# Programming Things – Assessment 2016-17 Zumo: Search and Rescue

## Tasks Achieved:

Against the objectives set in the assignment, my Zumo achieved the following:

**Task 1** – The zumo can be driven down the corridor using the w,a,s,d keys to move and the space bar to stop.

**Task 2** – The zumo automatically keeps within the corridor walls using the reflectance sensors to turn away from the walls.

**Task 3** – the zumo recognises that it has reached a corner by detecting a wall in front of it, stops and sends a message using the xBee indicating that fact. It then deactivates task 2 behaviour to allow task 1 control, and resumes task 2 behaviour with keypress ‘c’.

**Task 4** – The zumo enters a room and sends a message over the xBee indicating this, along with applying a room number and which side of the corridor the room is on. The zumo also records this information, and performs a scan of the room using the ultrasonic sensor for objects. A ‘c’ key press again resumes task 2 behaviour in the corridor.

**Task 5 –** The zumo detects objects in front of it using the ultrasonic sensor and reports back using the xBee. The zumo detects objects whilst moving and identifies over the xBee whether the object is in a corridor or a room. If in a corridor, the zumo stores the last room before the object along with the last turn.

## Key issues:

**Automatically keeping within the corridor walls (Task 2):**

This was a key issue with the assignment as the rest of the tasks depended upon it. I looked at the Arduino examples ‘BorderDetect’ and ‘LineFollower’ to try to devise a solution to it. I first tried implementing something similar to the borderDetect example where the zumo would turn right if it detected a wall under the leftmost sensor and left if the opposite was true, at first this was an alright solution but the constant ‘zigzagging’ made it difficult to detect corners as sometimes the robot would be at an angle. The difference in lighting conditions also meant using a static threshold for the detection of walls was a bad idea as it could change each run. To solve these problems I implemented a calibration of the reflectance sensors in the setup() function and adapted the lineFollower code to read the position of the lines relative to the Zumo in order to stay within the corridor more reliably. To improve on this further I also attempted to use the gyroscope in the Zumo to re-align the zumo after each wall-hit so that it would always be travelling straight after adjusting itself, but came across an issue where the gyroscope couldn’t be calibrated due to the weight/interference of the xBee shield on top of the zumo.

**Recognising a corner (Task 3):**

This was another key issue as occasionally if the Zumo was going towards a corner at an angle, the sensors would pick up that the wall was on the far left/right of the zumo and assume it was just a wall. This would then activate the task 2 behaviour and the zumo would turn around the corner on Its own. I would have liked to have used the re-alignment behaviour I mentioned previously in task 2 so that the zumo would always be travelling straight into a corner to avoid this issue, but as I couldn’t use the gyroscope I had to think of another solution. My end solution was to use a timer to check whether the zumo had done more than two corrections within a small amount of time (3ms). If this was the case, it was fair enough to assume that the zumo had been stuck in a corner and I could treat it as such. I combined this solution with using the raw data from sensors [2] or [3] to check if the middle of the zumo was on a line.

**Storing room information/scanning rooms (Task 4):**

This was an issue mainly because of the collection classes you are able to use in the Arduino IDE, to get around this I used a light version of the STL Vector that had been ported by ZacSketches on GitHub ([**https://github.com/zacsketches/Arduino\_Vector**](https://github.com/zacsketches/Arduino_Vector)). This seemed to be the best option around and I created a class for rooms which contained information about the room number, side of corridor it’s on and whether any objects are contained in it and then stored a Room object in a vector each time I scanned a room.

**Detecting objects whilst moving (Task 5):**

The key issue with this task was detecting objects whilst at the same time moving, in order to solve this I looked at an example in the newPing library called ‘NewPingEventTimer’ which shows how to use the newPing library in order to have a more event driven program. This allowed me to fire the ultrasonic sensor every 50ms and check against a threshold whilst at the same time moving through a corridor.

**Optimising a return route and navigating to the start of the course (Task 6):**

I didn’t manage to finish implementing this task, but the idea behind this was to use task 2 and 3 along with the stored information about turns and rooms to have the zumo travel back along the track, turning the opposite way to what was recorded at a corner. The Rooms and Objects were stored in classes and vectors and information about each room would have allowed the zumo to ignore any rooms that it did not find objects in, and turn into each room with an object using the stored information on which side of the corridor the room was.

## Sources:

The code for my Zumo incorporates libraries and code from 3 sources: **Polulo** (manufacturer of the Zumo), **Tim** **Eckel** – [tim@leethost.com](mailto:tim@leethost.com) (developer of NewPing library) and **Zacsketches** on github (developer of lightweight STL vectors on github - <https://github.com/zacsketches/Arduino_Vector>)

**Polulo -** [**https://github.com/pololu/zumo-shield**](https://github.com/pololu/zumo-shield)

* **PushButton –** This library was used to detect presses on the zumo’s pushbutton to allow the user to stop the zumo at any time and when calibrating the reflectance sensors to give the user time to move the robot.
* **ZumoBuzzer –** This library was used to create sounds to inform the user of actions.
* **ZumoMotors –** This library was used to control the 2 Zumo motors for movement.
* **ZumoReflectanceArray –** This library was used to perform task 2 by checking the position of the Zumo with regards to the walls (lines) and to perform task 3 by checking the raw data of the middle reflectance sensors to check for a hit against a wall.
* **LSM303 –** This library was used as I attempted to operate the gyroscope in the Zumo to allow precision movement for task 2/3.

**Tim Eckel -** [**http://playground.arduino.cc/Code/NewPing**](http://playground.arduino.cc/Code/NewPing)

* **NewPing –** This library was used along with the provided example ‘NewPingEventTimer’ in order to have my robot continuously ping the Ultrasonic sensor in order to check for objects.

**ZacSketches -** [**https://github.com/zacsketches/Arduino\_Vector**](https://github.com/zacsketches/Arduino_Vector)

* **Vector –** This library was used to implement a lightweight version of the STL vector to provide a collection class for rooms and objects to be stored in.

I also used the following source when writing code to try keep to a clear and readable code style:

<https://www.arduino.cc/en/Reference/StyleGuide>