**Group Report**

The inspiration for our assignment came from an older project titled RobotCake a C#, Arduino, Kinect Robot that follows your gestures (CodeProject, 2013). In this project the developer uses an Arduino much like ours however the robot they use is more sophisticated allowing them to control the robot with much more precision. This project proved that the idea for this assignment was at least viable however we did not yet have any idea how the kinect would move the robot and calculate speed/direction or how they would communicate to each other.

For this project we set out with two basic aims. The first was for us to write a program so that the Kinect would detect the movement of a user and pass a message to the Arduino. The second was for us to program the Arduino so that it would interpret the commands sent from the Kinect and control the Zumo. We had an aspirational aim of creating a visual on a pc or mobile application displaying a map of where the Zumo had travelled. How or if we have met these aims will be described below. We utilised the following components: 2 laptops, a Zumo shield, a Microsoft Kinect, an Arduino Uno, a X-Bee Shield, 3 X-Bee radio modules and its corresponding software.

We achieved the first basic aim by sourcing an example of a C# program which showed us how to get the Kinect to display a camera feed and map the skeleton to a user stood in front of it. This came from a project titled KinectStreams and became the basis for our assignment (Pterneas, 2014)(See code snippets 1, 2, 3, 4). We then used this to detect the movements of these joints. We adapted this source so that when the joints are in a certain position it sends a command to the Arduino for example when the right hand Y position is greater than the right hip Y position it sends a command to the Arduino of turn right. We originally intended to transfer data over Bluetooth however we decided to change this as we discovered it would be possible to have this functionality using X-Bee shields which we had prior knowledge of as we had used it during another project. It sends a message using a serial port such as a usb. However, to prevent the serial port becoming overwhelmed, we only send the message if it is not the same as the last. The speed and the command are sent in the same message, an example message would be a,50!. The command contains an end marker character, in this case a ‘!’, for use later when splitting the message in MQTT.

We achieved the second basic aim which was for the Arduino to accept and interpret the command sent from the Kinect to control the Zumo. To do this the Arduino reads in the command sent from the X-Bee in two parts. The first part it reads is the command letter this will either be w, a, s, d or x and the second part will be the speed at which it should carry out the command. First of all, we use the command letter part of the message to set the direction of the Zumo motors and then we use the speed to set the speed of the motors. However if the speed in the message is greater than 125 we set the speed to 125 to prevent damaging the motors.

We had an aspirational aim of mapping on a screen where the Zumo had travelled but we were unable to get this going to due to the inaccuracy and unreliable compass that the Zumo has. Instead we have messages sent via MQTT to a smartphone, so the user of the phone knows what movements the Zumo is currently making. An example of a message would be ‘Zumo moving forward’. In order to implement this, we have set all the X-Bee’s to broadcast mode so that when the X-Bee which is sending messages processed from the Kinect they are sent to both the Zumo X-Bee and another X-Bee so that we can use these messages to transform it and then we can send it over MQTT.

References

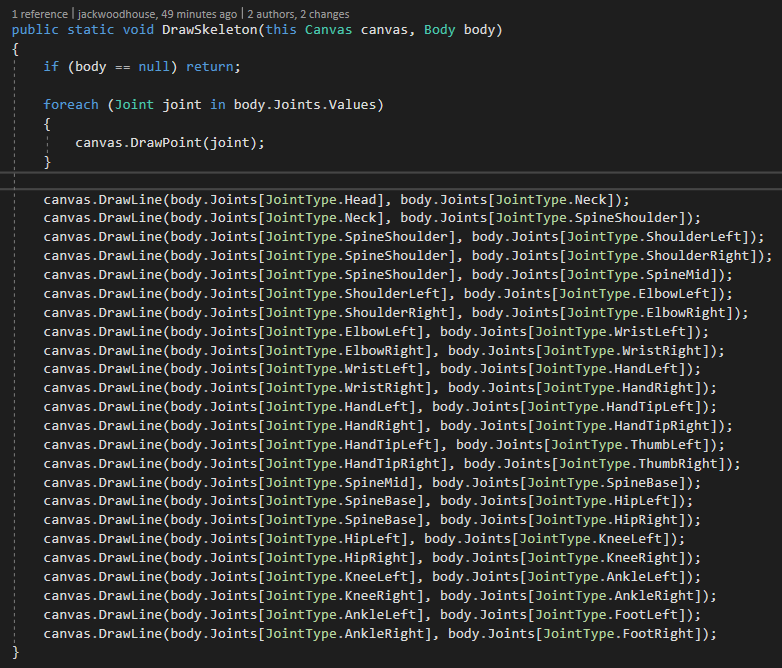
<https://pterneas.com/2014/02/20/kinect-for-windows-version-2-color-depth-and-infrared-streams/>

<https://www.codeproject.com/Articles/672336/CakeRobot-A-Csharp-Arduino-Kinect-Robot-That-Follo>

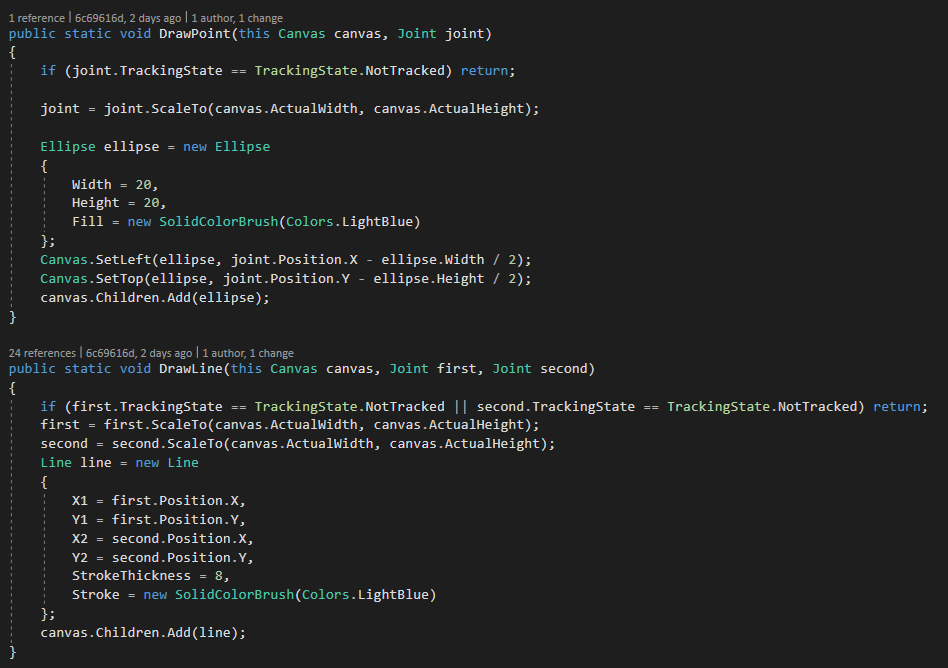
<https://github.com/amazedsaint/cakerobot>

Code

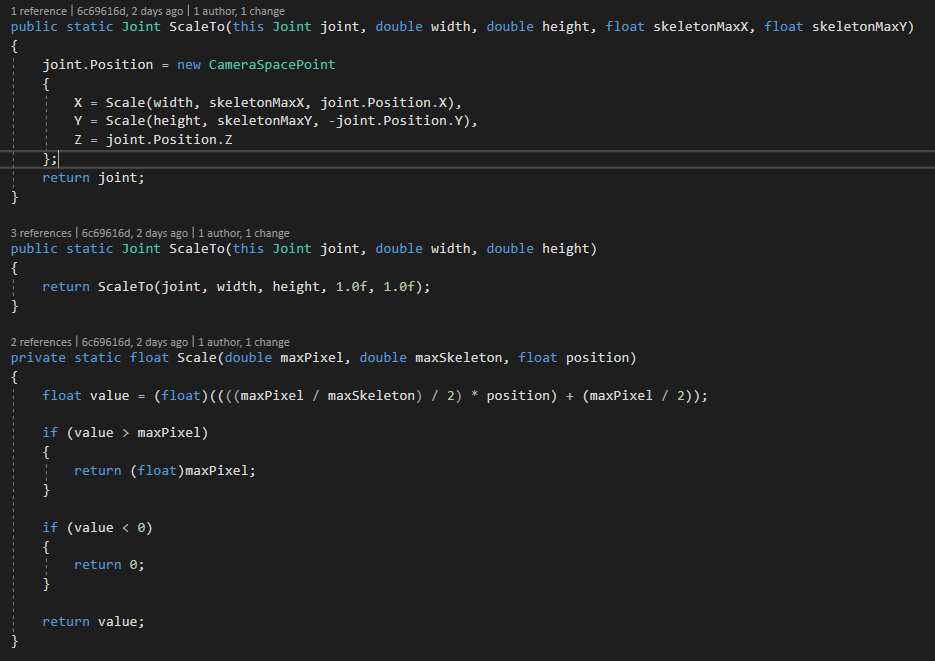
1.



2.



3.



4.

