Zumo: Search & Rescue

In this assignment, the objective set was to modify and code a Zumo robot so that it could successfully navigate around a track looking for “people” as it did so. These people or obstacles could be in a room which the Zumo would need to go into and search or in the corridor, both of which would be found using ultra-sonic sensors.

## Task 1

The objective of task one was to simply get the Zumo moving using the w, a, s, d keys and to use the space bar to stop. This was achieved by reading the incoming char value from the serial and comparing it in a set of if statements. The Zumo motors were then set at different speeds according to which key had been pressed. This wasn’t difficult as it was something we had gone over in a tutorial.

## Task 2

This task involved keeping the Zumo within the confines of a track (or “corridor”) while the user was still in control this meant temporarily overriding the current direction and realigning the robot by directing it back towards the centre of the corridor. The reflectance sensors at the front of the Zumo were used for this purpose, the six sensors were divided up and used to detect when the front of the robot had crossed over a line. The calibration of the sensors caused some issues at first because uneven light levels meant that despite the sensors being covered by a shield they all required different levels. By this I mean that if the trigger value for the line was set too low then in some cases the sensors would think they had crossed a line when all that had happened was they had moved into a shadow or turned away from the light. To combat this I took the low-level calibration for each sensor and put it into an array, I then added this to an “allowance” value so that each had a trigger level that accounted for their current light level and any differences in sensitivity between the sensors.

## Task 3

This was essentially just an extension of task 2 in that the Zumo needed to detect when it was crossing a line so it could recognise a corner. I began by separating the sensors into three equal sections (left, right, middle) but this proved to be insufficient as the Zumo would often try to turn the corner itself using task 2’s behaviour rather than using the new corner behaviour. To solve this I changed the sensor groups so that there was a single sensor at each side for wall checking and all four of the central sensors for corners. This still did not entirely fix the issue so I increased the trigger level for the outside sensors meaning that if they all crossed a line at the same time the corner would be detected by the inside sensors first. It was also required that the behaviour from task 2 was disabled once a corner had been detected so I created a variable called “go” to determine whether the Zumo should guide itself or not. Once the corner had been turned the key ‘c’ was then used to set the go variable to true and allow normal corridor behaviour to resume.

## Task 4

Task 4 was tricky in that the robot needed to do multiple things without being interrupted by the (human) controller or other behaviours. To do this I set up an if statement which first checked if the room behaviour had been activated by the controller, all the normal behaviour was put into the else statement so that it would be ignored while the robot was checking the room. To activate the room behaviour the controller stops the Zumo outside of a room and presses the ‘r’ key, which prepares the Zumo for recording which side the room is on. The controller then turns either left or right and stops once the Zumo is facing the room. The Zumo will recognise this behaviour and move forwards into the room. It will then turn to the left of the room, turn right through 180° across the whole room, and then back left to return to its starting position. During all three of these turns the robot scans a distance of 30cm using the U/s sensors to check for people. There was particularly an issue with the turning timing which reduced dramatically as the batteries drained, fortunately this largely doesn’t affect the detection of people, however if they were placed in the corners closest to the corridor it may cause issues if the batteries are not fully charged.

## Task 5

Detecting the people in the corridor was a simple requirement as it just meant sending a ping with the U/s sensors every time the program looped. Recording the most recent feature however was the difficult piece as it meant reworking a lot of the already established structure. It meant that every corner had to be recorded along with its direction. Once they were all capable of being recorded it also meant setting a global variable called “priorFeature” which would be overwritten at every new corner or room. It is initialized as start in case a person is detected before any corners or rooms.

## Task 6

Here the Zumo was required to get to the end of the corridor and using the key ‘e’ the controller would signal this to the Zumo. It would then navigate to the start on an optimal route automatically without any user intervention, and return to any locations where it previously found people. While the corners were not an issue for this task, the rooms were the eventual downfall of this as I couldn’t manage to get the robot to stop outside the room in the correct place. This was due, I think, to an issue I previously mentioned about the batteries. The theory was that I would time each section of the corridor between the corners and rooms and then make the robot move forwards for the same length of time on the return. Due to failing batteries however the time was always out and how far the robot got became less and less as time wore on.

## Task 7

I created the GUI using processing and, in the end, controlP5. I initially had moderate success using processing alone creating a GUI where the directional keys could simply be hovered over with the mouse. I found this gave a lot more control over the robot than having to press the buttons. Unfortunately, I couldn’t get the Serial text to display properly as every time something was shown it would be quickly overwritten by another signal. If I had managed to get this working the messages would have displayed in real time, but the controlP5 method I have ended up using requires a button press for the messages to update. The buttons must also all be clicked on due to the restrictions I found with controlP5. To create the GUI code I used the example code that controlP5 have on their website for both the buttons and the textfield.