

FUNCTIONAL SPECIFICATION

CCE4999



MOHAMED Al-janabi

Shirwa HAJI

Date: 28/03/2017

Contents

[Background 2](#_Toc478745209)

[Introduction: 3](#_Toc478745210)

[Functional Specification 4](#_Toc478745211)

[User Interaction with the system: 5](#_Toc478745212)

[Limitations: 5](#_Toc478745213)

[How the system will work: 5](#_Toc478745214)

[Positioning of components: 7](#_Toc478745215)

[How the user operates the chair? 8](#_Toc478745216)

[Functionalities Of The Modules? 8](#_Toc478745217)

[Client Running the Test Client and Initializing a Simulation 9](#_Toc478745218)

[Possible Causes Of Error Reading/s 9](#_Toc478745219)

[Calibration: 10](#_Toc478745220)

[Visual Display Of Error Readings & Chosen User Interface Format 10](#_Toc478745221)

[Spreadsheet displaying dummy data in Excel 11](#_Toc478745222)

[Recording of measurements 11](#_Toc478745223)

[Possible causes of errors and impediments in chair simulation 12](#_Toc478745224)

[Possible case scenarios 12](#_Toc478745225)

[Hardware related issues 12](#_Toc478745226)

[Software related issues 12](#_Toc478745227)

[Interconnected Software and Network Connectivity issues 12](#_Toc478745228)

[Errors in Network Connectivity 12](#_Toc478745229)

[TCP 12](#_Toc478745230)

[Possible solutions 12](#_Toc478745231)

[What the User sees 13](#_Toc478745232)

[Errors in Sensor values 13](#_Toc478745233)

[Visuals 13](#_Toc478745234)

[Conclusion: 13](#_Toc478745235)

# Background

This project will provide a 6DOF motion platform for Middlesex University. We believe the most elegant way to achieve this is through the use of a robotic configuration called a Stewart platform, first used by Eric Gough in 1954 and later published by D Stewart in 1965. A Stewart platform consists of six independently controlled linear actuators, all mounted to a fixed base and the movable platform.

# Introduction:

This project is about designing and building a software and hardware system that provides information on real time performance of the 6DOF platform (the Middlesex University 6DOF chair). This system will measure, capture and display differences between actual movements and the desired movements, as controlled by motion simulators and test software.

Desired accuracy is for static translational measurements to nearest millimeter, rotation to nearest 1/2 degree.

Dynamic orientation is to be at least every 2 seconds.

In our project we will measure x, y and z angle and translational motion in x, y and z direction and give these values to the controller named as Monitor.

For measuring the angle and the translational position, we are measuring the length of 6 actuators attached the base. Their length relative to each other decide the angle and position of the chair in x, y, z direction. As for measuring the length of the actuators, we are using 6 sonar sensors, model MB1043, one on each actuator.

These sonar sensors are capable of reading with a resolution of 1mm, which is our required value and working range up to 5000mm. We are using 2 Arduino Mega 2560 boards to take the readings from these boards. 1 board takes the value of 3 sonar sensors.

After that, these values are sent to a third Arduino R101 via a I2C communication. From Arduino, we send these values to a software module in computer via a serial communication. After importing these values in a software code named as Monitor, we will calculate the error.

The user will receive a table of readings in the form of a spreadsheet. This will be used to list actuators (displacements and orientations) and the readings taken. This is readings from the commanded values, to what the chair actually does.

# Functional Specification



By independently controlling the length of each actuator based on some very complex math, the platform can precisely move with 6 degrees of freedom (forward/back, left/right, up/down, and rotation about each axis). This is ideal for motion simulation as it provides a much more realistic experience than the typical 2 or 3 DOF simulator.

Our project is based on a variation of the Stewart platform that uses rotary actuators rather than linear actuators. We built a small Arduino-powered prototype to help us better understand the geometry and mathematics involved in making it the platform move where we want. Aside from the physical construction of the platform and the complex mathematics involved with driving the actuators, the other key step is to setup an interface between the control electronics and the software used for simulation.

For measuring the angles and the translational positions, we are measuring the length of 6 actuators attached with base by means of sonar sensors. Their length relative to each other decide the angle and position of the chair in x, y, z direction. As for measuring the length of the actuators, we are using 6 sonar sensors MB1043, one on each actuators. A program written in python language will calculate the error between the commanded values (coming from middle ware) of orientation and the actual values (coming from sensors).

## User Interaction with the system:

In order for the user to interact with the whole project the user has to start the middleware first then start the chair and this will be explained in details in another document. But in our case the user that is basically a Test Client will in the form of simulation containing information of the coordinates at which the chair will move.

These simulations will first go to the middleware and the middleware will give commands to the actuators for adjustments in the position of chair according to the desire of user, now these values will be passed on to the Monitor module where an error will be calculated between the given coordinates and the measured ones.

An error message will display on the Monitor for the user in the form a simple graphical user interface that can be in the form of graph or excel sheet. But it is difficult to show error values in run time on excel sheet that is why we are using graphical representation for displaying the error values continuously.

## Limitations:

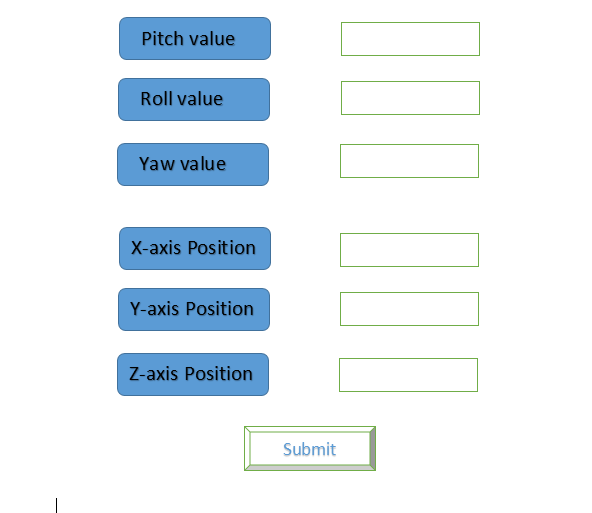
The limitations in the design occurred just because of the sensors used because the processing speed or time required for a sensor to become active is 80-96 msec that is why we cannot increase the processing speed or measuring speed of the sensors and actuators because both values are inter dependent. Once we will receive the values from the sensors and actuators then we will be able to compare them with the values or co-ordinates coming from the Test Client. The maximum dynamic orientation speed we can achieve is 96-100 ms.

## How the system will work:

When a client will ON the controlling device, a Test Client module will run and chair will go to maximum values in 5 degree of motion, taking 5 step for each. First the chair will test the pitch. It will reach at maximum positive value of pitch taking 5 steps and after it will again come to center in 1 step. Then it will move toward negative peak value in 5 steps and again come to center.

System will test 5 degree of motion in same fashion taking 5 steps for each, following below sequence. After each step platform will come to the center position.

1. Maximum value of pitch
2. Minimum value of pitch
3. Maximum of roll
4. Minimum of roll
5. Reach to the maximum value of forward displacement
6. Reach to the maximum value of backward displacement
7. Peak value of rightward movement
8. Peak value of leftward movement
9. In the end of test system will go at maximum height in 5 steps and come back to its center position.

After testing, a dialog screen will pop up where he will be asked to give the value of orientations, 3 angles and 3 translational positions. After entering value, client will press the submit button. When he/she submits the values, these values will pass onto the Middleware and the chair will go in that direction according to given values.

We are measuring the static orientation (3 angles and 3 translational motion) of chair at every 1s. We measure the translational motion of chair up to nearest 1 millimeter and angle up to ½ degree of accuracy.

We will use three controller boards 2 mega and 1 Arduino 101. 1 mega board will handle 3 sonar sensor and send these values into Arduino 101 which is working as central board. From Arduino 101, we send these values to a software module in computer via serial communication. After importing these values in a software code named as watcher, we will calculate the error by comparing the values by commanded values.

By independently controlling the length of each actuator based on some very complex math, the platform can precisely move with 6 degrees of freedom (forward/back, left/right, up/down, and rotation about each axis). This is ideal for motion simulation as it provides a much more realistic experience than the typical 2 or 3 DOF simulator.

Our project is based on a variation of the Stewart platform that uses rotary actuators rather than linear actuators. We built a small Arduino-powered prototype to help us better understand the geometry and mathematics involved in making it the platform move where we want. Aside from the physical construction of the platform and the complex mathematics involved with driving the actuators, the other key step is to setup an interface between the control electronics and the software used for simulation.

For measuring the angles and the translational positions, we are measuring the length of 6 actuators attached with base by means of sonar sensors. Their length relative to each other decide the angle and position of the chair in x, y, z direction. As for measuring the length of the actuators, we are using 6 sonar sensors MB1043, one on each actuators. A program written in python language will calculate the error between the commanded values (coming from Middleware) of orientation and the actual values (coming from sensors).

## 

## Positioning of components:

The main components of the system are:

1. Arduino R101 (Central Board)
2. Two Mega Boards (Connected with sensors)
3. 6 Sonar sensors
4. Middleware
5. Monitor module

The Arduino 101 board which is the central board will be placed under the chair, it will be communicating with the two Mega boards and the Middleware. The two Arduino Mega boards will be mounted on the back of chair, because these boards should be near to the central board, for serial communication using wires.

The six sonar sensors will be mounted on the fix ends of the actuators. An actuator is like an arm with one fix end and the other one moving. The sensors will be mounted on the fixed ends, in a way that no distortion or error shold occur in the measurement of distance, due to the signals sent by any other sonar sensor.

Middleware is the circuitry, which will not be mounted on the chair. It will be installed on the base of the board, which is like the floor of the 6 degree of freedom motion platform. The reason behind this is because it is a complex circuit, which will enhance the signals coming from the Test Client and then it will give these signals to the actuators, for the movement of the chair.

The Monitor module is like a screen or Monitor which is attached with the Middleware through wiring it has a python script running in it. The Monitor module will show the error messages to the in the form of a real time graph. The graph will show the difference calculated between the actual values and the measured values of the actuators.

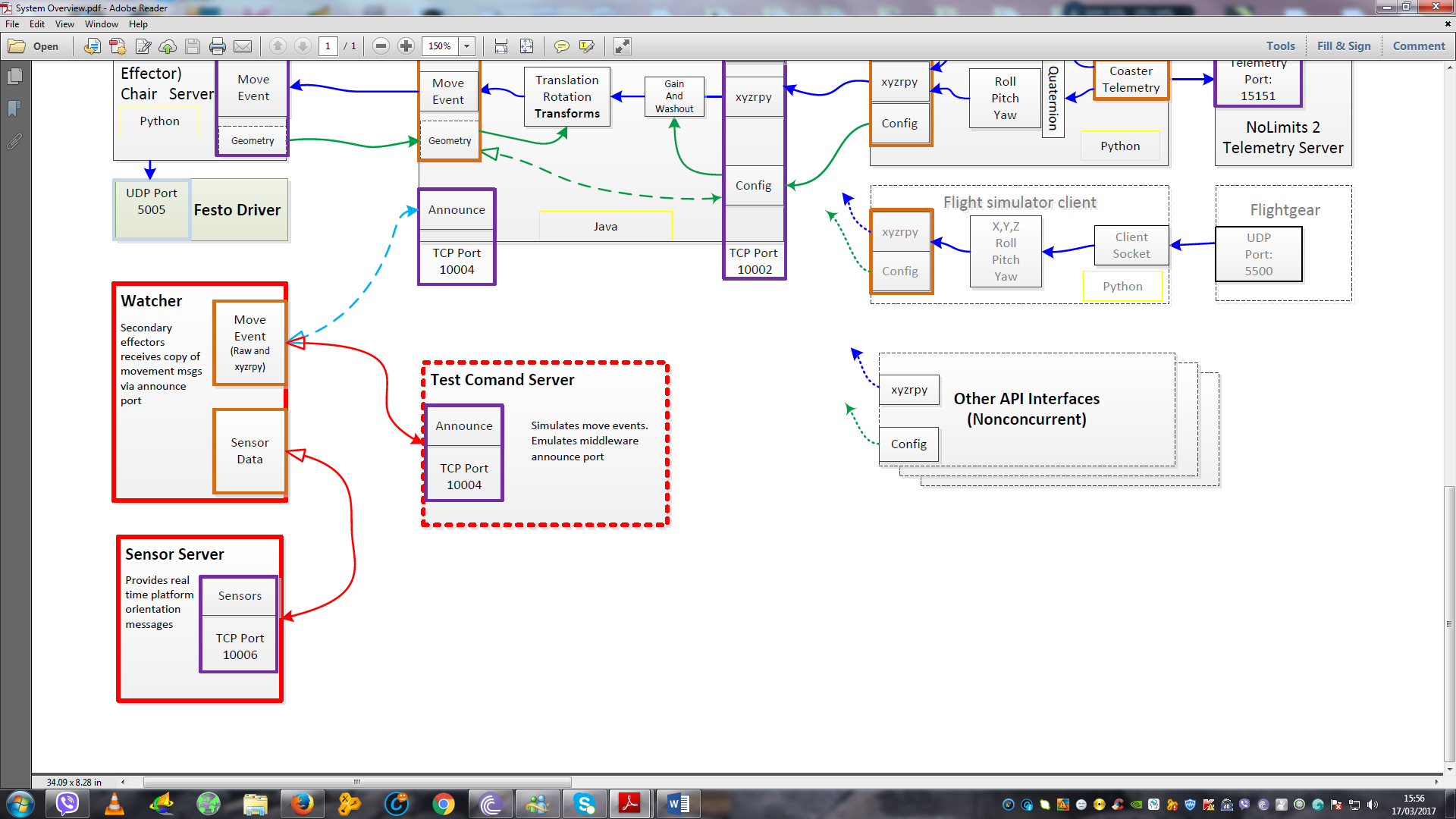
The placing of the Megaboards have been opted to be placed in such a way that is systematic, convenient and organized. The placing of the components is meant to be stress free, plausible and hazard-free.

# How the user operates the chair?

Still to be worked on

## 

# Functionalities Of The Modules?



The diagram above illustrates the bilateral communication occurring when the 3 modules are in full operation. The Watcher module has now been renamed as a Monitor module. The functionalities of the Watcher module remain unchanged, just the name.

# 

# Client Running the Test Client and Initializing a Simulation

When the Client runs Test Client, it will take the commanded values and the chair will respond to this commanded request and orientate the desired position/s. There will be a slight latency/delay in which the next command is accepted, as the muscles/actuators need to relax and revert to its central position. This is done to minimize and rule out complications and false readings.

The opted time for this latency/delay has been selected for 2 seconds/2000 ms. This will give the system sufficient time to re-adjust and avoid any unnecessary technical difficulties.

# Possible Causes Of Error Reading/s

In the likely event of an error occurring, this will need to be visually displayed, as to prompt the user. There are an enumeration of errors that can occur and these need to be established. These are to be the most likely causes and can range from hardware related to software related issues.

**Such causes can be:**

* Defect in muscle/s (actuators)
* Circuitry issues
* Loose wiring/s
* Power supply issues
* System lag
* System performance issues
* Sensor configuration issues
* Actuators are controlled by pressure, therefore they are sensitive if a particular pressure applied, can result in a sudden change of value
* The distance of the actuators vary depending on the weight of the individual
* The relationship between the pressure and the distance isn’t constant
* The calculation of length is in error, e.g. there is a problem in the algorithm
* The software which calculates the distances has some form of error
* The software may not be in sync with the actual movements, hence causing potential glitches to occur
* The calculation on how much to move the muscles to get any particular thing, requires knowing what the distances are
* One or both of the slave devices are unresponsive

There can also be unknown causes which cannot be limited to the above. Only the possible and most likely causes have been mentioned.

These all have to be considered and attended to immediately. These are deemed to be realistic and recurring issues, in which there has to be a favourable system in place, which will visually output this error display and allow the user to run system diagnostics.

## Calibration:

In order to calibrate and we make sure the chair is operating optimally and properly, a beta test will be performed. This test will facilitate the use of small, medium and large weights. 50kg, 70kg and 90kg weights would seem reasonable. 3 tests would then be performed.

The reason for the calibration would be for the use of error correction. The weight applied to the chair, can affect the amount of pressure applied, to orientate the necessary movements.

For each of the sensors, we will need to know what the measured distance is and what the total distance is. E.g. from the attachment point to the end of the target. The relationship between what the system is commanded to do and what it actually does, is affected by the user, e.g. for a heavier user, the system is lowered.

Secondly, that the pressure necessary needed to move that, which is the relationship between pressure and distance is not linear. This is when a curve occurs, between the amount of pressure and the actual distances.

The relationship between them is very approximate. The pressure applied is what is causing the movement of the chair.

One of the things that could be done, which can solve a number of problems in our software, would be to run a calibration software. This would be used to move the chair, where all the muscles are relaxed, increase the pressure incrementally and slowly, maybe a 100 steps.

This would be used to measure the distances. Furthermore it could then use that, when given a command to go a particular distance, and we can look up in the table what pressure to apply. This would ensure the calibration is correct.

The advantage of that is that we can have a table for each of the 3 weights. Once we know what the sensor readings are, from no pressure to maximum pressure. We can then run our test software and command it to go to certain distances and we can measure to see if it actually does that.

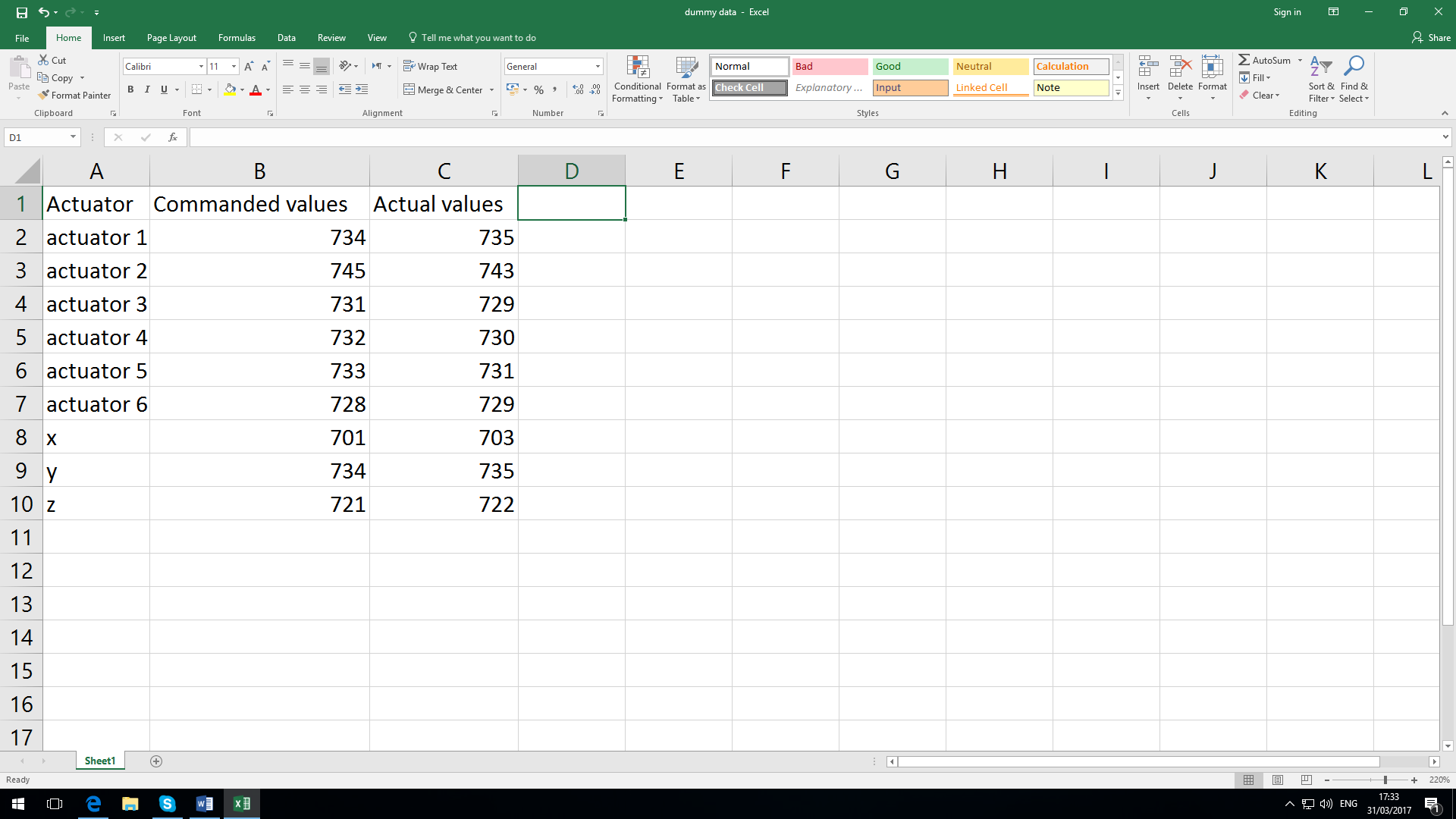
When we’re calibrating with no pressure at all, we can make sure if the chair is level. If the chair isn’t level and then there is no pressure, then we may need to adjust the chair.

# Visual Display Of Error Readings & Chosen User Interface Format

To display the error values, this will be done in a Python shell. These values will also be continually saved in an XML file. One can view the error data in the form of a graph, from the data in an XML file.

The data is intended to be shown on runtime in the form of a graph. Currently, this happens to be quite a difficult process.

# Spreadsheet displaying dummy data in Excel



The spreadsheet above is taking measurements from commanded values and actual values. This is dummy data, in which Yaw has been excluded, as Yaw measurements will not be recorded. No metric has been assigned to the commanded and actual values, as these will be normalized values.

## Recording of measurements

We will need 5 different movements for each degree of freedom. For a displacement, 5 movements from the centre forwards, 5 movements from the centre backwards. 5 movements from the left side, 5 movements to the right side, 5 movements up and 5 movements down. Altogether there will be 55 measurements recorded. There has to be a sufficient number of movements for each degree of freedom. The reason for this is to gather a reasonable amount of data and to test the accuracy, in order to compare from the commanded to actual values.

# Possible causes of errors and impediments in chair simulation

It is common that a variation of errors will be encountered during operation of the chair.

These are mainly:

* Hardware issues
* Software issues
* Network connectivity issues
* Interconnected Software and Connectivity issues

One or many of these problems are likely to occur, if the system is not flawless and has some form of deficiency, which has not been addressed.

# Possible case scenarios

## Hardware related issues

It is possible that hardware issues can become inoperative or suddenly malfunction during simulation and produce a combination of errors. As the chair simulation is delicately designed and structured, a loose wiring could accidentally slip out of it’s input slot, to the degree that the assumption is that the cablings appear to be secured in place.

Another issue that could occur, is that the sensor is producing garbage data. The sensor/s could possibly have a faulty lens interface and hence produce inconsistent results. The sensor could also be incorrectly calibrated and is not in focus. This is just of a few examples.

## Software related issues

If it happens to be that the sensor is producing garbage data, then this means that the sensor is either faulty or there is heavy interference occurring. The Sonar sensors respond to sound waves and if a multitude of sensors are in operation and giving readings, then it can be likely that the target sensor is sensitive and is susceptible to produce garbage data.

## Interconnected Software and Network Connectivity issues

There are also a combination of Software and Network Connectivity issues combined, which involves categorizing the problems under this heading. Such an example is the Monitor is not receiving data from the Middleware. Since we know the Middleware and Monitor connect via a TCP port, this factors in the problem, that the possible system diagnosis is connectivity related.

# Errors in Network Connectivity

## TCP

Errors in TCP connections are possible, if the correct IP address has not been assigned. IP address conflicts are also possible. This will be seen in the form of a dialog box/prompt screen. If prompted, the user should check to see whether the correct port has been assigned.

Another variation can be if the local network is down temporarily. This would involve checking with a technician, as this can be a likely cause of service interruptions.

# Possible solutions

If the Monitor cannot receive data from Middleware as stated previously, then this could be a connectivity issue. This would mean that the network would either have to be reset or the correct TCP port overlooked and set to correct configurations.

# What the User sees

## Errors in Sensor values

If the user finds that the sensor isn’t working, this will usually be evident by the sensor giving false readings or garbage data. The readings will also display inconsistently and with arbitrary values. It would also be sensible to check if the sensors are inserted securely in the Arduino Megaboard serial ports.

# Visuals

Assuming the system is functioning optimally and without any problems, then the user will start a batch, will launch Python scripting and the simulation will initiate. Python scripting will command the values to orientate the chair in the desired positions.

One simulation is complete, then the results will be shown in the form of an Excel spreadsheet. These will list the actuators, commanded values and actual values. It will then be evident to see if the sensors took accurate measurements.

# Conclusion:

This document looks to encapsulate what is needed and what components are required to make this project a success. A project of this scope clearly is highly demanding and requires immense research time and dedication. As the project develops and significant progress is made, the modules and sensors should work in harmony.

It has already been mentioned that the programming for the 3 software modules will be programming using Python scripting. The sensors will be programmed using C programming. The selected use of programming languages was chosen due to simplicity and past minor experience.

However, it was important to address every key element explicitly and comprehensively, in order to harmoniously combine all components. The remainder of the project should unfold and progress naturally, without any hindrances or further difficulties.

This document addresses specifically the functionality of the system. It is important that all scenarios are looked at it. These can be how to operate the chair, from concerns such as errors and malfunctions. These malfunctions normally result in a user being prompted with an error message and a guide on how to resolve.