eYSIP−2018

A system for solving jigsaw puzzle using multiple robots

Aniket Anantraj Navlur
Kiran Suvas Patil
Ashis Kumar Maharana
Mentor 1: Abinav Sarkar
Mentor 2: Kalind Karia
Duration of Internship: 21/05/2018 – 06/07/2018

2018, e-Yantra Publication

A System for Solving Jigsaw Puzzle using Multiple Robots

Abstract

The prime motive of this project is to develop a multi Robot based Autonomous Puzzle Solver system that can solve a Jigsaw puzzle.

Overview of Task

Task No.	Task
1	Python, OpenCV, Firebird V Intro,XBee Communication
2	Pose and orientation calculation of 2 Firebird robots using color/Aruco markers
3	Programming the Go-To-Goal Controller for single Firebird V robot. Tuning the PID values to perfection
4	Implementing path planning with Firebird V where obstacles have been placed in arena
5	Detection of jigsaw puzzle blocks usingTemplate Matching
6	Pick and place of blocks - gripper mechanism building
7	Implementing the entire solution for a givenjigsaw puzzle
8	Documentation and reporting results

Completion status

The project is completed till task.6 i.e. building Gripper mechanism. Implementation of the entire solution for a given puzzle is in progress.

1.1 Hardware parts

- FireBird Robots(2)
- XBee(3)
  - Xbee module
  - xbee usb adapter board
  - usb type B to type A converter
- overhead camera
- ArUco markers
- Servo motors for gripper mechanism
- flex sheet for arena
- puzzle pieces

1.2 Software used

- python
- openCV
- AVR Studio 5.1
- XCTU(latest version)
- AVRDude
- Fusion360

1.3 Assembly of hardware

Circuit diagram and Steps of assembly of hardware with pictures for each step

Circuit Diagram

Circuit schematic, simplified circuit diagram , block diagram of system

Step 1

Steps for assembling part 1

Step 2

Steps for assembling part 2

Step 3

Steps for assembling part 3

1.4 Software and Code

1.4.1 Dependencies

Before configuring the XBee modules one should first understand the different parameters of it. This article should help. For configuring the xbee modules latest XCTU software is used. The channel is chosen so, such that no other device interfere the communication.
Detection of ArUco markers using python requires integration of aruco library to the current python version installed. The aruco library can be found in opencv-contrib module. For installing opencv-contrib follow these steps(installing modules using pip).

```
pip install opencv-contrib-python
```

For communicating to xbee module using python requires xbee and pyserial module. There are many similar modules for serial communication and xbee, but we need these two modules. If you want to learn you can search and go through other modules also.

1. pyserial

   ```
   pip install pyserial
   ```

2. xbee

   ```
   pip install xbee
   ```

3. optional digi-xbee

```
pip install digi-xbee
```

The xbee model provided(XBee S2C) cannot be read properly using the earlier versions of XCTU. So it is recommended to use the latest version. Follow the previous Docs for configuration using XCTU. After configuration and testing over XCTU, communication is again tested using the python script. Once the communication is established it is time to connect the xbee modules on the Firebird V robots. How to guide can be found in the hardware manual of Firebird V.
Then comes the localisation of robots. For that purpose we are using overhead camera and ArUco markers. The position and orientation of the ArUco marker is first calculated using python script and math module which is inbuilt in python. Angle is calculated such that it ranges from 0 to 360 degrees. The python script can be found here.
For the Go−To−Goal Controller we first found the orientation and distance of the robot from its goal or destination. Then the error angle is calculated by subtracting angle with respect to frame from angle with respect to destination. The code can be found here. In another way the error angle can be found by

$$errorangle = \arctan\left(\frac{m-n}{1-m\times n}\right) \tag{1.1}$$

m = orientation of robot with respect to frame
n = orientation of robot with respect to destination

This error angle is then passed to the PID controller which then adjusts the speed of motors to maintain a zero error. The python script sends data packets to the Firebird V robot via Xbee. These data packets are formed by the following values.

$$\langle T|d_x|d_y|P|K_p|K_i|K_d|R|r_x|r_y|A|angle|\rangle \tag{1.2}$$

< symbolizes beginning of data packet
T symbolizes target
$\mathrm{d_x}$ = destination x co-ordinate
$\mathrm{d_y}$ = destination y co-ordinate
P symbolizes PID gains
$\mathrm{K_p}$ = proportional gain
$\mathrm{K_i}$ = integral gain
$\mathrm{K_d}$ = differential gain
R symbolizes robot position

r_x = robot x co-ordinate
r_y = robot y co-ordinate
A symbolizes error angle
$>$ symbolizes end of packet

Once received the packet is successfully parsed by the robot in an ISR and the speeds are assigned to the motors as PWM duty-cycle.The c code and python script can be found here.

Github link for the repository of code

Brief explanation of various parts of code

1.5 Use and Demo

Final Setup Image

User Instruction for demonstration

Youtube Link of demonstration video

1.6 Future Work

What can be done to take this work ahead in future as projects.

1.7 Bug report and Challenges

Any issues in code and hardware.

Any failure or challenges faced during project

Bibliography

[1] Ad Kamerman and Leo Monteban, *WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed band*, 1997.