DESIGN DOCUMENT

Contents

[System Overview: 3](#_Toc478158131)

[detailed design: 3](#_Toc478158132)

[Software design: 4](#_Toc478158133)

[TEST CLIENT: 4](#_Toc478158134)

[Working of hardware components 5](#_Toc478158135)

[Working of Sensors: 5](#_Toc478158136)

[Working of Mega Boards: 5](#_Toc478158137)

[Working of central Arduino 101: 5](#_Toc478158138)

[Working of MiddleWare: 6](#_Toc478158139)

[Design Specifications: 7](#_Toc478158140)

[Calibration: 7](#_Toc478158141)

[Sonar sensor MB1043: 8](#_Toc478158142)

[I2C protocol: 8](#_Toc478158143)

[Monitor: 8](#_Toc478158144)

[Handling error correction: 9](#_Toc478158145)

[Working functional specification: 9](#_Toc478158146)

[Possible Causes Of Error Reading/s 9](#_Toc478158147)

[Modules insight and functionalities 10](#_Toc478158148)

[Software: 10](#_Toc478158149)

[Monitor module 10](#_Toc478158150)

[Middleware Module: 10](#_Toc478158151)

[Test Client: 11](#_Toc478158152)

[Tasks that should be done: 11](#_Toc478158153)

[This is the list of all major tasks included in this project: 11](#_Toc478158154)

## System Overview:

The work in this project is that we will make it sure that chair move exactly in the specific direction according to commanded values. For this we will measure the dynamic orientation of chair at every 40ms. We measure the translational motion of chair up to nearest 1 millimeter and angle up to ½ degree of accuracy.

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

## detailed design:

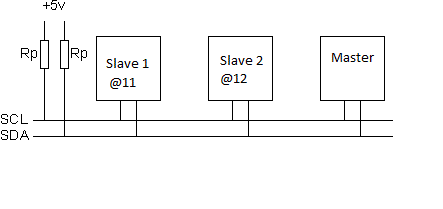
We are measuring length of 6 actuator using Maxbotic sonar sensor MB1043 HRLV-MaxSonar EZ4. 1 sensor is attached to each actuator in such a way that each sensor and its target is attached at both end of actuator. Sensors are capable of reading 30cm to 5000cm accurately, precision up to the nearest 1mm on their serial pin. This is if the sensor is at the bottom end of its target and at the top end of the actuator.

To receive the values from the sensor, it have to be trigger high at pin 4 for at least 20us. Sensor start giving value serially at its serial pin, after 80ms of trigger and complete at 96ms. In the programing, more than 96ms delay is introduced, intending to receive the right value.

The sensor is controlled electrically via a Atmega 2530 board, which is an Arduino board and programmed in C language. Atmega2530 is selected due to availability, of more number of serial port. It has 3 serial ports available on the board, so it controls 3 sonar sensor serially at the same time, at a baud rate of 9600.

For 6 MB1043 HRLV-MaxSonar EZ4, 2 Atmega 2530 boards are being used. These Atmega 2530 boards are connected with a I2C bus in slave mode. 1st Atmega is at address 11 and 2nd is at 12.

They receive command to trigger the sensor from the Master on a I2C bus. The speed of an I2C bus used in this connection is 100K bit/s. After receiving the commands from an Atmega 2530 trigger, all the sensors read the distances serially from all sensors and send back the values of distance from sensors to Master on I2C bus.



Arduino101 is the Master on I2C bus. The reason of selecting this is it has on board Bluetooth low energy (BLE) and gyro sensor. This board give command to slave board Atmega 2530 on I2C bus and receive actuators length values.

This also calculate the 3 degree of angle with the help of on board accelerometer and gyro meter sensors. (Yet to explain) This value is precise up to nearest ½ degree of angle. It also get 6 distance values from Atmega 2530 boards via I2C bus. After getting 9 values it transmit all the data serially to a software running in computer written in Python named as Watcher.

## Software design:

On the software side we have two module written in Python language running side by side. One is the Test Client and other one is watcher.

In the Test Client module, Client values are predefined hypothetically (These value are just for the testing purpose and will be replaced by actual Client commanded values).

Watcher module is meant for getting two set of values one from Test Client and other one is from the central board Arduino101. It access to the Test Client values simply by importing that module. It get other set of values that are actual values from central board, serially. For getting values from the central board, serial port is specified in the programing.

After getting the two set of values watcher compare these values individually and find the difference between them. This difference between the values is actually the error. Watcher send these error values to middleware via TCP port. Watcher also save these error in form of text file in the computer.

## TEST CLIENT:

In the Test Client module, client values are predefined hypothetically (These value are just for the testing purpose). When a client will ON the controlling device, Test Client module will run and the chair will go to maximum values in 5 direction of motion, taking 5 steps for each. First the chair will test the pitch.

For this it will reach at maximum positive value of pitch in 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 in 1 step. For this testing, pre-defined values are fed in the client module.

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

After testing, a dialog screen will open where he will be asked to give the value of orientations, 3 angles and 3 translational positions. After entering value, client then presses the submit button. When he submits values, these values go in the Middleware and the chair will go in that direction, according to given values.

# Working of hardware components

## Working of Sensors:

The sensors we are using in this project are MB 1043. It is a narrowest beam width sensor that is an excellent choice when dealing with the detection of larger objects. It has a resolution of 1 mm and the reading rate is 10 Hz while a 42 kHz ultrasonic beam of sensor measures the distance of object from a fixed point where the sensor has been installed. The sensor operates at 2.5V-5.0V and has 3.1 mA average current requirement that is favorable for our project because an Arduino can handle a device with 5V and a maximum of 40mA of current limit. The operational temperature ranges from 0°C to 65°C and has a real time automatic calibration quality in it. These sensors will continuously measure the values of positions of actuators and transmit them to the main central board but the time constant or the time required by the sensor to become active and then send value to the board will just produce a little delay in the whole process.

## Working of Mega Boards:

The board we are using to communicate with sensors and central board is Arduino Mega because of its more number of ports for serial communication. Three sensors at a time can communicate with Arduino Mega board and two are Arduino Mega boards have been used to communicate with 6 sonar sensors. There is separate code running in these Arduino boards which is mapping the values coming from the sensors to make it useful information for the central board and then this information will be transmitted to the main board containing 6 values obtained from the sensors. The reason mentioned above for using separate Arduino boards with sensors is a major reason but the main reason behind using two Arduino boards just to get values from the sensors and then transmitting them to main board is the increase in processing speed. Because if we use just one Arduino we can perform only one task in one clock cycle but if we are using three boards with same clock periods, we can perform three different tasks at a time in just one clock cycle like the concept of parallel processing.

## Working of central Arduino 101:

The translational motion as mentioned in the project proposal was to be measured after every 40 ms but due to the limitation of sensors used we are measuring the translational motion after every 100 ms. Because the sonar sensors used take about 80 ms to activate and then the total time taken by the sensors to send information to the micro-controllers is about 96 ms approximately. The reason behind this delay in measurement is that we are using real time serial communication protocol to communicate with the sensors and Actuators. In the same time that is 96 ms we are measuring the angular values from the main Arduino used.

The accuracy of measurement of Dynamic translational position is up to 1 mm and the accuracy of angle measurement is ½ degree. 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.

The values received by the two Arduino Mega boards from the sensors are now transmitted to the main board (Middleware) which is Arduino 101. The main Arduino board is like the brain of the system, it is receiving the values coming from the two Arduino boards and in the same time it is measuring the angular quantities. After getting values from the two Arduino Mega boards that is the 6 translational motion positions returned from the sonar sensors, the three angular values and the values from the Test Client, the main Arduino 101 will send these values to the Monitor module.

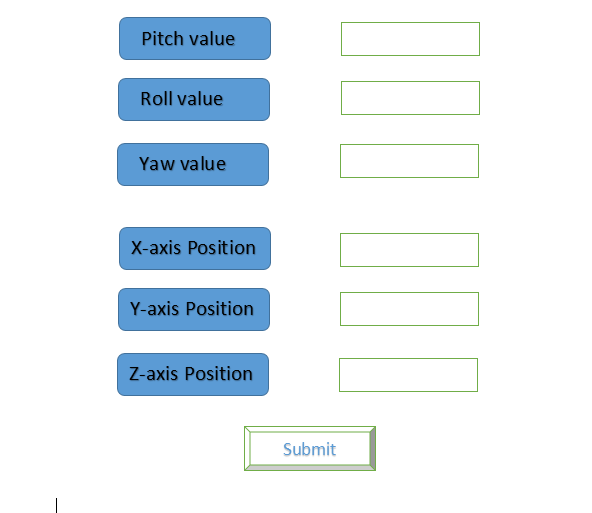
A python script is running in the Monitor module which will then then calculate the error between the given values of Test Client and the measured values coming from the sensors, it will now send the error value to the Middleware so that it can remove the error in the movement occurred.

Hence in this way, the main Arduino 101 is assisting Monitor module in error minimization. The main aim of the project is to show error messages on the display Monitor to the user, but we are adding an additional function in our system that is error minimization. In this functionality, the Monitor will see that if there is any difference between the given values from the Test Client and the measured one’s it is going to send command to the Middleware having information regarding x-direction, y-direction and z-direction co-ordinates.

The actuators will now adjust their position according to the command given to minimize error in the dynamic translational motion. The Test Client in our case will communicate with the Middleware only through a simulated input, giving the desired position of movement of the chair.

## Working of MiddleWare:

A middleware is a main circuitry that is attached with the chair for controlling its motion and to execute the commands given by the Monitor and Test Client. The information given by the Test Client in the form of simulation will be passed to the middleware. The middleware will give command to the chair to move at those specific coordinates. Now when the middleware ware will receive the error value calculated by the Monitor in the coordinates coming from middleware and from the two Mega boards, it will now give a command to the chair to adjust its position in a way so that the error can be minimized. So, basically the middleware is a driving circuitry of the chair.



**Monitor:**

Watcher module is meant for getting two set of values, one from Test Client and other one is from the Central board Arduino R101. It access to the Test Client values is simply by importing that module. It receives the other set of values that are actual values from the Central board, serially. For receiving values from the central board, serial port is specified in the programming. After receiving the two set of values, watcher compares these values individually and finds the difference between them. This difference between the values is actually the error. Monitor sends these error values to the Middleware via a TCP port. Watcher also saves these error in the form of a text file in the computer.

# Design Specifications:

These are some technical details of our project.

## 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.

From these values, we calculate the error value and give it to the Monitor which control the muscles. This is something like a feedback system. I am saying this a feedback mechanism because in feedback system we continuously measure the output and input is a function of output we are doing exactly the same thing here. We measure the current position and angles of chair and compare it to the commanded values, and then find the error, this error decides the input.

## Handling error correction:

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

## Working functional specification:

We are transmitting these 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.

# 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. This can range from;

* Defect in muscle/s
* Circuitry issues
* Loose wiring/s
* System lag
* System performance 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

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 display and allow the user to run system diagnostics.

# 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.

## 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.

# 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
* 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
* Python shell will be used to display the error message/s in the form of an XML file
* Looking to use “Programming”. This is particularly useful for visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping. [**https://processing.org/**](https://processing.org/)
* Another useful way of displaying data is to actually request a log of data to be recorded in Python and constructed in the form of a table.