Sheffield Hallam University

2019

Zumo Search and Rescue Report

MODULE TITLE:

Programming things

PROJECT BY:

SIMAEI, SAM

Introduction

This assignment which we had to develop a Zumo robot with Arduino board to perform a number of tasks. The robot is also equipped with an XBee module which allows the robot to perform all the commands given to it wirelessly.
Also, an appropriate GUI needed developed using Processing application and G4P Builder toolkit.

Tasks Achieved

The assignment had 6 tasks that needed to be developed out of which I achieved the following:

- Task 1 – Manual control of Zumo
- Task 2 – Autonomous control of Zumo
- Task 3 – Turning Corners
- Task 4 – Zumo searches a room
- Task 5 – The T-Junction

Acknowledgments and sources

The application is developed using libraries of two different sources, Pololu which is the manufacture of Zumo V1.2 and Tim Eckel who is the developer of the NewPing library.

Pololu

The robot comes with built in libraries developed by Pololu company some of them are:

Library	Usage
ZumoBuzzer	Enables robot buzzer which signals different stages.
ZumoReflectanceSensorArray	Used to keep the robot within the two black lines.
ZumoMotors	Used to enable Zumo's built in motors.

The Reflector Sensor library I used was inspired from a Pololu examples called maze solver. The code from this example got abstracted and developed on so instead of following a black lane keeps itself in between two black lanes. The robot calibrates itself to understand the level of the darkness on the black line and stores it in an array for each of the sensors.

Tim Eckel

Tim Eckel is the developer behind of the NewPing library. The library uses two ultrasonic sensors which is mounted in front of the robot to detect the objects that were places in the map or in our case survivors.

Operation Overview

1- The way the Zumo robot behaves, is by giving an initializing chime using the buzzer which alerts the user for awaiting command. After the appropriate command was given the robot sweeps on the one of the black lines so the sensors can know the difference threshold value of black line of the map and the white space. This makes the robot to be functional on different surfaces or different maps as they can have different colors or reflection according to their material.

2- Using the GUI, the robot starts its autonomous future until hits a wall in front of him and correcting its path if it detects a side wall. After encountering a wall in front of it the robot a message will be displayed on GUI and lets the user know that it has switched to the manual mode. Then the user turns the robot manually and gives the applicable command that the turn is complete, this re-enables the robot's autonomous behavior and waits for user command to either stop for a room or to stop in case of emergency command.

3- When robot stops to search the room, the user tells the robot the direction of the room and turns it manually towards it after the turn completed the robot goes forward and sweeps the room for survivors. After searching the rooms and finding the survivors the robot lets the user know that either a survivor has been found or not and goes back to manual mode the user takes the robot out of the room and re-enables the automatic feature.

4- Robot at this stage will hit a T-Junction and user decides either to go left or right after the decided to turn the robot goes to auto mode and searches the room on its way and at the end hits the end wall. The user will let the robot know that it's one of the 2 ends of the map and turns the robot and re enables the automatic feature.

5- At this stage until the Zumo has not crossed the T-junction it should ignore any incoming messages and goes forward automatically, after crossing the user lets the robot know that it has crossed the T-junction and robot will again listen to all incoming messages.

Key Issues

Avoiding the corridor walls

This was one of the main challenges I faced during development of this project and its crucial to the project success, using the example available on Zumo's library for solving a maze and by storing the values of the black and white lines of each sensor in an array instead of using the standard QR-Threshold, this way each time the robot encountered a side wall as its only checking the values of sensor number 0 and 5, it will reverse a bit and make the turn required to fix its path and continue the course on a straight line.

Detecting survivors

Finding objects in the rooms were also part of this assignment and it was challenging the ultrasonic sensors are very sensitive and they managed to break during development and got replaced by a new one, also the it was very important to set the right distance the ultrasonic sensors needs to send its way and measure the response time.
To make sure the robot finds the survivor no matter the location the same sweep mode that the robot uses to calibrate the reflector sensors was used to sweep the room for any object and to stop if there were any survivor and point to them.

T-junction

This part of the assignment was a bit tricky, making sure the robot will ignore all incoming messages and wait only for the T-junction crossing made me to change some of the structure of the Arduino code in order to accommodate this feature.

Displaying Zumo's logs on Processing

Zumo uses the serial window available built-in in the Arduino application for displaying all outputs written by the coder, the same serial port also sends and receives all the commands from user transmitted with XBee device. Using this channel, I coded all these commands to be sent via XBee to Processing incoming channel to be displayed on the GUI designed.
This wasn't an easy task as Serial port sends all the commands like a stream and the GUI wasn't displaying the messages properly, by using a string terminator that I learned last year on java, the Processing could detect the next message each time a message contained and string terminator which usually in practice is the explanation mark.

G4P Builder compatibility issues on Mac OS

As of the time of this report, due to the new Java version released al also the new Mac OS Mojave release by apple G4P builder has a major compatibility issues once the design of the GUI is finished, I wasn't able to re edit the interface and this made G4P tool to crash and gets un accessible. To overcome this problem, I had no choice but to use a virtual machine

windows to design my GUI and export it to Mac OS for further improvements as windows 10 had no problem re editing existing designs.

Walls in front of the robot:

The reflector sensors in front of the robot are not very accurate and sometimes the robot thinks that it has encountered a side wall, this puts the robot into infinite loop and a nonstop path correction. To solve this, a fix counter was implemented to count the fixes robot does and if its more than the defined number robot will know that it has hit a wall and changes the status to manual mode.

Over Flown Serial ports

As it has mentioned in processing:

the console is relatively slow. It works well for occasional messages, but does not support high-speed, real-time output (such as at 60 frames per second). It should also be noted, that a println () within a for loop can sometimes lock up the program and cause the sketch to freeze.

This was an ongoing issue and required a bit of work around for displaying the messages properly. Most of the time processing was crashing if the print was in a while loop or there were few of them one after another. So, having the messages as a string and adding them all together to use one Serial Print was stopping the processing from constant crashes.