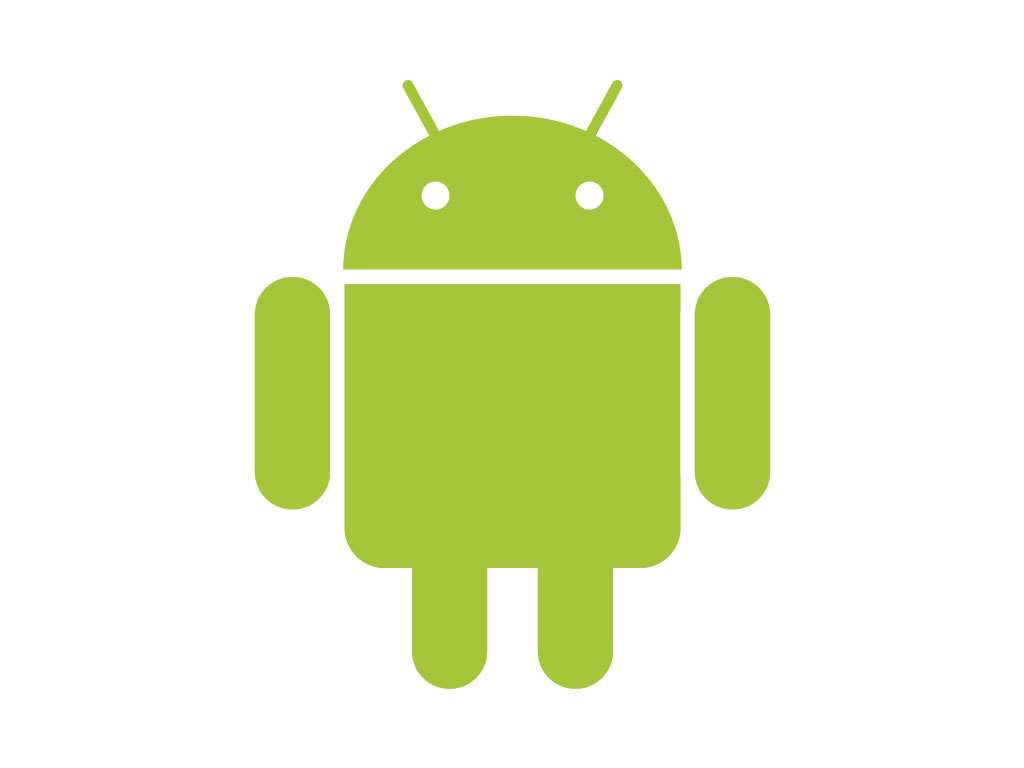


**Group 20 – ACE Phase 1**

**Class:** CS413 Embedded Systems

**Course:** BSc Software Engineering

**Students:** David Leishman, Gavin Donnelly and Michael Gallagher

I confirm and declare this report and the assignment work is entirely the product of my own efforts and I have not used or presented the work of others herein.

Signature: ......................................................................... Date: .........................................

Signature: ......................................................................... Date: .........................................

Signature: ......................................................................... Date: .........................................

Table of Contents

[Introduction 3](#_Toc433103548)

[Assessment of Capabilities 4](#_Toc433103549)

[Arduino 4](#_Toc433103550)

[Raspberry Pi 6](#_Toc433103551)

[Development Environment 6](#_Toc433103552)

[Budget Chart 7](#_Toc433103553)

[Plan 8](#_Toc433103554)

[Design 9](#_Toc433103555)

[Function 9](#_Toc433103556)

[Design Considerations 9](#_Toc433103557)

[Hardware Design 10](#_Toc433103558)

[Software Design 12](#_Toc433103559)

[Arduino Software 12](#_Toc433103560)

[Android App 12](#_Toc433103561)

[Control Scheme 13](#_Toc433103562)

[Using a Phone as a Controller 13](#_Toc433103563)

[Phone Interface Considerations 13](#_Toc433103564)

[Additional Design 16](#_Toc433103565)

[Movement Tracking Considerations 16](#_Toc433103566)

[Final Chosen Design Diagram 17](#_Toc433103567)

# Introduction

The brief of the project was to develop an embedded system. This could should be unique gadget like and use an Arduino or Raspberry Pi or combination of the two. After brainstorming a couple of ideas the group came up with a robotic arm. The basic idea is this would use an arduino to dictate movement and servos in each of the joints. The user can then control this arm with their smart phone to move and pickup items.

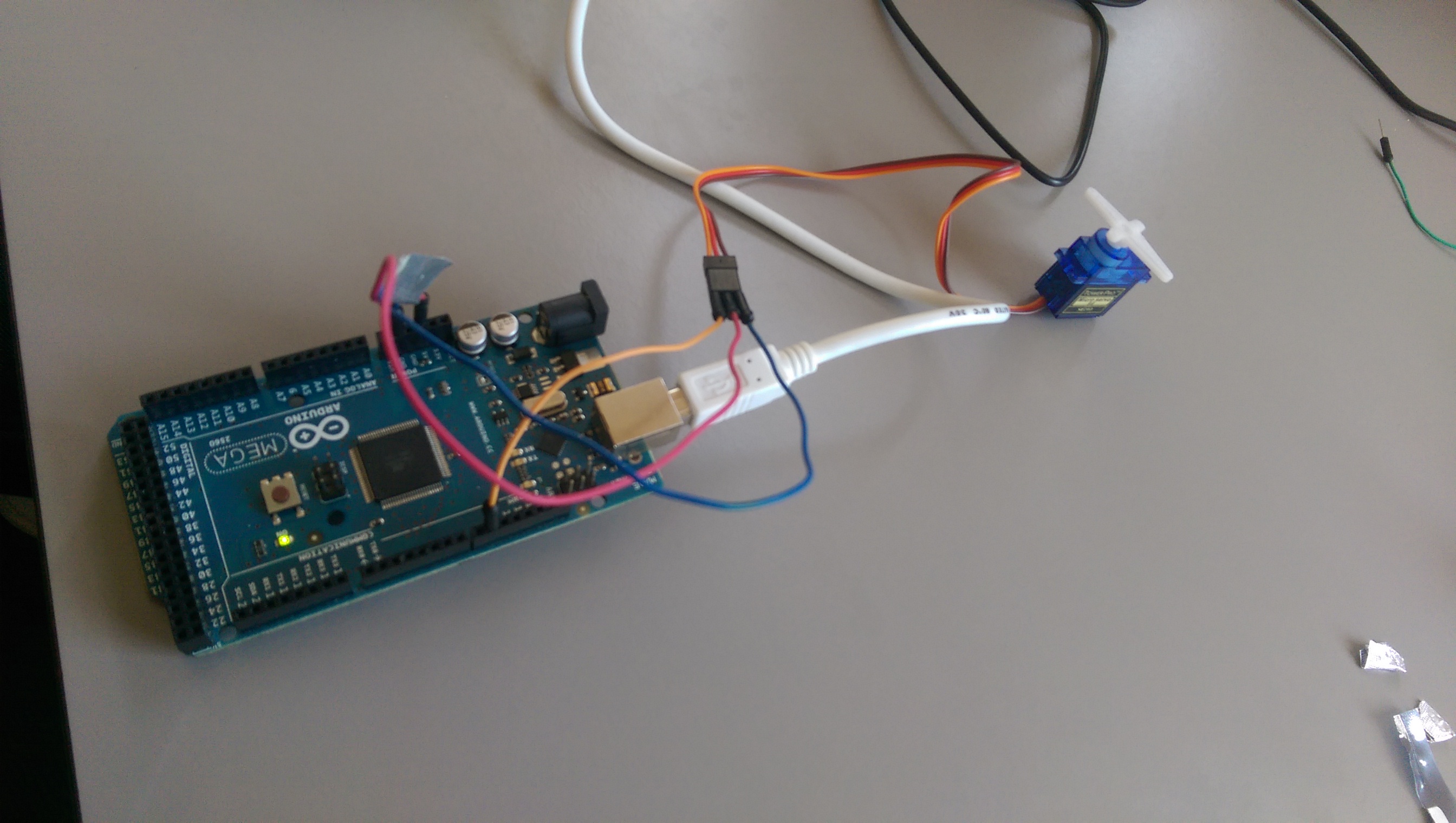


Figure : Early Testing of Servos

# Assessment of Capabilities

## Arduino

An Arduino is a small microcontroller based kit that is used to build and develop digital and interactive objects primarily to sense and alter the physical world. The first Arduino board was built in 2005 and aimed to be a low cost solution for hobbyists and professionals to experiment with creating devices that interact with or alter the real world using sensors and actuators.

The boards use either various forms of 8-bit Atmel AVR microcontrollers or 32-bit Atmel ARM processors at their core. They provide a set of digital and analogue I/O pins that can be used to connect to various controllers, sensors, or extension parts. Some models include USB so that they can load programs from various personal computers.

They are programmed using Arduino’s very own IDE which includes support for C, C++ and Java programming languages. Programs written in this IDE can be uploaded and stored onto the Arduino’s on-chip flash memory. This simplifies development by allowing the use of any general purpose machine instead of an external system to program.

All of the software systems and hardware used by Arduino is open source, and as such the circuitry can be replicated or have its capabilities expanded by an experienced circuit designer. Its software libraries can be expanded via the use of various C++ libraries.

Arduino boards have been used by people all over the globe to create gadgets and systems, and are simple enough that both professionals and beginners to access its features. The capabilities of the system are limited only but the users experience with the system and imagination. They have been used to create things as complex as bipedal robots, world clocks, remote control lawnmowers and even an entire gardening ecosystem.

The Arduino supports and has access to almost every type of sensor available including thermostats and accelerometers, as well as a variety of motors including servos. As long as the power supply is strong enough, the project’s complexity is not limited by any hardware.

The Arduino is much more lightweight in terms of hardware when compared to the Raspberry Pi, and the nature of our project does not require the raw power a Raspberry Pi provides. As such we decided that an Arduino board would be used exclusively over the Raspberry Pi.

|  |  |
| --- | --- |
| Microcontroller | [ATmega2560](http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf) |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |
| Length | 101.52 mm |
| Width | 53.3 mm |
| Weight | 37 g |

Figure Arduino Official Specification Table

## Raspberry Pi

The Raspberry Pi is a small low budget computing device that allows users of all experience levels to delve into the world of computing. It is essentially just a miniature version of any other personal computer, and has the same capabilities such as plugging it into a monitor, TV or other USB controlled devices.

It can also do almost anything a PC can do in regards to software as well. A user can load an operating system like Linux onto it then use it like a normal PC, allowing the browsing of the web, and even playing games so long as the hardware is good enough.

It was developed with the intention of providing a low cost solution for educational institutes to teach basic computer science.

The Raspberry Pi is mainly limited by its hardware in terms of comparisons to a regular PC. The processor runs at around 600-1000 MHz, which is equivalent to the Intel Pentium processors, and the GPU is comparable to that of the original Xbox. Given the size, these statistics are very impressive.

Since most of the hardware, such as the memory, is built into the board, it cannot be replaced or added to. This limitation is due to the foundation’s goal of providing the device at as low a cost as possible while also maintaining a decent set of hardware capabilities.

The power supply used in the Raspberry Pi is independent of the system, and thus it depends on the application that dictates how much power will be needed. Around 2.5A is recommended by the foundation if you want to make sure you can use it to maximum efficiency.

We initially conceptualized using the Raspberry Pi in order to communicate with the robot arm via a phone interface using Bluetooth or Wi-Fi. We discovered after our initial research, however, that Arduino boards have components available to

## Development Environment

Our development environment will be composed of Arduino’s native IDE for software, as well as a combination of project management and repository tools including Trello, GitHub and SmartSheet.

# \\studenthome.cis.strath.ac.uk\homes\system\Windows\Desktop\Capture.PNGBudget Chart

Figure : Budget Chart

# Plan

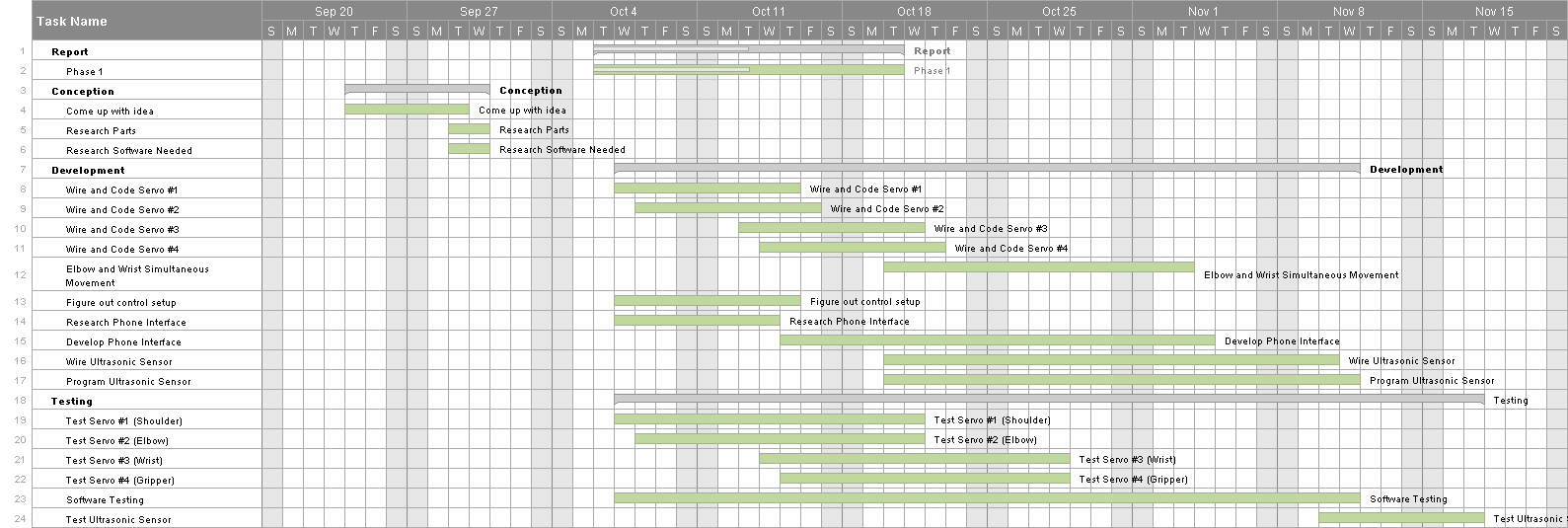


Figure : Gantt Chart Plan

# Design

## Function

The physical arm itself, it must be able to rotate 360 degrees as well as lift small objects of various shapes. The arm should be flexible enough to grab hold of objects that are relatively close as well as be able to stretch quite far out. The initial concept also includes the ability to detect and react to movement, meaning the arm will follow an object moving in front of it. The arm must also be controllable wirelessly from a mobile interface.

## Design Considerations

**Multiple Moving Parts**

The robotic arm is going to have multiple servo motors controlling each movable part of the arm. There are two main things to take into consideration when thinking about this part of the specification.

1. Control Scheme. Since the robot is more complicated than just an X and Y axis, special consideration needs to be given to the way the user will control the robot. An interface that clearly labels the part and how it is controlled is crucial. Some ways this could be done are through a standard keyboard input scheme, accompanied by a graphic that shows which key controls which part.

We also thought of the idea that we could use a mobile interface and control it either via

Wi-Fi or Bluetooth, both of which are easily accessible by a Raspberry Pi or an Arduino. The interface would include a basic graphic of the robotic arm, with selectable joints that could then be controlled either by dragging or tapping arrow keys.

1. Complicated wiring. Having up to five servo motors as well as a Bluetooth or Wi-Fi adaptors may become complicated in the hardware aspects of the project. Luckily this whole worry is alleviated by the fact that Arduino has servos specifically developed for use by the Arduino.

This means that at most we will need to extend some of the servo wires in order to reach the desired part of the arm.

## Hardware Design

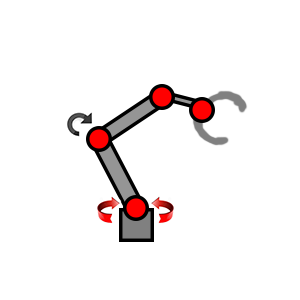


Figure : Servo Diagram

The arm was initially conceived to be made up of 5 servo motors, each acting as a joint on the arm. At the base the servo rotates the entire arm 360 degrees in either direction. At first we did not consider that the servo motors we initially bought would not be strong enough physically to be able to turn the weight of the rest of the arm, and opted to upgrade to a more powerful one when it came time for construction.

The second servo, “the elbow”, will move up and down and will be built in a way that allows the arm to bend in a way that allows it to pick up objects close to it. Again consideration must be made in order to ensure that the servo is strong enough to move the rest of the arm up and down.

The third servo, near the gripper, will act as a simple pivot allowing for more refined and precise movement. In terms of hardware it will be no different from the rest, but in order to maintain precision when controlled, it will have to undergo some experimentation with speed.

The final hardware consideration when considering servos is in the gripper. We have two options when designing and building this part, and they are as follows;

1. Make one half of the grabbing part stationary, and simply close the other half using one of the smaller servo motors. The main benefit of doing it this way is that the servo will follow the implementation of the others, and won’t need many changes as the code can be re-used. We can also save a bit on costs by not needing the extra servo. The main downside however is that only having one servo will severely limit how much weight can actually be lifted, as most heavier objects will simply not be gripped hard enough by just one motor.
2. The second option is combining two separate servo motors, allowing both sides of the “hand” to grip. The obvious advantages here are that it will be a lot stronger in terms of what it can lift. Some disadvantages include the fact that this will introduce the need to move two servos simultaneously which means the code for the other servos cannot be re-used. It also introduces a new challenge in terms of physical design, in that the arm will need to be big enough to accommodate for the two servos.

The last piece of hardware that we have thought of in our design is the ultrasonic sensor. In terms of hardware the actual implementation is not that hard, but its physical design influences the way we put it together. For example, the sensor needs to be placed in the centre of the arm somewhere, and also needs to be close enough to objects so that it will be able to detect movement. This means it either needs to be fixed in the centre of the grippers claws, which may cause obstructions, or on the outside of one of the claws, meaning the claw will need to set vertically rather than horizontally as initially planned.

The final solution to this problem will require us to try out both ways as well experiment with the effective range of the sensor.

For the actual physical parts we have chosen to use Meccano. Originally we thought of using K’nex as the main building blocks but after thinking about the servo weight issue we came to the conclusion that K’nex would also have some issues with weight. For this reason Meccano and super glue is a better option.

# Software Design

## Arduino Software

The software for the robot arm will be mainly using the Arduino C++ based language. The basic setup of the code is simply a busy/wait that waits for input send via Bluetooth from the phone interface, and the servos will react accordingly.

Each servo will have an assigned variable and the servo phone will send a message stating which servo has to move. Perhaps the most difficult aspect of this part of the design is how the Arduino will handle continuous messages. For example, how will the Arduino handle the user continuously holding down the arrow key? After some initial research it was discovered that the Arduino Bluetooth serial adaptor can handle this, as it operates using an input stream and does not close until the user specifies. This means it can continuously receive commands and move until the user stops pressing the button.

## Android App

For the control on the mobile phone side we will build an android app, there are a couple of options for building this app.

We need to choose software option that can give us the capability for the user interface we need and can also send and receive the Bluetooth signals effectively. This gives us a couple of options.

1. Googles android studio, this is the core android app development studio, this will give us all the options we need in terms of user interface and Bluetooth connection and will be a mix of Java and C mainly, so could be our best option in terms of languages we are experience with.
2. Python, there is the option to write python scripts, but we are unsure how stable these will be with a Bluetooth connection or how much flexibility we will have with GUI.
3. Web page, we could run a local web app on the phone, this would be affective but not as stable as the other two solutions.

# Control Scheme

## Using a Phone as a Controller

The robotic arm will have some sort of interface built into a mobile phone. This means that whatever hardware we use will need to have some way to communicate with the phone, via either Bluetooth or Wi-Fi.

1. Bluetooth – Bluetooth is a technology used to send data over short distances. Bluetooth uses a master and slave model, with the master device being able to send data to the slave. The roles can be switched if both devices agree, meaning that both linked devices can act as a receiver or transmitter. The limitation of this is that only one of the devices can send data to the other at the same time meaning it would be a less effective option if the specification required a lot of back and forth between the devices. There is also very little user configuration options as it is a simple symmetric connection between two devices
2. Wi-Fi – Wi-Fi is an alternative solution to Bluetooth and was designed as a replacement for high speed cabling in larger networks such as work offices. Wi-Fi is different to Bluetooth in that all traffic is centred on an access point from which all traffic is routed. Wi-Fi is also user configurable meaning it can be set up for several different purposes and supports much higher speeds than Bluetooth.

One of the main factors we took into consideration when making the decision between the two was cost factor. Wi-Fi transmitters and receivers are both quite expensive. We looked at what we needed – a (probably) simple message sent from the phone controller to the receiver on the Arduino board, so we made the decision that Bluetooth would be better for our needs. Using Wi-Fi would be too expensive and heavyweight for what we need.

## Phone Interface Considerations

The task of creating a user interface on an android device seemed simple at first but when it came down to actually thinking about it, it was quite difficult to come up with an interface that is intuitive to the user.

The initial idea was to simply have arrows depicting the direction to move the arm in, and then have the software calculate which servo’s it was necessary to move in order to reposition the arm in response to the user’s input.

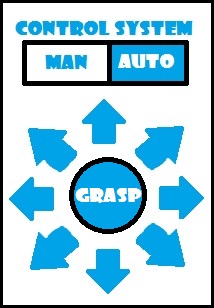


Figure : Initial GUI Concept

We liked this idea for the interface because it is very simple and there’s not a lot for the user to learn. However it was decided that this interface didn’t really convey very well to the user exactly how they were moving the arm, which of course is of utmost importance. It would also be quite complicated to actually take the press of an arrow, and turn it into a movement for the arm. For example we asked ourselves how exactly holding the top-right arrow translates into movements for each individual servo, and it was difficult to provide an adequate answer.

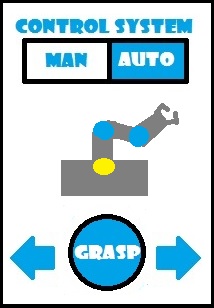
Another idea we had was to have an image of the arm itself, with each servo being selectable, this allowed us to simplify the interface a little bit as each servo can only move in forwards or reverse so once a servo is selected, we only need to provide 2 directional arrows to the user for selection. This also means that once the software receives an input from the user, it can simply move the servo in the desired direction, without needing to consider moving any other servos. The image below shows the idea that the servo at the base of the arm has been selected (highlighted in yellow). The other servos are not selected so they remain in blue. The user could then use either of the arrows to rotate the arm in the desired direction, and they can press the grasp button to close the “hand” to grasp an object.

Figure : Final GUI Solution

In the end, we decided that the second interface better suited the needs of any potential users and was also simpler in regards to calculating the movements of the arm. If the user selects the “Auto” control style, the rest of the interface is locked and the arm will enter the “Movement Tracking” mode, discussed later.

# Additional Design

To enhance the design and features of the arm, we have thought up some additional design mechanics. That can be added on to the core arm features.

## Movement Tracking Considerations

One interesting mechanic that could be added to the arm is movement tracking, where the arm can be put into a mode where it attempts to track object in front of it and move where they move. To implement this we would need a component to do the tracking. Two possible ways are through an ultrasonic sensor wired to the arduino board or a small camera.

1. An Ultrasonic Sensor could be attached to the end of the arm. This means that the Ultrasonic sensor could then be programed to track objects in front of the arm. The code would detect when the ultrasonic sensor is covered and where it is not from an x y axis then move the arm on this axis the opposite direction of where the signal is not bouncing back to the sensor. The Pros to this implementation is the ultrasonic sensor is a reasonable price and could work well. The Con is that it would not have a huge range so if the object moves to far away the tracking would stop.
2. A Webcam could be attached to the board, and we could use OpenCV a library which can support us in developing the motion tracking. The theory of how it would work would be very similar to the ultrasonic sensor. The Pros is that is would be much more detailed tracking and at better ranges, and could even open up the possibility to stream back the live feed to the phone app. The Cons is, it would be more expensive for a webcam and we may need to improve the structure of our arm to hold the weight of a webcam. This also may mean we would need stronger servos to move this weight.

From a user interface standpoint, this feature could be turned on or off in the GUI for the arm on the mobile phone.

# \\studenthome.cis.strath.ac.uk\homes\system\Windows\Downloads\12165845_970424539671122_94837203_n.jpgFinal Chosen Design Diagram

Figure : Final Design