2π –Mapper

Virtual Jockeys

Introduction

Our project consisted of developing a robot that maps an unknown environment through feedback using ultrasonic distance sensors located in all four directions, and the live map along with the movements of the robot, is updated and displayed on the laptop screen in real time.

Motivation for Project

The motivation for our Project came from the navigation of remote environments, mostly those which are not navigable by humans, by small vehicles for the purpose of creating a map automatically. Inspired by this, we decided to create a low-height robot that would automatically create a map of its surroundings for further purposes.

Concepts Behind the Project

Now before you dive deep into our project, it is better to have an overall picture of the project, the Big Picture. To aid you in this regard we have created a flowchart which also served as a guide for our next step during our project.
To view the flowchart click here.

Technical Aspects of the Project

Here's a brief description of all the modules used in our project.

➢ Stepper Motors

From the Motors Bootcamp, this is what we learnt about Stepper Motors (some new, some old) -

- Quantized rotations
- Very accurate
- Rotation in full circles possible
- Difficult to operate compared to others
- Can control the rpm, rotate specific angles (this one's very helpful)
- Can rotate with minimum angle of 1.8 degrees (varies from motor to motor)

- Draws over 1A current (can't be controlled using L293D because it gets burnt and the motor vibrates a lot). We used L298 Driver to run our Stepper motors

We used a stepper motor than can handle 3 kg (we're using 2, so they should serve the purpose).

Here's a link that explains the separation and sequence of wires very well.
Stepper Motor Basics - 6 Wires Unipolar / Bipolar Motor

➢ Raspberry Pi

One of the many challenges in using R-Pi was how to setup the R-Pi. In this document we have described this process in detail.
Coursera R-pi course-Basic course on raspberry pi
Pdf book on RPi-Getting started with Raspberry Pi by Richardson Wallace
I2C LINK:
http://www.instructables.com/id/Raspberry-Pi-I2C-Python/?ALLSTEPS

➢ Ultrasonic Sensors:

We have decided to use HC-SR04 Module Sensors.
Controlling ultrasonic sensors via raspberry pi-
http://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

➢ Gyroscope

This link gives documentation of the smbus library for i2c in R-Pi
This link shows how to configure i2c between rpi and mpu
We went ahead with the gyroscope module M270 HMC5883L (triple axis) for correcting the orientation and direction of our robot.

Project Details

➢ SLAM

This is the link for a course by udacity on SLAM.
After discussing further, we decided to ditch particle filters and used an alternate

form of the technique. The localization part of the bot would be determined using Inertial Navigation System, termed appropriately as "*Local localization*".

➢ Systematic Exploration and Mapping Algos:

The map will be represented by a 2-D Array.
There will be 0's and 1's. The robot will also keep track of all the blocks that it has visited/encountered as obstacles and will update them on the map as well. Along with this, it will also keep track of how many times it has visited each block.
At a T-turn, if no block has been visited, it will turn left; if one block has been visited, it will turn towards the other block; if both have been visited, it will turn towards the one with less visit-frequency. (It would take into consideration all the blocks (max 4 in number) for visit frequency to decide which way it has to go.
The priority order for precedence of which each move (left, forward, right, or back) executed is L, R, F, B - i.e., not taking visit frequency into account, the robot will first try to move left; if there's an obstacle there, then right; then forward, and finally if all 3 directions are locked, it would move backwards (after taking U-turn, that is)
This priority order ensures that

➢ Specifications of the Map:

The size of one block of the map would be 24 x 24 cm. The grid would consist of 10 rows and 10 columns. Some of the blocks might have obstacles and some might not. The robot doesn't know that and so it has to find out. The obstacle themselves would be square blocks a little smaller than the size of one block, about 22 x 22. (about 1.2 times the size of our bot).
The height of the obstacles would be enough such that the ultrasonic sensors is able to detect them. There is currently no limit on the complete size of the map. The map might have as many blocks as possible and the robot should be able to map the environment.

➢ Output of the map

We showed the output of the map on the laptop using a 2D map with different shades of grey to represent the conditional probability of an obstacle being present in that specific block. One important modification is, that we didn't show the gridlines in our map, so that it gives a better feel of what the robot is actually doing.

This link gives a description about PyGame, a library which we used to create our game-board.
The code given on this link in the main answer directed us exactly how to create a game-board.
This link provides us with the RGB schemes of various colors which we'd directly use in our code to show various degrees of probability of presence of an obstacle.
http://www.rapidtables.com/web/color/RGB_Color.htm
This link helped us in creating the grid for the output of our map.
http://programarcadegames.com/index.php?chapter=array_backed_grids

➢ Shortest Path Problem

Since we had quite a good amount of time left (about one week) and the debugging was almost complete, we decided to explore the Shortest Path between two points-problem as well. Here's a short description of the same:
To compute the shortest path we start from the initial position of the robot and explore each of the possible neighbours from there. As we are exploring the neighbouring states we keep track of the number of steps it took to reach that state. We continue this process until we reach the goal position. In this way, by keeping track of the cost it took to reach each state we are able to compute the shortest path to the goal. The name of this technique is **A-star algorithm** and it is based on the **breadth-first mode of search**.

➢ Timeline of our Project

- ❖ WEEK ZERO
  - ➢ Did plenty of research on SLAM, watched videos from online courses, read papers.
  - ➢ Decided on the use of Omni Wheels and decided to go ahead with only 2-D maps instead of a 3-D Map
  - ➢ Discussed the pros and cons of using ROS or Raspberry Pi, and ultimately hit upon R-Pi
  - ➢ Surfed online for components that we might need to buy
  - ➢ Visited Lamington Road once to check the availability of the components of our project and noted down their price to have rough estimate of the expenditure.

- WEEK I
  - Studied the SLAM algorithm, ideated the code, and completed the code on Python.
  - Learnt the use of Motor Driver Circuits
  - Achieved basic usage of Raspberry-Pi
  - Interfacing of Ultrasonic sensors with R-pi completed
  - Decide values of some of the variables in the project (mainly the code values)
  - Buy all the components required for the project:
    - Raspberry Pi 3
    - 2 Ikg Nema 17 stepper motor, 2 L298 motor drivers
    - 4 Ultrasonic HCSR04
    - Lithium ion Battery
    - 2 Wheels
  - Designed the base of the robot using acrylic.

- WEEK II
  - Making of the bot - placing all the components on the bot after deciding their positions
  - Integrating the bluetooth functionality (Since we were told to ditch WiFi in the review meet, we integrated the bluetooth functionality of the R-Pi with the system)
  - Learnt to use MPU 6050 and how to use its values (double integration) in our project for localization.
  - 65% progress in producing the final output using Processing 3, a software which we were going to use in order to show the displaying of the final output of the map and the movement of the robot in the grid.

- WEEK III
  - Integrated the M270 HMC5883L into the movement of the bot (to correct its direction after every turn)
  - Completed the Pygame coding part (we were told to ditch Processing 3, so we learnt it and completed it during this week)
  - Created obstacles for the map and developed maps using thermocol by stacking and sticking them up.

- ➢ Did plenty of test runs through various shapes and figures testing all the features of the robot
- ➢ Uploaded video on Facebook group B-)
- ➢ Did work on Shortest Path Algorithm

❖ WEEK IV

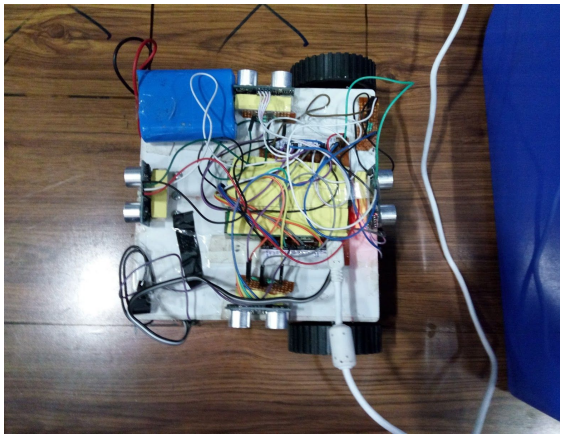
- ➢ Abstract completed by now
- ➢ Did a few corrections, like decreasing the size of obstacles (making it less than the cell size, so that the bot doesn't collide while turning)
  - ■ More appropriately, we increased the cell size
- ➢ Worked on adding Shortest Path Problem feature:
  - ● Bot would be able to figure out and move on the shortest path between any two points, which would be selected by the user

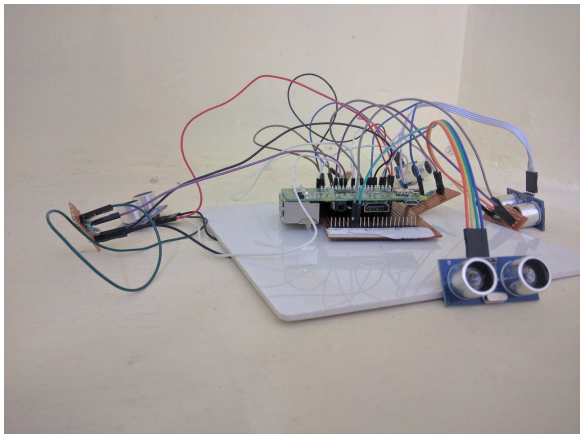
➢ Our project code

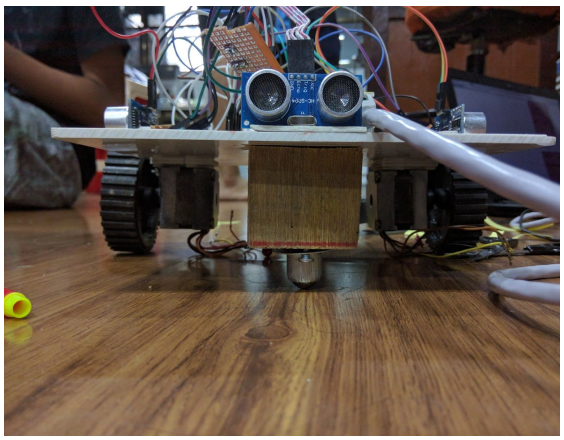
The entire code of our project is uploaded on GitHub over here.

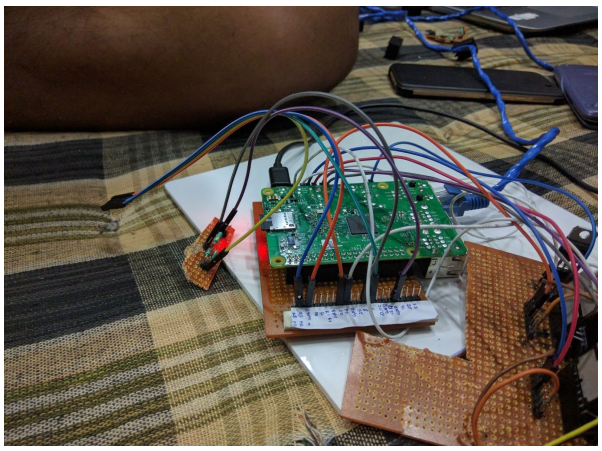
➢ Online Memorabilia

To view all the pictures and videos of this project view this folder.
Some of them are given here:









We also published a YouTube video which can be viewed here.

These are the slides prepared by us for the weekly Review Meets:

1. Link for First Review Meet: https://docs.google.com/presentation/d/1x5woR_QSLkak4-zzeQyAn9LFjl1R67WEwfqaCk4iTKo/edit#slide=id.p
2. Link for the Second Review Meet: https://drive.google.com/open?id=10y-K-Z5vvKKI_MRCczCsIsbObgc24A1ivoJrtCnTcGc
3. Link for the Third Review Meet: https://drive.google.com/open?id=1Mgfl25WTJj8OVoA0JX7qvol_ud15ym3gdZ0dFj-N7nU

➢ Components' List

- Raspberry Pi 3 - Silikon Electronics Lamington @ Rs 3500
- Ultrasonic sensors HC SR04 - MAC NET Technology @ Rs 100
- Explore Labs Magnetometer 3 Axis - HMC5883L (I2C) - Amazon @ Rs 500
- Stepper Motors Nema 17 3 Kg-cm Torque - Servo Electronics @ Rs 250
- Lithium ion Battery - VIDEOTON Lamington @ Rs 900
- L298 Motor Drivers - Mangaldeep @ Rs 110
- IC7805 - Mangaldeep @ Rs 30
- Acrylic sheet - Mangaldeep @ Rs 70
- Thermocol sheets - IIT Main Gate @ Rs 350

- **Sources of error:**
  - Slipping of wheels and hence integration of error which can lead to erroneous localisation
  - Manufacturing error which causes the same result as above.
  - If the bot somehow collides or even touches any obstacle from its front such that its movement is halted for some moments, then it can get a completely wrong idea of where it is in the map - fortunately the verification by the accelerometers and the gyroscopes can help here.