

Technical Specifications

CCE4999



MOHAMED AL-JANABI

SHIRWA HAJI

Contents

[Background: 2](#_Toc477982319)

[Technical Specifications: 3](#_Toc477982320)

[Calibration: 3](#_Toc477982321)

[Sonar sensor MB1043: 4](#_Toc477982322)

[I2C protocol: 4](#_Toc477982323)

[Monitor: 4](#_Toc477982324)

[How the Client/user will interact with the 6DOF platform? 4](#_Toc477982325)

[Handling error correction: 6](#_Toc477982326)

[Possible sources of errors in readings: 6](#_Toc477982327)

[Modules insight and functionalities 7](#_Toc477982328)

[Software: 7](#_Toc477982329)

[Monitor module 7](#_Toc477982330)

[Middleware Module: 7](#_Toc477982331)

[Test Client: 8](#_Toc477982332)

[Tasks that should be done: 8](#_Toc477982333)

[This is the list of all major tasks included in this project: 8](#_Toc477982334)

# Background:

Our goal is to build a 6 degree-of-freedom motion platform for chair simulation. 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.

This project is about designing and building a software and hardware system that provides information on real time performance of the 6DOF platform (the chair). This system will measure, capture and display differences between actual movement and the desired movements as controlled by motion simulators and test software. 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 to 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 MB1043, one on each actuators.

These sonar sensors are capable of reading with a resolution of 1mm, which is our required value and working range is 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 Uno via I2C communication. From Arduino, 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.

# Technical Specifications:

These are some technical details of our project.

We are transmitting 9 values from which 6 values coming from the sonar sensors and the 3 values coming from the translational motion of actuators in x,y and z direction into the Monitor module via a serial connection. In Monitor, we are also receiving a set of values from the command server. The values that we are receiving from the command server, also contain the 3 angles and 6 actuators length, commanded by the Client.

We are required to measure the values at every 40ms and we can get the dynamic orientation at every 100ms approximately.

Accuracy of the actuator has to be measured up to the nearest 1mm. Dynamic orientation measurement is required to be measured up to 40ms, and we are measuring the dynamic orientation at every 100ms. This is in conflict due to the sonar sensors we are using.

The sonar sensors that we have chosen for this project, give serial readings at every 80ms and the results are displayed at 96ms.

The angle measurements is being measured to the nearest ½ degree.

## 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.

## Sonar sensor MB1043:

After researching about many sensors, we came to know that in our project, sonar sensor

MB1043 is the best fit for us, because it has a resolution of 1mm and working range is up to

5000mm. This sensor is available and affordable. It is also suitable for us as it operates at low voltage from 2.5 to 5.0v.

We have to place 1 sonar sensor on each actuator, to avoid interference we will place sensor on the top side of actuators, so that their direction will be slightly opposite. Further to avoid interference, we may place a card board separation between them.

Sonar sensor sends a high frequency sound pulse and then calculates how long it take for the echo of the sound to reflect, after colliding some object. The sensor has two openings on its front. One opening transmits the ultrasonic waves and the other receives them. Then the time is calculated between transmitting and receiving a wave, by this time and speed of sound in the air, sensor calculates the 2x distance.

## I2C protocol:

I2C is a two wire communication protocol. It is a serial protocol for two-wire interface to connect low speed devices like EEPROMs, microcontrollers, analog to digital and digital to analog converters, Input/output interfaces and other peripherals in the embedded systems.

We can access multiple devices via I2C, just by connecting all of them with 2 wires. Out of two wires, one is SDA and other one is SCL. In I2C, we have master device and multiple slaves. There may be more than one master in I2C bus. Master can send and receive data from the slave just, by accessing them with their address. Each device has a unique & bit address on I2C line. Data is transfer via a SDA in the form of 8 bit packets.

## Monitor:

Monitor is a software program written in python language in the computer. It calculates the error between the commanded values (coming from middle ware) of orientation and the actual values (coming from sensors). In Monitor we get two set of values, one set is of commanded values and other one is measured values from the sensors.

## How the Client/user will interact with the 6DOF platform?

For the project, Python scripting will be used for programming the 3 modules. Python scripting was chosen as it is a low level language and easy to learn. The 3 microcontrollers which are: Arduino Mega boards and Arduino R101 are all to be written in C programming.

In order for the user/Client to begin interacting with the chair, he/she will have to run the test Client module in order to begin orientation of the chair. The 6DOF movements are: Surging, Sway, Heave, Roll, Pitch and Yaw.

The module that is responsible for receiving the commanded values and displaying the actual values is the Monitor. If an error is to be displayed, then this will be in the form of a GUI (Graphical User Interface). This GUI will display the error measurements as to what it should have been.

In monitor we will find the error between the 2 set of values. This error will transmit to a middleware, we will also save this error and have this displayed.

Each reading will be taken at an instant in time, which happens to correspond at a very specific orientation of the chair. 9 readings will be taken, 6 for the actuators, 3 for the angular movements.

Approximately between 8-10 readings are needed for each orientation, as this will need logged and compared from the commanded readings to the actual readings. Duplicate readings are therefore needed for each orientation, to give steady and solid readings.

A column would be represented the commanded position and the actual position. The actual distance is presented by the 6 distances and the 3 angles.

A basic user friendly GUI can be in the form of an Excel file. This would only be used to display shortcomings, such as error measurements. There are 2 forms of errors which could be displayed. This can be software related, which can be tied into the system not functioning as it’s supposed to.

A secondary error message can be due to the difference in commanded movements and actual movements of the chair.

**“JSON”** is a protocol that will be used for display of messages on the monitor. While, “**Xyzrpy”** is the name of a field, that is an array of xyz translations and rpy rotations, which is a Client command.

**“rawArgs”** is the movement field, in which there are 6 values indicating the length of the muscles (actuators), it also has the xyz angles, translations and rotations.

We will need a quantitative measurement, because we need to know what exactly the error is.

{“jsonrpc”:”2:0”, “method”:xyzrpy

* **“Jsonrpc”** is a format
* **2:0** is following the protocol for “jsonrpc”
* **“Method”** is saying what the message is, which is a command
* **“moveEvent”** is what type of method it is, a message which has information about a movement
* **“rawArgs”** is a set of fields, with the six values in the array
* **“xyzrpyArgs”** has the 6 translations

**“rawArgs”** are the movements of the actuators, these do not have a metric assigned, but are known as normalized. This is the relative movement of the actuators. For example 0 is the central position

The actual metric is the real length of the actuators. For example, -1 means the actuator is at its full length. Being normalized, the software which displays the chair, doesn’t care how many mm the actuator is, it only cares about the relative movement.

For the chair to tilt -20 degrees, the muscles need to drop by 20% and other muscles raise (as a form of compensation) by 20%.

## Handling error correction:

* To know the actual length of the actuators
* To know the measured distances are – These will be obtained from the sensors

## Possible sources of errors in readings:

There are a range of issues that could occur in receiving conflicting and incorrect readings.

These will now be addressed:

* 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

# 

# Modules insight and functionalities

## Software:

Software part of this project include these modules.

## Monitor module

* This module should be able to receive the data from central board and the middleware
  + - * The central board will contain all the sensor data
      * The sensors would need to be programmed with accurate coding’s for measurements.
* This should find the error between commanded values and actual values
  + - * This would be used to measure the commanded positions to the actual positions.
* This should save error detection
  + The error detection would need to be created and clearly address the issue at hand.
* This should display the error occurred in the movement of chair.
  + To facilitate a simple GUI for displaying error message/s
    - * Display in the form of an excel file or a graph
* Calculate (AX,AY,AZ) and (DX,DY,DZ) from 6 actuators length
  + The monitor module requests AX,AY,AZ and DX,DY,DZ displacements from the Arduino 101 central board
* To send data back to middleware
* To check that the monitor module has enough capacity to store the relevant data, of both the sensor and test Client

What this module does is that it receives information from the rollercoaster. There is Python module which has the user interface. The Python script generates the x,y,z,r,p,y messages for moving up and down.

## Middleware Module:

The Middleware module is fixated on Java scripting language. The Middleware can control the gain, which is another advantage of normalization. We may need the chair to be more dynamic and every movement amplified. By using normalized values, what we can do is multiple the normalized values by a factor, to increase or decrease its value.

The Middleware is capable of changing the dynamic range, as well as washout. Washout takes large movements, if the motion simulator is doing a loop. It starts the chair accelerating in a given direction, to give the body a sense of that movement.

It also recognizes that the total movement is more than it can do, so it slowly drops the chair back to level again. This is known as washout. This will not be needed in our project, but feel it should be explained, as it is embedded in the Middleware module.

The Middleware does the calculation to calculate for a given commanded position, on what the angles need to be. “**xyzrpyArgs**” are the 3 translations and 3 rotations.

The event above will be sending **rawArgs** as well, 6 raw values. The purpose of the prototype above, is not to influence the design of the system, rather just act as a guidance of what happens.

The prototype above has to be adjusted where the sensor data just being constantly sent.

## Test Client:

A test Client is a simulated version of the input coordinates needed by the middleware to start the orientation of the chair. The user will only have to run this module, it will send the simulations to the middleware and then these coordinates will be sent to the actuators, to adjust their position according to the desire of user.

For testing the system, we are making a Test Client and this will consist of 55 set of values and it will test the system in such a way, that it will take the chair first in one direction, step by step, then likewise in the other direction. It will then re-adjust to the central position. It will continue this testing for all degrees of motion.

# Tasks that should be done:

## This is the list of all major tasks included in this project:

* Measuring the length of actuators
  + - Obtaining the actuators algorithm and measuring the difference (from above the brackets there is a small gap. This will be the remainder of the distance that will need to be measured with a tape measurement)
    - Another means of obtaining a measurement from an actuator is to sample a reading from a sonar sensor. The sonar sensor will focus on a target and the remainder of the distance occupied will be hand measured
* Protection against interference
* Place the sensors in alternate directions of the actuators (vice versa positioning)
* Take individual readings
* Place a partition, like a cardboard or some other effective guard to block interference
* Calculation of angle
* The Arduino R101 will specifically be used to measure angular movements
* Collection of all the data in the central board
* Make sure the Slave devices (Mega Boards) send data simultaneously to the Arduino R101
* Make sure the Slave Boards are sending the correct values
* Make sure the protocol established is working
* Send data from the central board to the monitor module
* This module should receive all the sensor data from the central board
* It should also receive data from the watcher
* This module should have the error correction programmed and saved as a file
* This should compare the two sets of data that it receives and save error correction to file, which can be viewed as excel or another basic user interface
* Display the results
* We are displaying the error data in the form of table in a Python shell
* We are also receiving that data in an XML file
* This data can be used to construct the graph