subscriber node

```

1 #!/usr/bin/env python
2
3
4 import rospy
5 from std_msgs.msg import String
6
7 def callback(data):
8     rospy.loginfo('%s', data.data)
9
10 def listener():
11
12     rospy.init_node('listener', anonymous=True)
13
14     rospy.Subscriber('team_abhiyaan', String, callback)
15
16
17     rospy.spin()
18
19 if __name__ == '__main__':
20     listener()

```

```

^Carlikar@srikar-VirtualBox:~/catkin_ws$ roslaunch beginner_tutorials reverse.py
[INFO] [1650255854.560320]: maET naayihbA skcoR
[INFO] [1650255854.760537]: maET naayihbA skcoR
[INFO] [1650255854.960696]: maET naayihbA skcoR
[INFO] [1650255855.161815]: maET naayihbA skcoR
[INFO] [1650255855.361206]: maET naayihbA skcoR
[INFO] [1650255855.560937]: maET naayihbA skcoR
[INFO] [1650255855.760684]: maET naayihbA skcoR

```

Figure 3: Output for the reversing node

Listing 3: Reversal node

```

1 import rospy
2 from std_msgs.msg import String
3
4 def rev_sentence(sentence):
5
6     words = sentence.split(' ')
7
8     reverse_sentence = ' '.join(reversed(words))
9
10    return reverse_sentence
11
12 def reverse(s):
13     str = ""
14     for i in s:
15         str = i + str
16     return str
17
18 def callback(data):
19     snew=rev_sentence(data.data)
20     snewnew=reverse(snew)
21     rospy.loginfo('%s', snewnew)
22
23
24 def listener():
25
26     rospy.init_node('listener', anonymous=True)
27
28     rospy.Subscriber('team_abhiyaan', String, callback)
29
30     rospy.spin()
31
32 if __name__ == '__main__':
33     listener()

```

Note: I wasn't able to create a new topic in the same launch file. Therefore, the reverse string is also being published over the same topic.

## 2.3 Task 3

Was able to run turtlesim.

We can kill the turtle using the command `rosservice call /kill turtle1`.



Figure 4: Killing the turtle

We can spawn two turtles using the command `rosservice call /spawn turtle1`. The locations and orientations chosen are  $[5\ 2\ 0]$  and  $[5\ 8\ 1.5707]$



Figure 5: Spawning two turtles

### 3 Section B

#### 3.1 Subsection B 1

Task completed.

Used binary search to find the required location.

Listing 4: C++ Code snippet used in the Problem

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int binarysearch(int arr[], int l, int r, int x)
5 {

```

```

6   while (l <= r) {
7       int m = l + (r - l) / 2;
8
9       if (arr[m] == x)
10          return m;
11
12       if (arr[m] < x)
13          l = m + 1;
14
15       else
16          r = m - 1;
17   }
18
19   return -1;
20 }
21
22 int main()
23 {
24     int m, n;
25     cin>>m>>n;
26     int k;
27     cin>>k;
28     int M[m*n];
29
30     for (int i=0; i<m*n; i++){
31         cin>>M[i];
32     }
33
34     int r = binarysearch(M, 0, m*n - 1, k);
35
36     if (r==-1) cout<<"False"<<endl;
37     else cout<<"True"<<endl<<r/n<<" "<<r%n;
38 }

```

## 3.2 Subsection B 2

Was not able to install opencv properly on both windows and virtual machine.

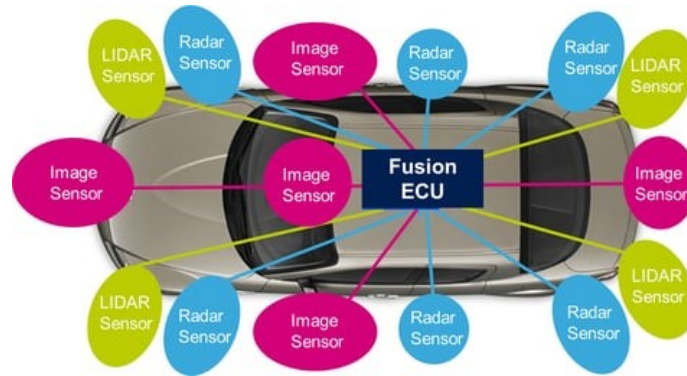
On windows, trouble arised with setting up path environment variable and runnning visual studio.

On linux, pip install command was not running properly.

## 3.3 Subsection B 3

1 There are mainly three different sensors:

- Camera
- Radar
- Lidar



Mono and stereo cameras in conjunction with radar systems provide a precise evaluation of speed and distance as well as the outlines of obstacles and moving objects. LIDAR's central premise is to act as a vehicle's eyes to always see in all directions.

- Camera:

Cameras work on the basic principle of landing the light emitted from the objects on a photosensitive surface through a lens. The position of the object can be defined as a vector with 3 elements  $[x,y,z]$  in space which is going to get projected through a lens to a point on the image plane  $[x',y']$ . The projected points are converted from the metric unit to pixels so the image can be used for further processing and extracting information i.e. detection, classification, tracking of objects that are in the path of our autonomous vehicle.

- Radar:

RADAR stands for Radio Detection And Ranging. A radar system helps in detecting the range and velocity of objects. Since it works using EM waves, it can work under any condition.

- Lidar:

LiDAR stands for light detection and ranging. The LiDAR sensor on a vehicle emits single particles of light, known as photons, that strike nearby objects such as cars, pedestrians, and trees. The photons then bounce back to the sensor. The LiDAR system records each photon's roundtrip data, measuring the distance and time to every object in the vehicle's vicinity.

### Sensor Fusion-

Sensor fusion is the process of collectively taking inputs from RADAR, LIDAR, camera, and ultrasonic sensors to interpret environmental conditions for detection certainty. Using multiple types of sensors together allows autonomous driving systems to experience the benefits of sensors collectively while offsetting their individual weaknesses.



2 The paper explains how the LIDAR system can be used to detect types of intersections and the objects nearby. It starts with how GIS, GPS, and INS systems fail to overcome this task.

The Lidar system emits beams of light in all directions, which are broken down into blocks of areas that decentralize object detection. Based on the shapes and curves of the Lidar emitted light it can detect using Support Vector Machine (SVM) whether the intersection is T-shaped or +-shaped. Based on this data the program the total accuracy achieved was above 80% but under 85%.

## 4 Acknowledgements

- ROS Wiki
- Google books
- Basic Radar and Lidar explanation sites