Programming Things: Assessment 2017_18
Zumo Search and Rescue

For this assignment we were assigned 5 tasks to program Zumo for a simulated search and rescue scenario. Summary of these tasks is as followed:

1. Zumo can be driven in the corridor wirelessly via xBees using a "GUI"
2. Zumo detects walls and keeps itself in corridors. It stops when there is a wall in front
3. When it detects wall, as in task 2, or if it detects a corner; a user can take control of Zumo and turn it into right direction. When user is done Zumo can resume "task 2".
4. Zumo is stopped by user when there is a new room or corridor. User can direct Zumo into new room or corridor by sending appropriate command and appropriate direction.
   A. Zumo records whether it went to a room or a corridor. If its moved into a room it should scan the room for any object using U/S sensor. If it finds an object it sends a message with location of that object.
   B. It recognises if another room was in same corridor or a in a different corridor
   C. When Zumo is finished search a side corridor it stops. It is then told to exit corridor. At this point Zumo recognises that it is exit side corridor and is return to last corridor (main corridor)
5. If Zumo reaches end of main corridor it can be told to come back. At this point it processes all information recorded in task 4 to backtrack every corridor and every room in them. If rooms that had objects are not empty it alerts and sends message.

Acknowledgement and Sources

Pololu

Pololu being the manufacturer of the Zumo provides range of libraries which are used extensively for this project. Pololu also provides a number of example projects to be used with Zumo which were very helpful. Libraries that I used are as followed:

Library	Use
ZumoMotors	Controls speeds of motors on the Zumo
ZumoBuzzer	Controls buzzer on the Zumo shield
Pushbutton	Controls buttons on the Zumo shield
ZumoReflectanceSensorArray	Controls reflectance sensor array mouse under Zumo shield
QTRSensors	For using sensors such as ultrasonic sensor and reflectance sensor

Tim Eckel – NewPing Library

NewPing was developed by Tim Eckel to be used with ultrasonic sensor for best performance. I used this library with ultrasonic sensor to detect any objects in a given range for the distance.

Netbeans IDE

I used Netbeans IDE along with **javafx** to easily create a GUI. **Javafx** allowed me create components such as buttons, combo boxes and text areas for a better user interaction.

jSerialComm

I used this library with Netbeans IDE to communicate with Zumo via xBee. This library allowed me to list, open, close and use available ports.

Overview

The Zumo uses its reflectance sensors array to determine if its above back line or a white ground. I used border detection example program as a reference for detection walls to the left and right. When started Zumo continues to go forward if first sensor and last sensor in the reflectance sensor array are “not above line”. If first sensor is above line Zumo go backwards and turn left or if last sensor is above line it turns right. If both sensor are above line it processes it as a front wall (more on this later).

Zumo increments a room number or corridor number when entering and decrements on exit. The program uses a structure called “nav” which contain corridor id, room id, a room or corridor and left or right. It stores this data in a array of nav which can be used to tell which corridor was on left or right or how many room a corridor have or whether we are in same corridor or a different one.

Variable and Functions

Some variables and their purpose that are worth mentioning

Variable	Purpose
LED	The pin number for the LED on Zumo
TRIGGER_PIN, ECHO_PIN	Pin numbers used for ultrasonic sensor
MAX_DISTANCE	Maximum acceptable distance for ultrasonic sensor
ABOVE_LINE	If the Zumo is above a black line
QTR_THRESHOLD	Threshold for reflectance sensors
TURN_SPEED, FORWARD_SPEED	Speed at which Zumo motors will be set
REVERSE_DURATION, TURN_DURATION	Delay duration for reverse and turn
startTime, endTime	These are “long” variables which determine time between left and right sensor when they are above line. For example, startTime starts the timer when first sensor in the array is above line and endTime stops the timer when last sensor in the array is on the line.
elapsedTime	This contains the difference of endTime and startTime. If this time is less than a certain number, then the Zumo has reached a corner. Essentially this is used to detect ping pong motion of Zumo when it hits a left wall immediately after a right wall or vice versa. There cannot be a left wall immediately after the wall on the right unless it’s a corner.
navData	This is an array of structure of type nav. It contains records of Zumo’s previous motion
nav (struct)	A structure containing corrID, roomID, roomCorr, leftRight
corrData	A nav “object” which is used for task 5.
corrID, roomID, leftRight, roomCorr	corrID and roomID is the identifiers of what corridor we are in and what room of which corridor we are in. leftRight is the character ‘l’ or ‘r’ depending on if we went left or right roomCorr is also the character to identify if we went to a room ‘j’ or a corridor ‘k’
objectID, corridorID, roomID, roomIDCopy	These are of type int. ObjectID is for counting how many objects are found in all the rooms. corridorID and roomID are used temporarily to keep account of position

	on the building floor. If Zumo goes to different room or corridor roomID and corridorID are saved in nav “object” and then are reset to 0. roomIDCopy remembers how many rooms we visited in last corridor. For example if a corridor has two room and a subcorridor. We scan the first room and then if we want to enter the sub corridor we store roomID(number of visited rooms) of this corridor in roomIDCopy and reset roomID to 0. After we are finished with the sub corridor we go back in main corridor and make roomID equal to roomIDCopy and scan the second room.
inSubCorridor, isRoom, isExitingSubCorridor, resumed	These Booleans which are used to increment and decrement roomID and corridorID if they are true. isExitingSubCorridor activates the behaviour which remembers last corridorID and which direction to continue searching. resumed determines if we are controlling the Zumo or not. If false Zumo keeps itself in corridor
dir	A char to read the command from Serial

Some important functions and their purpose.

Function	Purpose
processCommand()	Reads from serial and uses a “switch case” to call desired functions
forward()	Resets the timers on startTime and endTime sensors (i.e. time between left sensor hit an dright sensor hit) and then resumes Zumo to keep in corridor.
stop()	Sets motor speeds to 0 and sends a message saying “stopping” then give control to user.
scanRoom()	Activate ultrasonic sensor. Detects and reports if any objects are found. Rotates around 360 degrees and checking for collision every 300 ms while rotating
keepInCorridor()	Detects left, right and front walls. If it’s a wall, then reverse and stop. If it’s a corner, then reverse and stop. If there is a wall on the left, reverse and turn right and continue forward. If there is a wall on the right, reverse and turn left and continue forward.
gotoLeftCorridor(), gotoRightCorridor()	Increment corridorID, make a copy of roomID. Set roomID to 0. Turn left or right and continue forward
exitSubCorridor()	Decrement corridorID and restore roomID(numbers of rooms visited)
turnLastCorridorDirection()	When exiting sub corridor if a wall is encounctered the turns to direction that it came from to continue searching.
sortByCorridor();	When we have reached of main corridor we sort navData by corridorID This give us data required to turn back
turnback()	Traverse the sorted navData using “i” as index from “navArrayCount” to 0;

Key issues and challenges

Staying in within walls

With the help on border detection example by Pololu it was fairly easy to detect walls on the left and right by detecting sensor [0] and sensor [5] from the array reflectance sensors are above black line.

- If sensor [0] is above line but sensor [5] is not. Reverse and turn left
- If sensor [5] is above line but sensor [0] is not. Reverse and turn right
- If both not above line. Continue forward

Detecting wall on front

Detecting wall on the front was a challenging task for me and below are few strategies that I tried.

First strategy was an obvious one. If sensor 0 and sensor 5 are both above line, then it's a wall. This strategy will work but only if Zumo is headed perfectly straight to the wall otherwise if sensor 0 come first on the line it will trigger "turn left" condition and same with sensor 5. This makes it extremely unreliable and it is guaranteed to fail because it is almost impossible to keep Zumo perfectly straight with mm precision. I also used electrical tape to draw lines which due to human error are also not perfectly straight.

I then tried to used built in millis() to calculate time between when the left sensor was above the line and time when right sensor above line. If sensor 0 come first on the line, we start the timer. Then we check if sensor 5 is on the line then, if it isn't then turn left else if sensor is on the line, we stop the timer. We then take time elapsed between when the sensor 0 was above the line senor 5 was above the line. If this time is greater than for example 50 ms then its definitely not a front wall. But if they time is less than 50ms then it means sensor 5 was above line immediately after sensor 0 and we have detected a front wall.

If sensor 5 instead of 0 comes first above line, we do the same thing but turn right.

This method worked very reliably and worked with various angle of attack for Zumo to the wall. It detected and stopped almost every time there was a wall on the front. However, no matter what I tried it didn't seem to turn left or right as it was supposed to if only one of the sensors was above the line.

Finally, the third strategy that I tried uses delay function. If sensor 0 is above line, then wait 50ms and check if sensor 5 is also above line. If yes, then it would be a front wall otherwise turn accordingly. This strategy seems to work reliably even if the angle of attack is slightly different. I achieved 50ms as the optimum delay time via trial and error because if delay is too long the Zumo would overshoot the line and not detect a wall at all or if delay is too short then scenario would be same as in first strategy.

Detecting corner

To detect a corner a I use millis() function to calculate elapsed time between when the sensor hits left wall and when it hits right wall. If the elapsed time is less than a second, then it would mean the left wall hit the right wall immediately after the left wall and therefore it's a corner. Similarly, if the sensor hits right wall first and hits left wall immediately after that would mean it's a corner.

Keeping track of corridors and room

Everytime we press the corridor button one object of "nav" is stored in navData array at the index of navArrayCount and navArrayCount is incremented. This "object" will contain "l" or "r" for leftRight "k" for roomCorr, current corridor id incremented by 1, and room id which will set to 0 because we just entered this corridor and haven't visited any rooms.

However, if we press a room button, again a nav object is created with "l" or "r" char for leftRight, "j" for roomCorr, current corridor id, and room id increment by 1.

The data will look like this.

```
{
 {1, 0, ‘k’, ‘r’}       //Entering new sub corridor to the right – corridor number 1
 {1, 1, ‘j’, ‘l’}       //Entering new room number 1 to the left in sub corridor  is still 1
 {2, 0, ‘k’, ‘l’}       // Entering new sub corridor to the left – corridor number 2
 {2, 1, ‘j’, ’r’}       //Entering new room number 1 to the right in sub corridor is still 2
 {2, 2, ‘j’, ’l’}       //Entering new room number 2 to the left in sub corridor 2
}
```

Wireless Communication

Using GUI in Netbeans IDE I was able to connect with port that xBee is using however, the data/message sent was almost always broken into parts and was unreadable and often missing important information. I fixed this issue by changing baud rate from 9600 to 57600. By doing this I was able to get full message almost all of the time, but I still receive broken messages if they are long. However, this also sometimes makes my GUI unresponsive. When pressing a button, it will not write to serial and we will have to try pressing the button again.

GUI Guide/Instructions

We start using gui by first selecting a desired port via which we can communicate with Zumo wirelessly. Once we select a port the buttons on the GUI are enabled. The GUI should display a message asking to press Zumo button to ready Zumo. When we have press the ZUmo button its buzzer should beep indicating that its ready. From here on we can move Zumo forward and backward etc. Once we press forward “^” or press continue “c”, the Zumo should start moving on itself. It will automatically stop when it is supposed where can control it using directional buttons. To make it continue on its own we press continue button again.

If we want to enter a sub corridor we **must** do following.

1. Stop the Zumo using stop button.
2. Press Corridor button
3. Press left or right Button
4. Zumo will then move make its way to corridor automatically
5. At the end of a corridor Zumo will stop
6. Press the Exit Corridor button and zumo will turn around and exit

If we want to scan a room, we **must** do following

1. Stop the Zumo using stop button
2. Press Room button
3. Press left or right Button
4. Zumo will steer left or right
5. Press Scan button
6. The Zumo will go forward and rotate 360 degrees
7. It will then stop
8. We will then steer Zumo out of the room
9. Press continue button

When Zumo reaches end of corridors we can press the Turn back button to turn the Zumo back to starting point.

Pressing Turn back button will also display the navigation data.