Programming ‘Things’ Group Assignment – Individual Report Jamie Pounds

For our group project, we had the idea of using Zumos to play a tic-tac-toe style game. This involved the use of different technologies including the Zumo, the Arduino board, Xbee’s, and a Raspberry Pi. Along with this hardware we planned to have it controlled by a GUI in Processing and all of it is linked to a MongoDB which then displays results and various other things on a website. The introduction of each of these technologies brought their own difficulties or problems that we had to resolve or find a way around. We chose these technologies because we were already familiar with the Zumo, Arduino Board and Processing from previous assignments within the Programming ‘Things’ module. But had integrated unfamiliar components like the Raspberry Pi to try and increase the functionality of the project. From other modules we had acquainted ourselves with website design and using MongoDB so adding this on to the project seemed to be an obvious way we could add functionality and increase the scope of our project.

The initial idea we had was to have two Zumos that navigate themselves around the tic-tac-toe board controlled by a Processing GUI, these were connected via Xbees and then the result of the game would be uploaded to a MongoDB which then displays on a website and the Raspberry Pi live-tweets the result of each game. We had theorised that each Zumo would have 2 lights on top of it to signal which team it was being used by, the light being on would mean crosses or the light being off would mean naughts. As the separate Zumos may cross paths if they tried to go to another square and the other Zumo was in the way, we also had the idea that a ‘bump’ function could be introduced. This function would involve the responsibility of reaching a certain square would be passed on to the other Zumo. Along with this the proposed switch of control would be signified by the light on top of each Zumo switching colour depending what its previous state was. The immediate issue with this was that the Zumo board could only had one GND (ground) socket per board, so we could not attach more than one light on there without the addition of a breadboard. So instead of the proposed double light on top, we opted for a single light that just turns on and off rather than alternating lights.

Early in the project we had a few notions that we eventually came to find wouldn’t work or could be done differently. One of these is we thought that we could get each Zumo to be controlled by their own separate pair of Xbees, however, later in the project found that this wasn’t necessary and could use one Xbee that would just switch connection between each Xbee that was attached to the Zumos. Another was the aforementioned plan about the light switching to signify what player a Zumo was being used by. One more notion we had was that the original Arduino board we were provided would be powerful enough for us to run our program properly, later it became evident that we required a more powerful board than the Arduino Uno. This is because the full program was using up approximately 78% of the memory on the Uno board and was not compiling properly. Once the board had been changed over to the Arduino Leonardo we had to import the correct serial library for it to work properly and then it proceeded to run our code as intended.

As the Zumos had to direct themselves around the board with the GUI commands, it meant that they needed to know how to get to every other square from wherever their position was on the board. We concluded that we could’ve done this with a 2D array that corresponded to every position but as we needed to begin testing the Zumos straight away, we opted to make one big function that handled all of this within if statements, the first layer of the function is a case statement for the direction the Zumo could be facing at the start of its turn (North, South, East or West), then within that there is an if statement for the current position of the Zumo; then finally there is a if statement for the destination, and within there is all the movement commands necessary for the Zumo to reach its destination. Using this method rather than a 2D array allowed us to quickly begin testing and to integrate other parts of the project. The commands for moving the Zumo included a function that moves the hardware over the line which decrements a counter value set outside of the function and another function that turns the Zumo 90 degrees left or right to make it face the desired way.

From the beginning of this we wanted to increase the scope and difficulty of this project, especially as we were using hardware that we were already familiar with (Zumos, Xbees etc). To do this we introduced the Raspberry Pi, MongoDB and a Website. The website would be used as a front-end display of the results of each game and the players involved, this and MongoDB come hand-in-hand as the results displayed on the website will be read from a MongoDB which is updated by Processing once a winner is announced. The Raspberry Pi can easily be integrated into both the MongoDB and the website as it will be handling a twitter feed, meaning it will be automatically ‘live-tweeting’ the results from games as they finish. This will be done by reading the results of each game from the MongoDB as it gets updated, then this twitter feed can be put into the website as a feature. The scripts I wrote to make the twitter feed work are written in Python, and although it was a good learning experience having to pick up a new programming language in the midst of this project, it turned out to be an issue as the JSON object that the MongoDB puts out can’t be parsed in Python. So, as it stands a tweet declaring the winner consists of the contents of the JSON object containing who the winner is without it being parsed and therefore just showing the intended tweet. Getting the Raspberry Pi to actually get started and to output anything at all was arduous enough, due to the extra equipment required to actually use one (monitor, keyboard, mouse, SD card, etc) so it was frustrating for the Pi to not actually be able to parse a JSON object. However, if I were to do this again I would try and code it in C++ rather than Python as C++ has the ability to parse a JSON object, unlike Python.

There have been some issues throughout this project, one of these arose during the production of the project that involved us not being able to use the parameters being sent from Processing, therefore meaning that we can’t actually command the Zumo via the GUI reactively. As an alternative to demonstrate the functionality despite the technical issues regarding passing the parameters from Processing, we opted to hard code some moves, this way we can properly showcase the rest of the project despite this issue. Prior to this we also had problems properly assigning what team each player would be, it involved us not being able to set the slider in the GUI post initialisation and then commanding the slider to switch between naughts and crosses. Another issue was with how we wanted the ‘bump’ function to work. Initially we intended for the Zumos to be close to each other before the player responsibility switch happened. However, this would require going through every if statement for every position, destination and direction it would be facing and then initialising when in the sequence of moves the hand over should happen. So as to save time, the switch over happens at the start of the Zumo’s move and the Zumo that would finish the move goes there as soon as the command is sent, rather than halfway through the original Zumo’s movement. This and a couple other things would be what we would perfect and improve if we had more time and resources to ourselves. But as this is a proof of concept and a show of what we can do, we decided our time would be best spent getting a few different ideas and functionalities working rather than perfecting one or two areas and then sacrificing the ability to showcase a basis of what we can do.