Report on Arduino Deliverable 2: Communication

This report will explain our reasons and include the stages of implementing a prototype suit which is capable of sensing close objects and aiding the visually impaired.

At first, we wanted to create a device which incorporated a sensor that could determine if a person was close to the wearer and alert the wearer of the person's presence. After checking the Arduino forum for such a sensor, we decided that we would implement our project with the use of ultrasonic sensors. In our first meeting we managed to successfully compile code that could detect when a person 30cm away from the wearer by using the speed of sound multiplied by the time taken for the speed of sound to first bounce off the object then return to the sensor. This method calculated twice the distance so in our code we had to divide the distance by 2 and store the result in a float. We further proceeded by adding an if statement saying that if a person was less than 30 cm away from the wearer to have a servo apply pressure on the person notifying the wearer of the person's presence.

From that point onwards, our idea transformed slightly as research from [1] indicated that there was more potential to our project because as opposed to being used for sensing fictional villains, this technology could actually be used to benefit people who are visually impaired. When running tests on our ultrasonic sensor we noticed that there was a significant delay in the sensor reacting to a faster incoming object and there were anomalies with the change of distance of the object, as when moved quickly the servo would vibrate. We then took to the internet to look for more accurate ways of measuring distance and came across proximity sensors.

Our calibration of proximity sensors was different to our calibration of ultrasonic sensors as the sensors had to be connected to the analogue pins hence the result was in volts. With the aid of [2] to convert the result into Centimetres we had to approximate the distance corresponding to the voltage returned by plotting a graph of the 3 voltages of the proximity sensor against the 3 distances away from a stationary source (the object). Our reference voltage was 5V and our reference distance was 26cm. Hence, our formula for calculating an arbitrary distance was (26*1024)/(5*amount of volts received by the proximity sensor) where 1024 is constant representing the maximum amount of volts which can be read by an Arduino.

After calibrating the proximity sensors, we then found that that there was a smaller delay in data as the speed of electromagnetic radiation is greater than the speed of sound, however the vibrations in the servos were still present. At that point we didn't think much of this ambiguity and moved on to implement a button on one Arduino connected to a servo and a proximity sensor and a similar set up excluding the button of the other Arduino. The reason for including a button was that we realised that different individuals using our system may have different preferences on how they received the data. The idea here was that the communicating Arduino would send the state of the button to the other Arduino whilst checking distance and implementing the servo or the LED simultaneously and the other Arduino would interpret the state implement the servo or the LED depending on the state of the button and the distance recorded. Insight from [3] ensured that debouncing was handled and then we had two working modules.

The next stage of this project involved attaching the Arduino modules onto clothes to enable the surrounding environment to be sensed. We decided to use the Arduino Lilypad as it was portable and could be sewn into clothes. However, when we finished sewing the Lilypad onto fabric we realised that we had to import the (the library) to enable communication as the Lilypad didn't support serial communication. When testing our modules with the servos and the distance sensors we realised that the output power wasn't large enough as it reduced the range if the proximity sensors and the servos weren't powerful enough to be felt by the skin. Thus, we had to make the trade-off of using the heavier Arduino Uno.

Our design at this point changed yet again as now we wanted one Arduino module (Mainboard) to be in charge of the proximity sensors and the servos connected to the person’s shirt indicating the direction of an incoming object less than 30cm and another Arduino module (Controller) to include a button (to change modes) and four LEDs, which when lit indicate the direction of an object. When a proximity sensor detects an object, it checks the current mode that the controller sent. If it sent the first mode then it moves the servo to apply pressure on skin. If the controller Arduino sent the second mode then the Mainboard would send data to the controller Arduino to tell it to turn on a specific LED. To code this, an array storing the servo values and proximity sensor values had to be coded to avoid code duplication

We first used serial communication to send data to and from the Arduinos however, there were detected anomalies for instance the servos would shake violently. It turned out that using serial communication was less efficient, therefore we then started to use master slave on a set of servos and proximity sensors to ensure that our code was working. The anomaly was still present, to stop this we decided to include in the code that if the change in distance between 10 microseconds was greater than 20cm then this was to be regarded as an anomaly. This is because if an object is out of range of the sensor the infrared sensor would give out random values. E.g. sometimes it would give a value which jumps from 90 to 25. So, this would be regarded as an anomaly.

Following this, we connected another set of servos to the Arduino board and added them into the code, we did the same for the next servos in an incremental process. One set of sensors would be in charge of the front, another for the two sides and another one at the back. The LEDs on the button board also corresponded to the set of sensors. We then taped the Arduino board to the front of a shirt and sewed all of the components onto the shirt. Our prototype was finished.

In conclusion, our prototype achieved its goal, it can notify a person if an object is in close proximity to them and the person has control of what means of output they’d like to activate. The current device is merely just a prototype, in the future we would like to make the boards communicate wirelessly using Bluetooth. To increase portability, it would be better if we used a smaller Arduino capable of outputting 5V of power and being as compact as the Lilypad. Considering that not all object can be threats to a person (people walking), a speed sensor can be later put into place to notify the wearer of fast objects, for example cars and the strength of the sensors can be improved to increase the range of the sensor. This project has increased our breadth and depth as computer scientists as we encountered and solved many problems in this project as well as using knowledge from other disciplines to reach the end prototype.

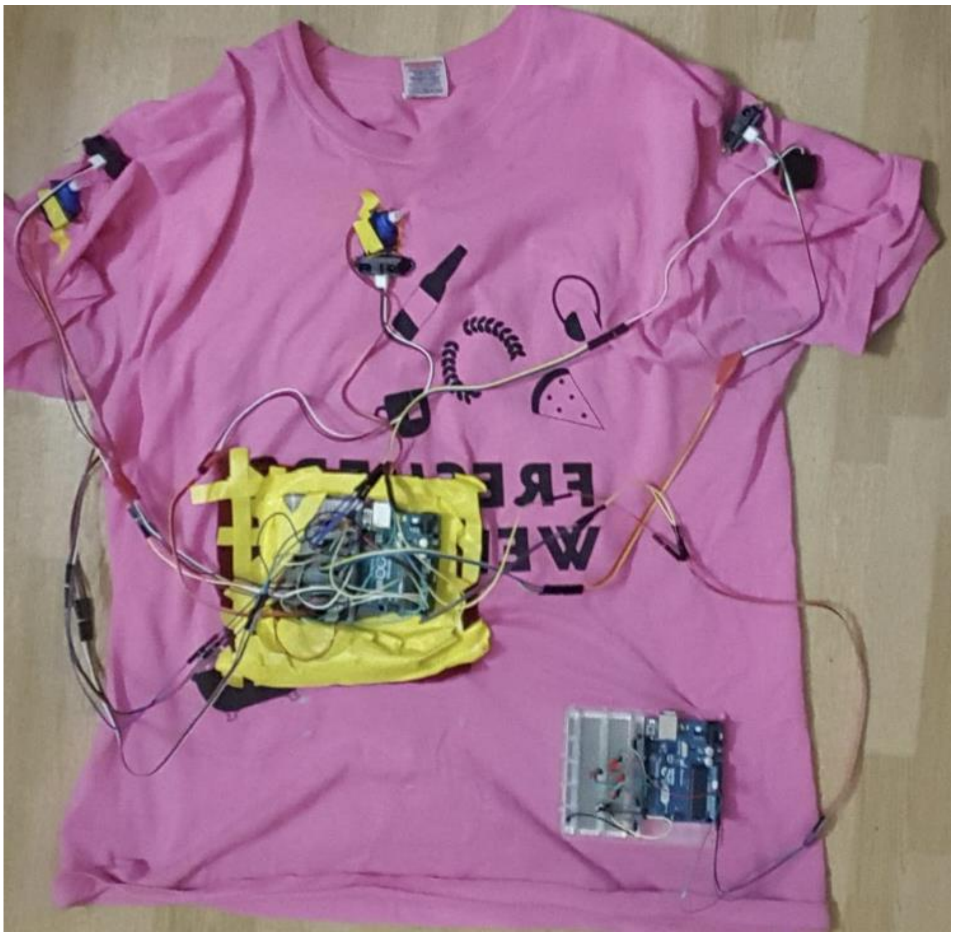
Appendix

Reference List:

1) *Azhar Muhammed,2018. Third Eye for The Blind Available from:* *https://www.hackster.io/muhammedazhar/third-eye-for-the-blind-8c246d [Online] [Accessed 26th November 2018].*

2)Chu Alex,ND. Sharp Sensor Tutorial with Arduino Uno-Learn To Measure Distance. [Online] Available from: https://www.instructables.com/id/Sharp-Sensor-Tutorial-With-Arduino-Uno-Learn-to-Me/[Accessed 27th November 2018]

3)*edu.8s.tv,2015. Arduino Button Debounce Tutorial[Online]. Available from: https://www.youtube.com/watch?v=jYOYgU2vlSE [Accessed 1st November 2018]. (for more information on what debouncing is please look at the previous Arduino report)*



A picture of the project Above. The control unit is the Arduino at the bottom left
And in the centre the distance sensing module is connected to the infrared proximity sensors and the servos stuck onto the shirt.