Palestine Polytechnic University

College of Engineering

Electrical Engineering Department

Graduation project

Title

Electrical Medical Bed for Disabled People

Done By

Shhade Rabee **Eyad Jbor**

Supervisor
Dr. Ayman Wazwaz

Submitted to the College of Engineering in partial fulfillment of the requirements for the Bachelor degree in Engineering

Contents

List of Figure:

List of Table:

Abstract

The project aims to design a smart medical bed that provides adequate comfort for the patient and facilitates treatment for the doctor. It aims to reduce errors in the nature of the patient's movement, this can be done by adjusting the angles of the head and back areas along with the feet area.

The bed consists of two moving platforms and one fixed platform. The moving platforms are supplied with linear actuators to enable the doctor and the patient to control the angular position of these moving platform.

The prototype enables the user to control the angular position of each moving platform of the chair remotely or using buttons attached to the chair.

The bed can also be converted into a chair to facilitate the transport of the patient to any place he wants depending on four wheels attached to the body of the prototype.

الملخص

يهدف المشروع إلى تصميم سرير طبي ذكي يوفر الراحة الكافية للمريض ويسهل العلاج للطبيب. يهدف التصميم إلى تقليل الأخطاء الناتجة من طبيعة حركة المريض ، ويمكن القيام بذلك عن طريق ضبط زوايا مناطق الرأس والظهر مع منطقة القدمين.

يتكون السرير من قاعدتين متحركتين بالاضافة الى قاعدة ثابتة. و قد تم تزويد القواعد المتحركة بمحركات خطية لتمكين الطبيب والمريض من التحكم بالوضع الزاوي لهذه القواعد المتحركة.

يتيح النموذج للمستخدم التحكم في الوضع الزاوي لكل منصة متحركة للكرسي عن بُعد " عبر تطبيق الهاتف المحمول " أو باستخدام أزرار التحكم المتصلة بالكرسي.

كما يمكن أيضًا تحويل السرير إلى كرسي لتسهيل نقل المريض إلى أي مكان يريده وذلك بالاعتماد على أربعة عجلات متصلة بجسم النموذج.

Chapter 1
Introduction

Chapter 1: Introduction

1.1 Overview

In this chapter, a brief description of the system will be presented, starting with the objectives, the importance of (smart bed), hardware and software requirements, and finally the project approach, time charts and part prices, and some pictures that express each topic separately.

1.2 Objective

1. Provide high precision angular position control of the head and feet platforms.
2. Increase patient comfort in all movement situations.
3. Facilitate the interaction between the doctor and the patient
4. Converting the bed into a chair for easy transportation depending on the attached wheels.
5. Avoid medical errors of fractures, wounds and all diseases from the doctor

1.3 Importance

1. Facilitate dealing with the patient.
2. Improving services for patients.
3. Ability to control the chair remotely and using the buttons.
4. To be one of the modern technologies in hospitals in Palestine.

1.4 Requirements

This project needs a programmed system, devices, and tools necessary to achieve the desired goals. It is a hardware project that contains the following components,

- Actuator (Linear motors).
- Mechanical parts.
- Arduino.
- DC drivers
- Sensors
- Other electrical and electronics interfaces.

1.5. Theoretical background

The medical definition of hospital bed is a bed used for patients (as in a hospital) that can be adjusted specially to raise the head end, foot end, or middle as required [1]. Recently, during the Corona pandemic that has swept the world, the demand for these beds has increased dramatically due to the necessity of remote control where the doctor cannot deal with patient directly. As these beds have great benefits in treating patients in general. As a result of most of them being used in hospitals, as happened in Palestinian hospitals .The need arose to design a medical bed model in Palestine in order to support the medical staff. This need did not appear in Palestine only, but medical teams all over the world needed such a bed. So we built a medical bed model that meets the growing needs of medical staff .The bed is designed to

meet all medical needs, which is the possibility to move the head and feet of the patient safely.

The most important features of hospital beds are [2],

- Wheels. Which enable easy movement of the bed, either within parts of the facility in which they are located, or within the room. Itself.
- Elevation. Beds can be raised and lowered at the head and feet.
- Side rails.
- Tilting, some advanced beds are equipped with columns which help tilt the bed to 15-30 degrees on each side. Such tilting can help prevent pressure ulcers for the patient, and help caregivers to do their daily tasks with less of a risk of back injuries.
- Bed exit alarm. Many modern hospital beds are able to feature a bed exit alarm whereby a pressure pad on or in the mattress arms an audible alert when a weight such as a patient is placed on it, and activating the full alarm once this weight is removed. This is helpful to hospital staff or caregivers monitoring any number of patients from a distance (such as a nurse's station) as the alarm will trigger in the event of a patient (especially the elderly or memory impaired) falling out of the bed or wandering off unsupervised. This alarm can be emitted solely from the bed itself or connected to the nurse call bell/light or hospital phone/paging system. Also some beds can feature a multi-zone bed exit alarm which can alert the staff when the patient start moving in the bed and before the actual exit which is necessary for some cases.
- CPR function. In the event of the bed occupant suddenly requiring cardiopulmonary resuscitation, some hospital beds offer a CPR function in the form of a button or lever which when activated flattens the bed platform and

put it in lowest height and deflates and flattens the bed's air mattress (if installed) creating a flat hard surface necessary for CPR administration.

- Specialist beds. Many specialist hospital beds are also produced to effectively treat different injuries. These include standing beds, turning beds and legacy beds. These are usually used to treat back and spinal injuries as well as severe trauma.

This issue of designing smart medical beds attracted the researchers around the world, for example, [3] proposed a FPGA based controller for hospital beds. The main advantage of FPGA controller is the huge processing speed, however it cost is very high. In fact, this application does not require such high processing speed, so the FPGA is not the best solution in terms of cost. In the research [4], the history of designing hospital beds was discussed along with deigning cost of these beds. In paper [5], the authors tried to show some relationships among physical demands with anthropometric factors for normal hospital. Project [6] proposes a cross hospital bed management system for control of occupancy of bed, we design a dynamic system that announces status of bed when it change with admission or discharge of a patient. Finally, [7] proposes a new method in a rehabilitation bed for transporting a bedridden patient.

1.6 Proposed structure design

Based on the mentioned properties, this project aims to design an effective hospital bed that can achieve the universal functions of these beds, in addition to reduce the production cost as much as possible.

To ensure that the proposed design will meet the required specifications, the structure was designed mainly from three parts. The first one is the fixed platform

which represents the base of the bed. The second part is the head section, which is a moving platform and responsible to move the head of the patient. The last one is the feet section, where it is responsible to move the feet of the patient. For each moving platform, three main positions were defined which upper, middle and lower. These locations can be arrived using the automatic control mode as shown in Fig.1.1 to Fig.1.6. While specific degrees rather than the standards, can be set manually using the manual control.

Figure 1 : Case 1 of the bed, completely horizontal.

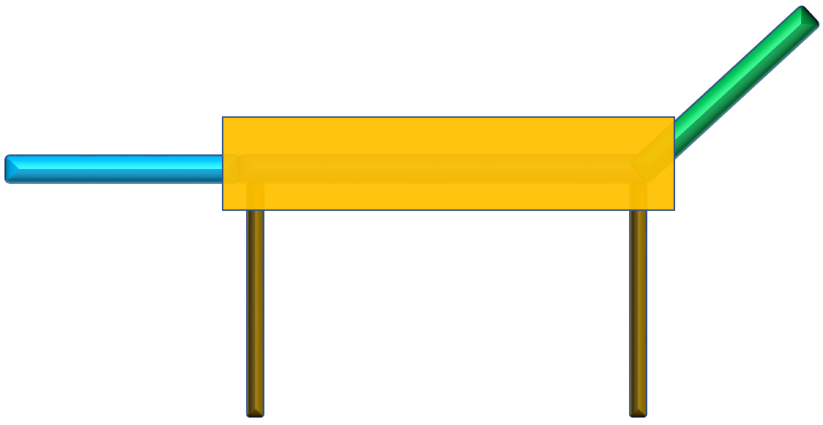

Figure 2 : Case 2 head raised by 60 degree

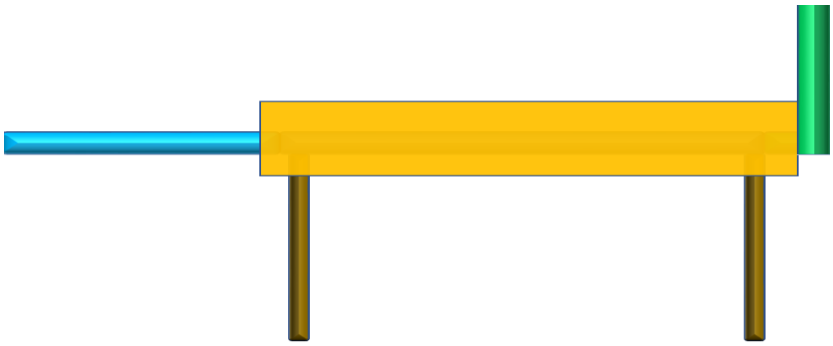

Figure 3 : Case 3 head raised by 90 degree

Figure 4 : Case 4 feet lowered by 60 degree

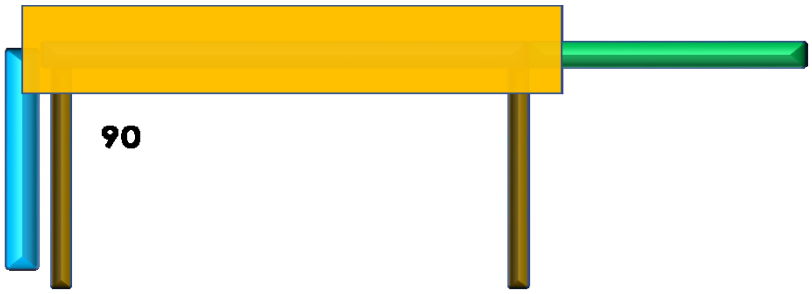

Figure 5 : Case 5 feet lowered by 90 degree

The total cost of the prototype is 985 NIS and the details are shown in Table 1.

Table 1 : Prototype cost

Component	Quantity	Price (NIS)
Arduino UNO	1	40
IBT2	2	100
Linear actuator	2	100
Ultrasonic sensor	2	30
Lm35 sensor	1	5
Human temperature sensor	1	50
Industrial push buttons	4	15
HC05 Bluetooth device	1	40
Bread board	1	10
Connectors	10	2
Mechanical components	1	300
Total cost		985

Chapter 2
Hardware System

Chapter 2: Hardware System

2.1 Mechanical specifications

The project consist of multiple mechanical and electronic components to achieve the desired object. The mechanical part consists mainly from the body which is the stationary part, and the moving platforms which are the dynamic parts. Each moving platform is connected to the body by the mean of linear motor attached to a revolute joint. The dimensions of base platform and the moving platforms are shown in Figure 6.

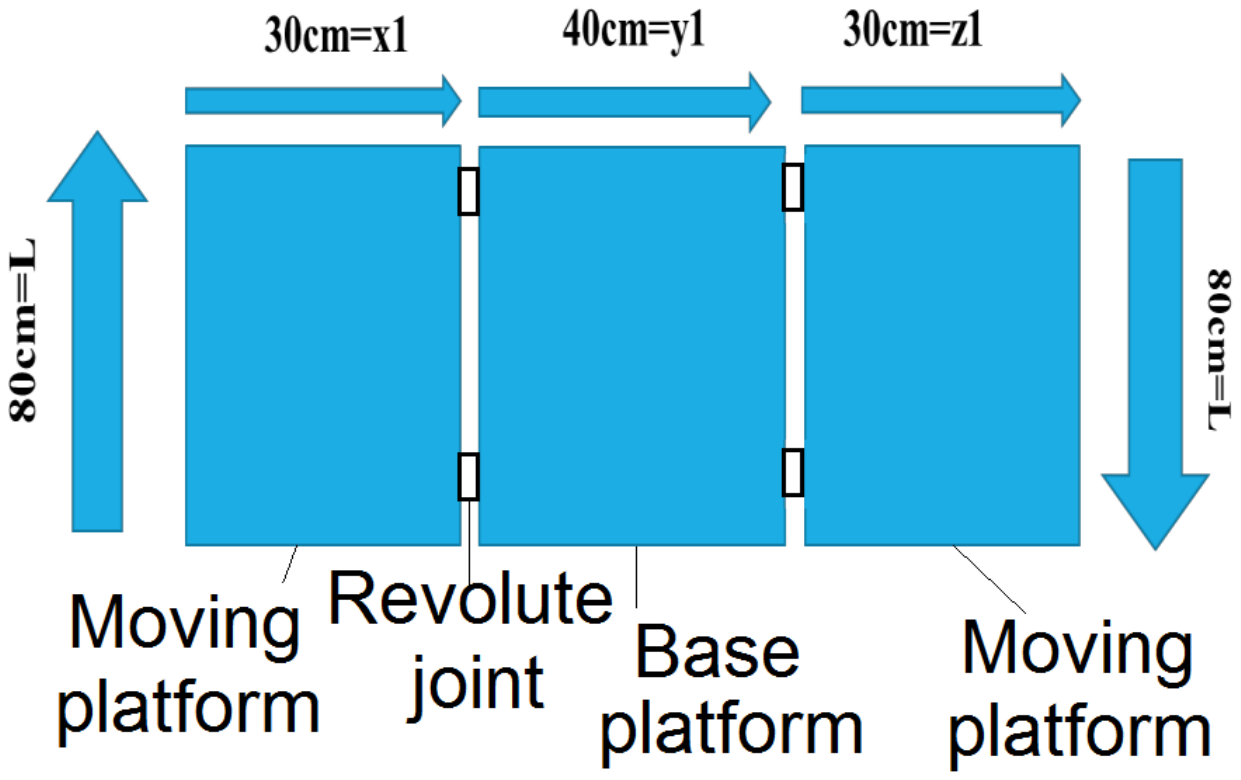

Figure 6: Mechanical parts dimensions.

The chair is supposed to lift an adult human body, suppose that the average mass of an adult male is 70Kg, then with respect to [8] the percentage mass of the total leg is 18.6% of the total mass, which means that the total mass of the leg is about

13Kg, and 26Kg for both legs. The percentage mass of the trunk part is 55%, which meant the mass of the trunk is about 38Kg. As the human body is lift by three platforms, the base platform will share a half of the legs mass and a half of the trunk mass. This is due to the fact that the legs will be held by the lower moving platform and the base platform, also the upper platform will not completely lift the trunk of the body. The overall mass on the lower platform becomes 13Kg and 19Kg for the upper platform. So the selected actuator must provide a maximum force equals to 200N.

The selection of linear motor was based on their ability to produce the required force on their extremities. In this project, the high speed is not required at all. As the high speed may affect the person who sleep on the bed, which almost is a patient. So the quiet and safe motion is required. The HIWIN linear motors was selected to perform the task. This motor is simple, can produce the required force, low cost and powerful motor. In addition, it can be controlled easily. The motor is shown in Figure 7, and its dynamic parameters are listed in Table 2.

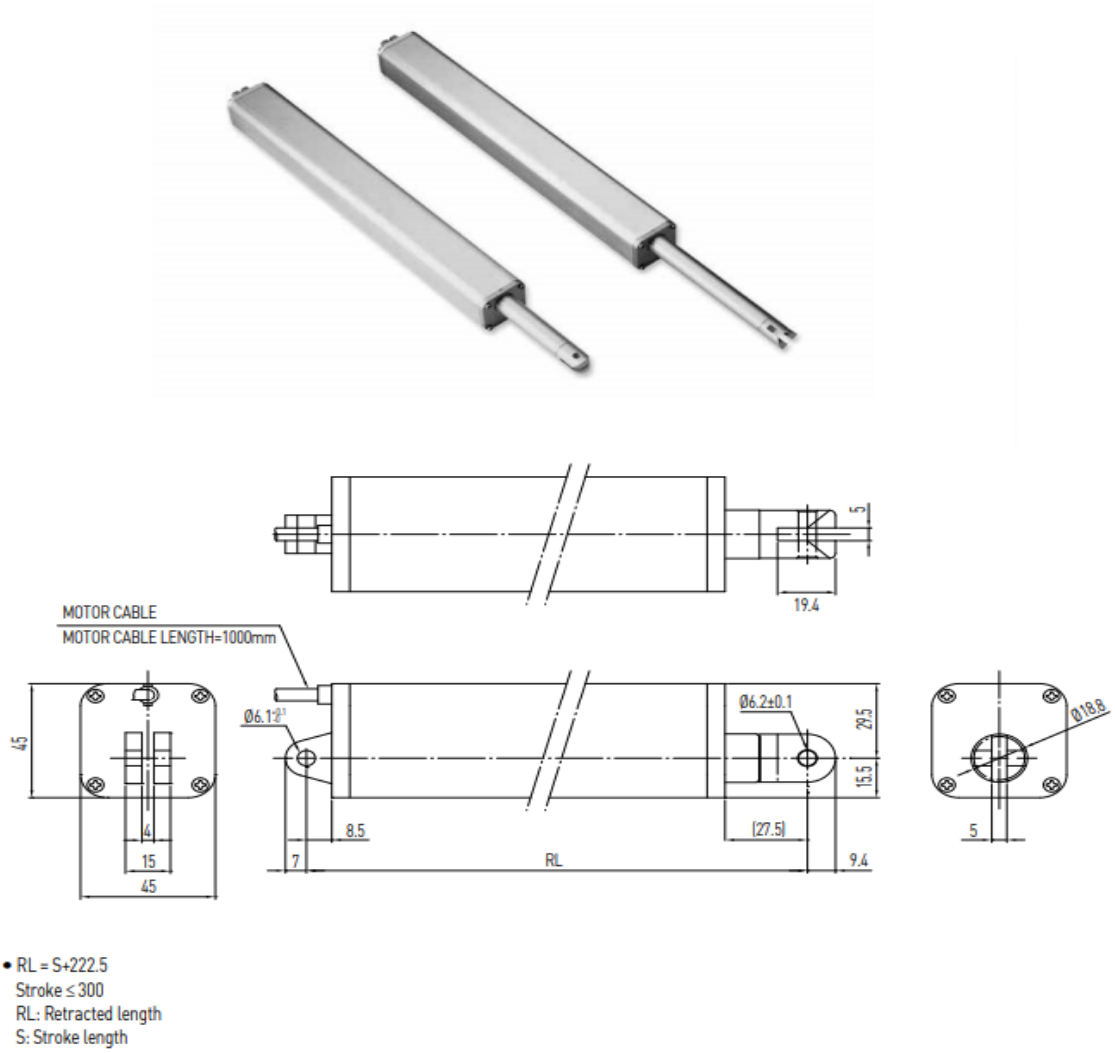

Figure 7: motor and dimensions.

Table 2: Dynamic parameter of the linear motor

Parameter	Value	Unit
Maximum load	200	N
Maximum self-blocking	200	N
Maximum speed	19	mm/sec
Stroke	200	Mm
Rated voltage	12	V
Rated current	1	A
weight	1.28	kg

2.2 Electrical and Electronic Components

The electronic circuit was mainly designed to control the linear motor, and to make them rotate the moving platform to reach the required angular position. To achieve this object, each actuator was supplied with an ultrasonic sensor. Which is responsible to measure the height between the linear motor and the base platform, which is directly depends on the angular position of the moving platform. The basic principle of the ultrasonic sensor is that it emits short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo. As the distance to an object is determined by measuring the time of flight and not by the intensity of the sound, ultrasonic sensors are excellent at suppressing background interference. The principle of work is shown in Figure 8 and the ultrasonic itself is shown in Figure 9.

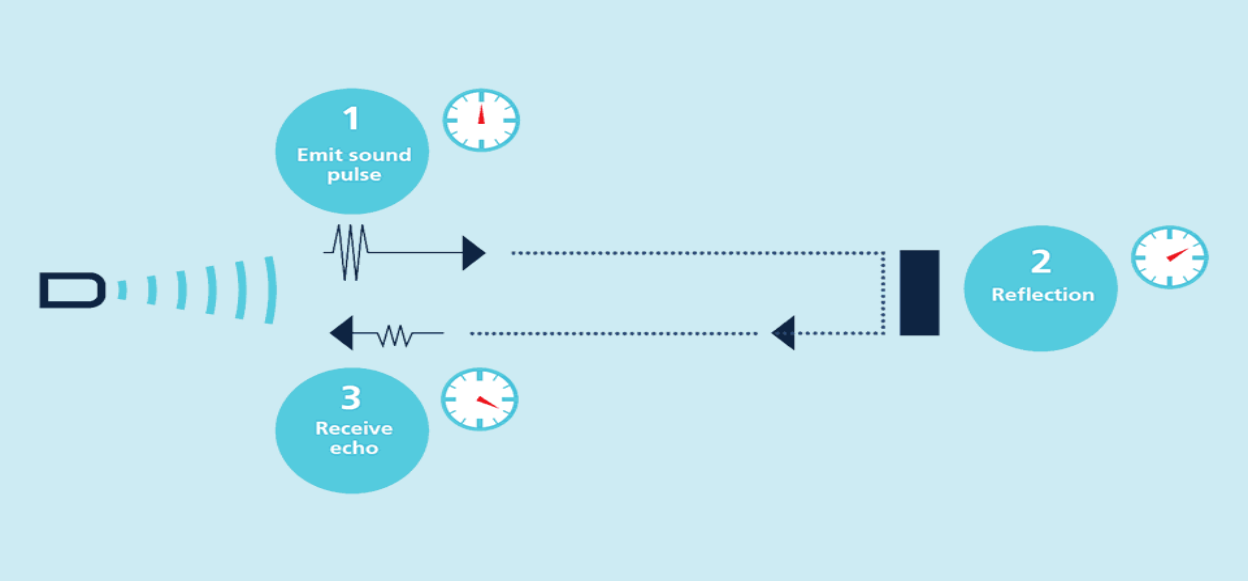

Figure 8: Ultrasonic sensor principle of work.

Figure 9: Ultrasonic sensor.

As the SFR04 ultrasonic sensors provide the angular position of moving platforms, this data is considered as the feedback signal in the control system. Then, to control the motion of the linear actuators a DC interface and driver module IBT2 was used as shown in Figure 10.

Figure 10: IBT2 driver

IBT2 represents an inexpensive, high power motor driver based on two BTS7960 chips. This driver depends on the principle of H-Bride to control the speed and direction of a DC motor by the principle of Pulse Width Modulation (PWM). These features are compatible for the start, stop and reversing the direction of rotation functions which require relatively high current. The control signal delivered to the

IBT2 driver will affect the speed and the direction of the linear actuator as shown in Table 3.

Table 3: IBT2 Control signal.

PWM	1-255	1-255	1-255	1-255
Right-PWM	High	Low	Low	High
Left-PWM	Low	High	Low	High
Direction	CW	CCW	Stop	Burn

The datasheet of this module provides more details such as,

- Output voltage : 6V-27V
- Maximum Current : 43A
- Input voltage : 3.3V-5V

Furthermore, as the user should be able to control the angular position of the moving platforms, four industrial push buttons were added to the system, the first pair of these buttons is responsible to move the head moving platform up and down, while the last pair is used to control the motion of the bottom moving platform. These buttons are shown in Figure 11.

Figure 11: push buttons for manual control

To provide the functions to control remotely, HC05 Bluetooth device was added to the circuit. It aims to transfer data between the hardware system and the software system. It is a module which can add two-way (full-duplex) wireless functionality to communicational applications. It can communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. The module communicates with the help of USART at 9600 baud-rate hence it is easy to interface with any microcontroller that supports USART. The supported baud rate is suitable for this project and the amount of the data that will be transferred. HC05 is shown in Figure 12.

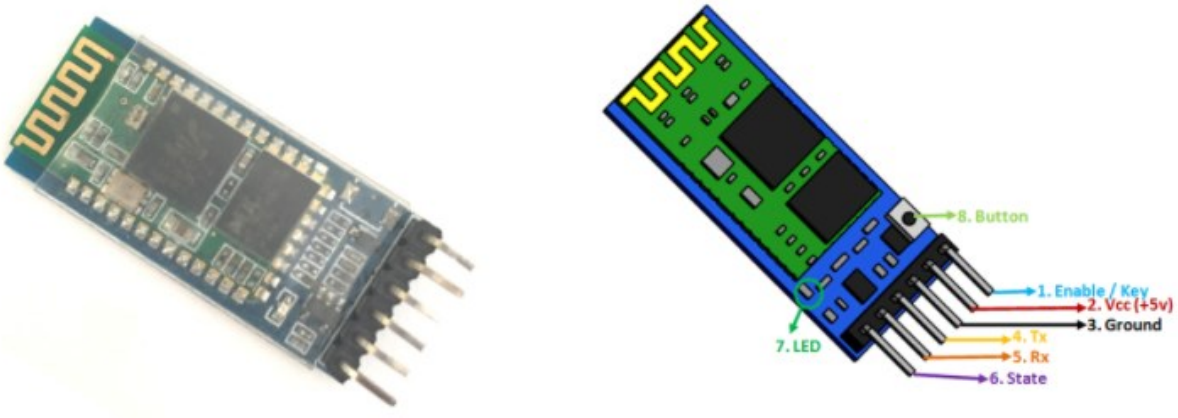

Figure 12 : a: HC05 module b: HC05 pinout.

To provide the function of measuring the ambient and the patient, two temperature sensors were added to control circuit. The first on is LM35 sensor as shown in Fig2.8, which is responsible to measure the temperature of the ambient. The output voltage of the LM35 sensor is proportional to the measured temperature with the following relation 10mV/1C, which means that the equation of this sensor can be written as follows,

$$T=100*V \qquad (2.1)$$

Where T is the ambient temperature and V is voltage measured whit the unit volt.

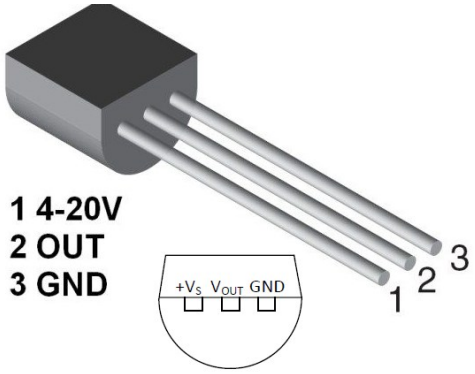

Figure 13: LM35 sensor

The second temperature sensor is thermistor sensor, which is responsible to measure the temperature of the human laying on the controlled bed. A thermistor is a type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors. Thermistors are widely used as inrush current limiters, temperature sensors (negative temperature coefficient or NTC type typically), self-resetting overcurrent protectors, and self-regulating heating elements (positive temperature coefficient or PTC type typically).

Thermistors are of two opposite fundamental types:

- With NTC thermistors, resistance *decreases* as temperature rises usually due to an increase in conduction electrons bumped up by thermal agitation from valency band. An NTC is commonly used as a temperature sensor, or in series with a circuit as an inrush current limiter.
- With PTC thermistors, resistance *increases* as temperature rises usually due to increased thermal lattice agitations particularly those of impurities and imperfections. PTC thermistors are commonly installed in series with a circuit, and used to protect against *overcurrent* conditions, as resettable fuses.

In this project, NTC type was used and shown in Figure 14.

Figure 14: Human NTC sensor.

All of the mentioned component were attached to Arduino microcontroller which presents the main component on the system. It is responsible to read sensors data, control the actuator through the drivers and to perform the all required communications with the external world.

In this project, Arduino UNO was selected to be used as it has the lowest cost of the Arduino microcontrollers family and it supports all required functions, inputs, outputs and interfaces as shown in Figure 15.

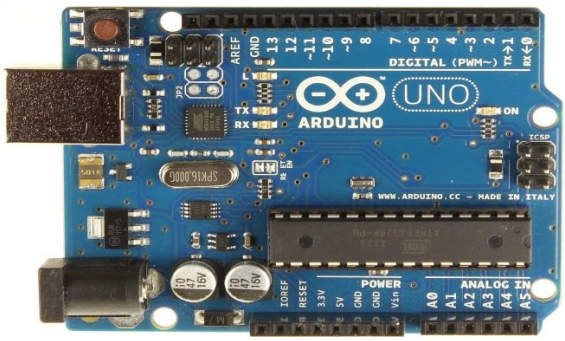

Figure 15: Arduino UNO.

Finally, the overall circuit diagram for all components and how they are connected to the microcontroller, is shown in Figure 16.

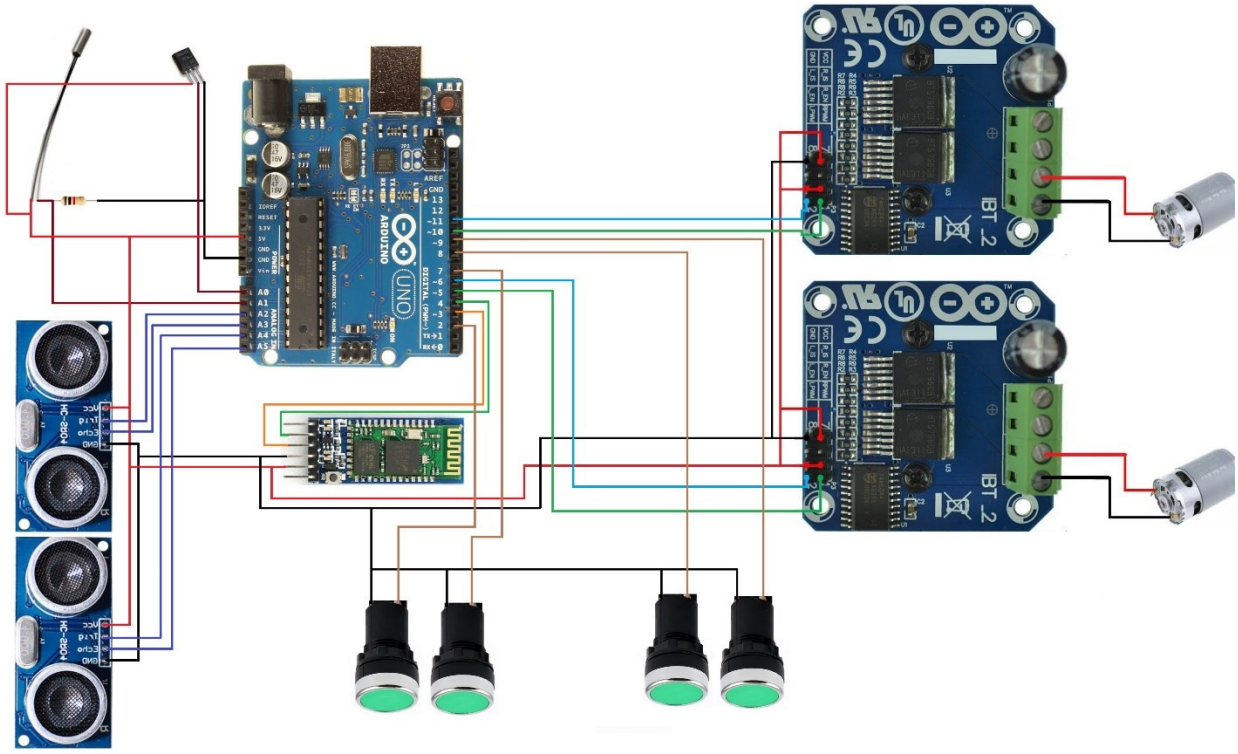

Figure 16: Arduino Connection Diagram.

Chapter 3
Software System

Chapter 3 :Software System

To achieve the desired objects, the software system was divided into two major sections. The first one was designed using Android Studio software. Which aims to provide a GUI control panel to enable the user to monitor the temperature of the environment and the patient, to control the angular position of each moving platform, to monitor the current state of the moving platforms and to provide alarming functions. The second one is responsible to control the hardware system and to provide the measurement related to temperature and angular position feedback and was implemented using C code and implemented using Arduino IDE.

3.1 Android application

The Android studio software was selected to perform this task as it is an open-source program that enables the user to add or edit specific libraries to complete the task. Also, this program was designed by GOOGLE Company the owner of the Android operating system. For this reason, the Android studio is considered as the most powerful tool to design Android applications.

The basic principle of the designed android application is to read and write signals based on Bluetooth sockets. Where the app gets the user options from the main screen and transmit it as a socket using Bluetooth protocol. The control signal consists mainly from two parts, the first one is the type of the control method which can be manual or automatic. And the other one is the reference signal. The reference signal is also divided into sub objects as follows. If the user selects the manual control, the signal type can be, Head up, Head down, Bottom up and Bottom down. Any one of these signals will move one of the moving platforms up or down for a

specific time. As the user repeat the selection, the app will send the same packet again until the user ends the manual control. On the other hand, the automatic control option consists of 6 different signals, three signals for each moving platform. The system defines three location for each moving platform as upper location, middle location and lower location. And the automatic control aims to move the moving platforms from the current angular position to the defined angular position. Full details about these signals are listed in Table 4.

Table 4: Control signal details

Control type	Signal type	Signal Value
Manual control	Head up	A
	Head down	B
	Bottom up	C
	Bottom up	D
Automatic control	Head upper location	E
	Head middle location	F
	Head lower location	G
	Head upper location	H
	Head middle location	I
	Head lower location	J

The previous signals were all generated by the Android app upon user selection. However, another type of signals is delivered to the application related to the temperature measurement. These signals are generated by the Arduino as an encoded message, which contains a symbol to notify the application about the begin of the packet, environmental temperature, a separator and a symbol to notify the application about the end of the packet. The general form of this packet is as follows,

*T1, T2#

Where,

- * is the symbol that notifies the begin of the packet.
- T1 is the ambient temperature.
- T2 is the temperature of the patient.
- # is the symbol that notifies the end of the packet.

This packet is decoded by the application when it is received. The first step of extracting data from this packet is to indicate the starting and ending symbols. If the application finds both of these symbols, it will decide that the data packet is received correctly and will continue executing to the next step. In the next step, the android application will create a string array of size two. Then using the split function, the data will be splitted into two values and stored inside the array. Finally, the values inside the array are parsed into integers to be shown in the main screen of the application. These steps can be summarized as follows,

1. Receive data packet.
2. Identify starting and ending symbols.
3. If they are exist, then the packet is received correctly.
4. Else, break.
5. Create a string array of size 2.
6. Split data using the separator ",".
7. Store splitted data into the array.
8. Convert data from string type to integer type using the function Integer.ParseInt().

When the temperature data is ready, the application will compare these values with the reference temperature values inserted by the user. If any temperature is larger than the reference value inserted by the user, the application will immediately create a visual and hearable alarm to notify the user about this issue.

The flowchart of control signal generation is shown in Fig.3.1 and the decoding process is shown in Fig.3.2. And the function of this application is ended when the transmits the control signal to the Arduino.

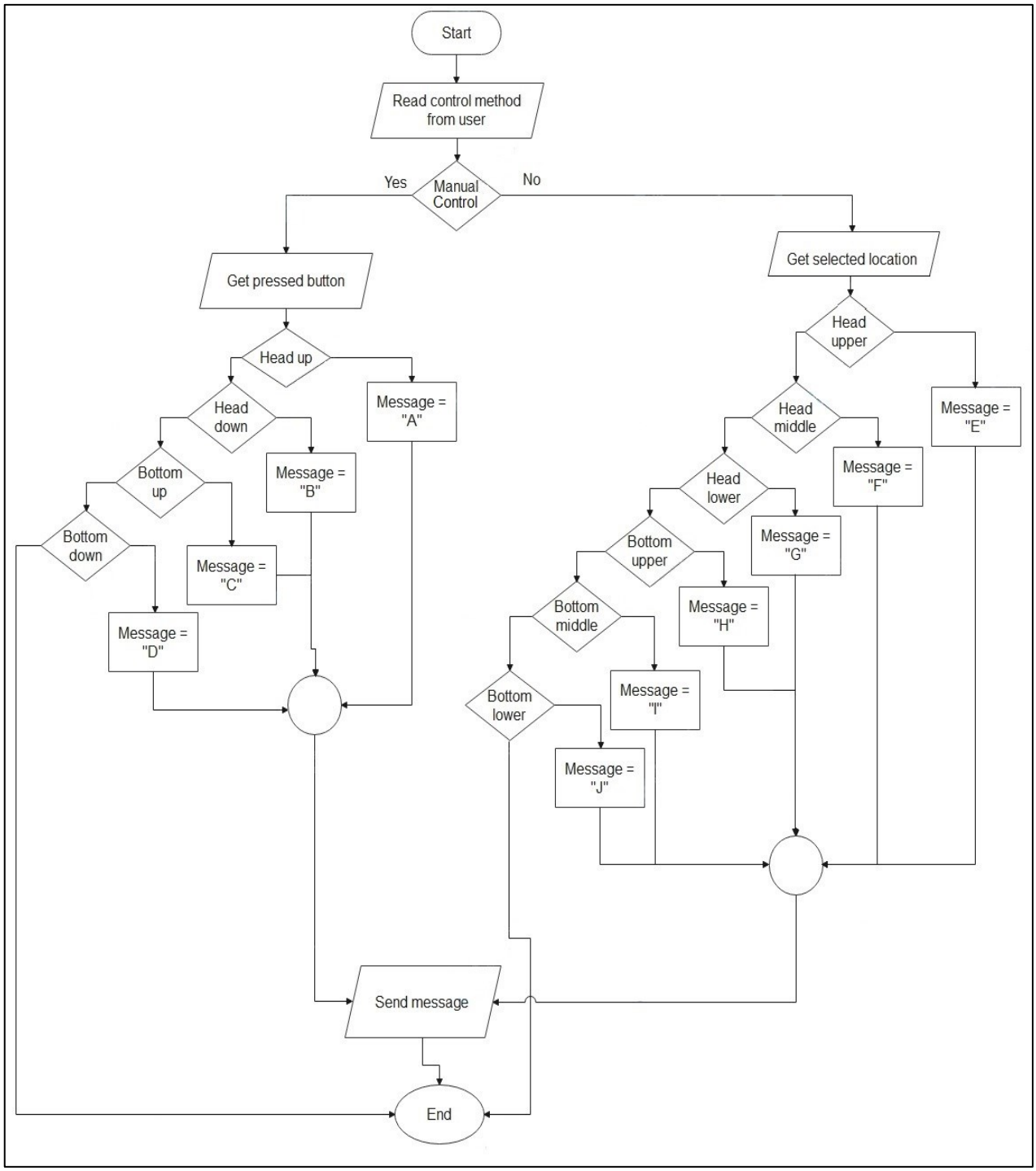

Figure 17: Android application Flowchart.

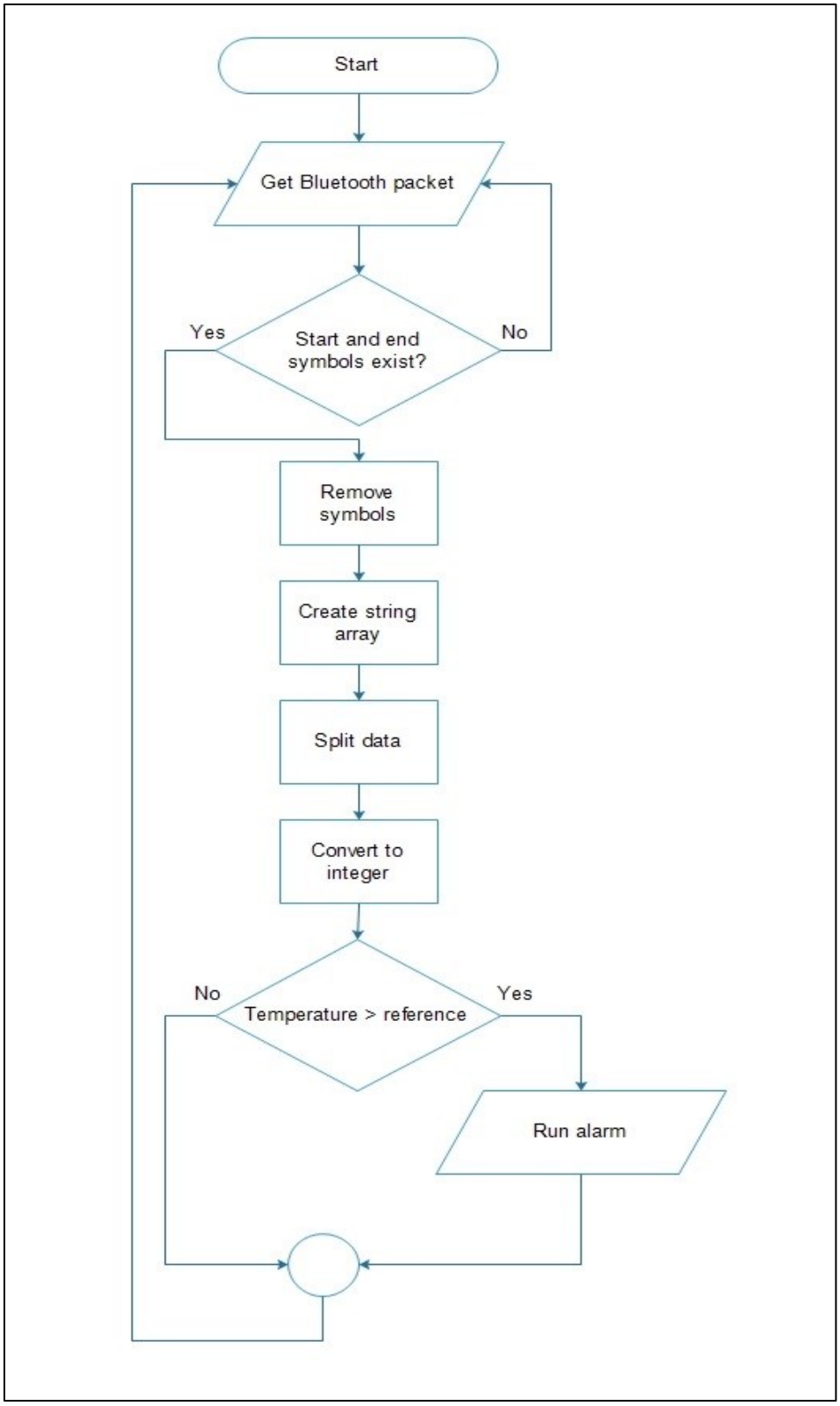

Figure 18: Receive and decode temperature signal.

To this end, the user can set the values of the reference alarm, monitor the ambient temperature and the temperature of the patient, monitor the state of the chair and control the state of the chair. These details are shown in the use case diagram in Fig.3.3.

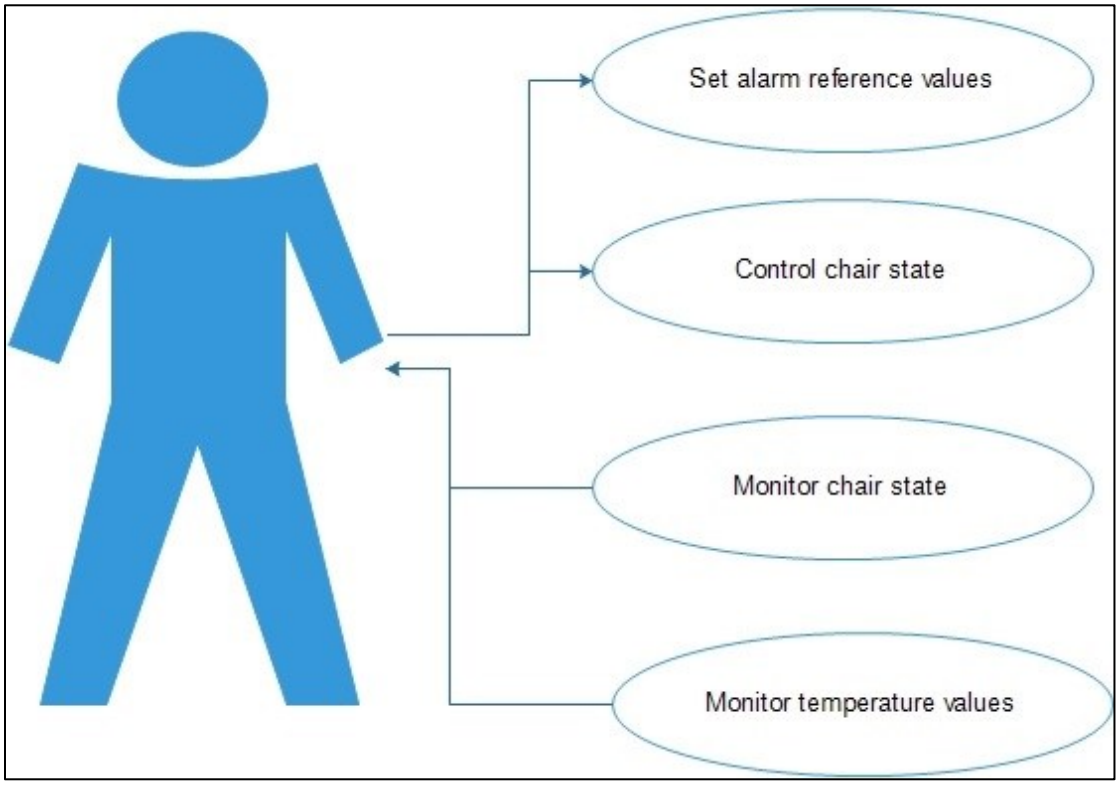

Figure 19: Use case diagram.

Finally the GUI interface of the android application is shown in Fig.3.4, where all components are listed below,

1	Low alarm value	11	Move the head to the bottom
2	High alarm value	12	Move he bottom to the top
3	Increase low alarm value	13	Move he bottom to the middle
4	Decrease low alarm value	14	Move he bottom to the bottom
5	Increase high alarm value	15	Manually move the head up
6	Decrease high alarm value	16	Manually move the head down

7	Actual room temperature	17	Manually move the bottom up
8	Actual patient temperature	18	Manually move the bottom down
9	Move the head to the top	19	Monitor chair’s state
10	Move the head to the middle		

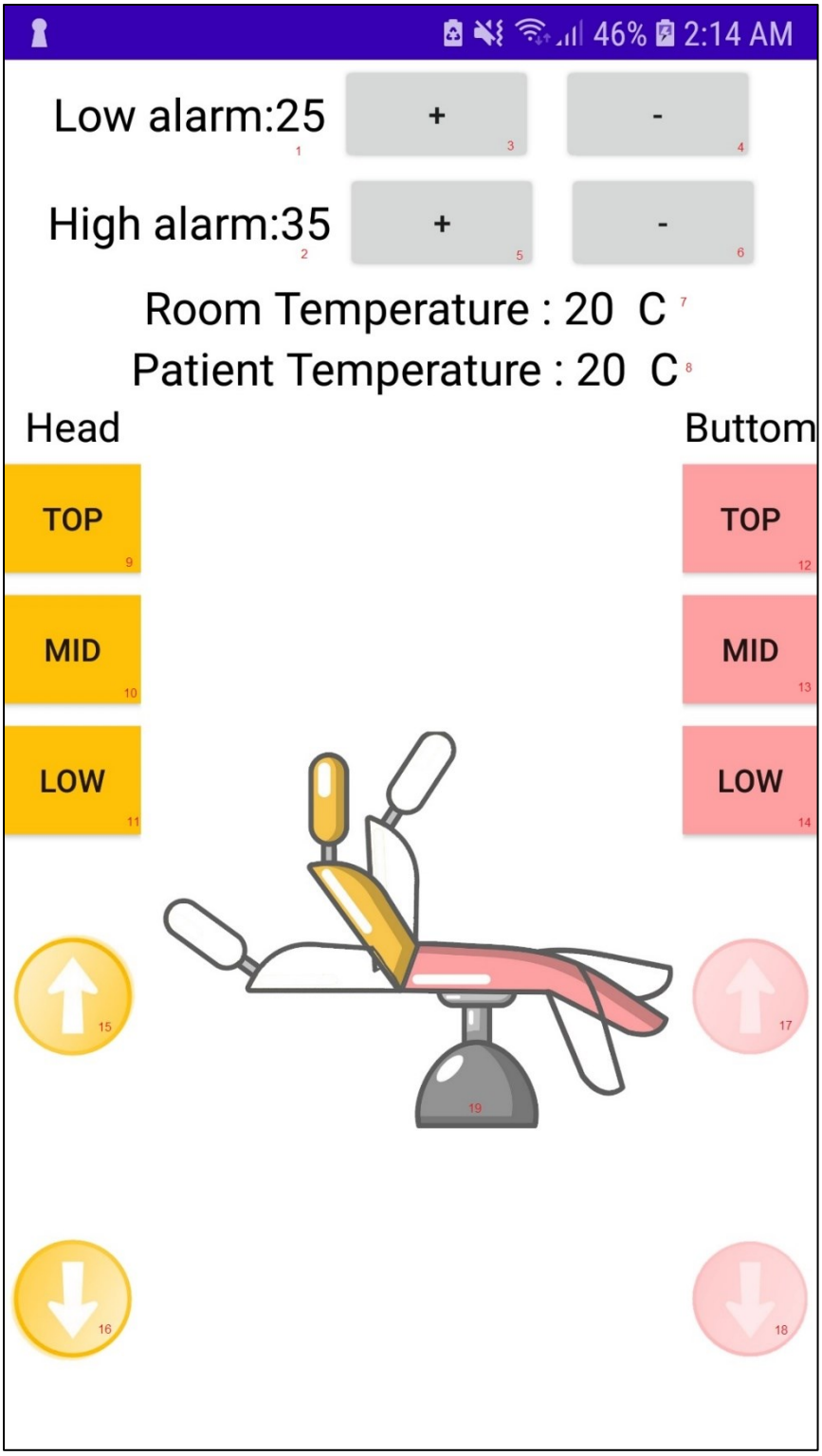

Figure 20 : GUI interface.

3.2. Arduino and control system

The Arduino IDE was used to implement all functions related to control algorithm, feedback measurements and temperature measurements.

3.2.1 Control algorithm

Control algorithm aims to move both moving platform to the desired angular position. The reference angular position can be determined by the user using the Android application as mentioned in the previous section. As both moving platforms have the same dimensions, mass, friction and actuators, the control law is the same for both of them. The Proportional Integral Derivative (PID) control algorithm was adopted to control their angular positions, where two standalone PID controllers were used to control the entire system. This converts the control problem into two Single Input Single Output (SISO) systems where each system will be controlled independently from the other, because the system is decupled. The implementation of the PID control system is shown in the block diagram in Figure 21.

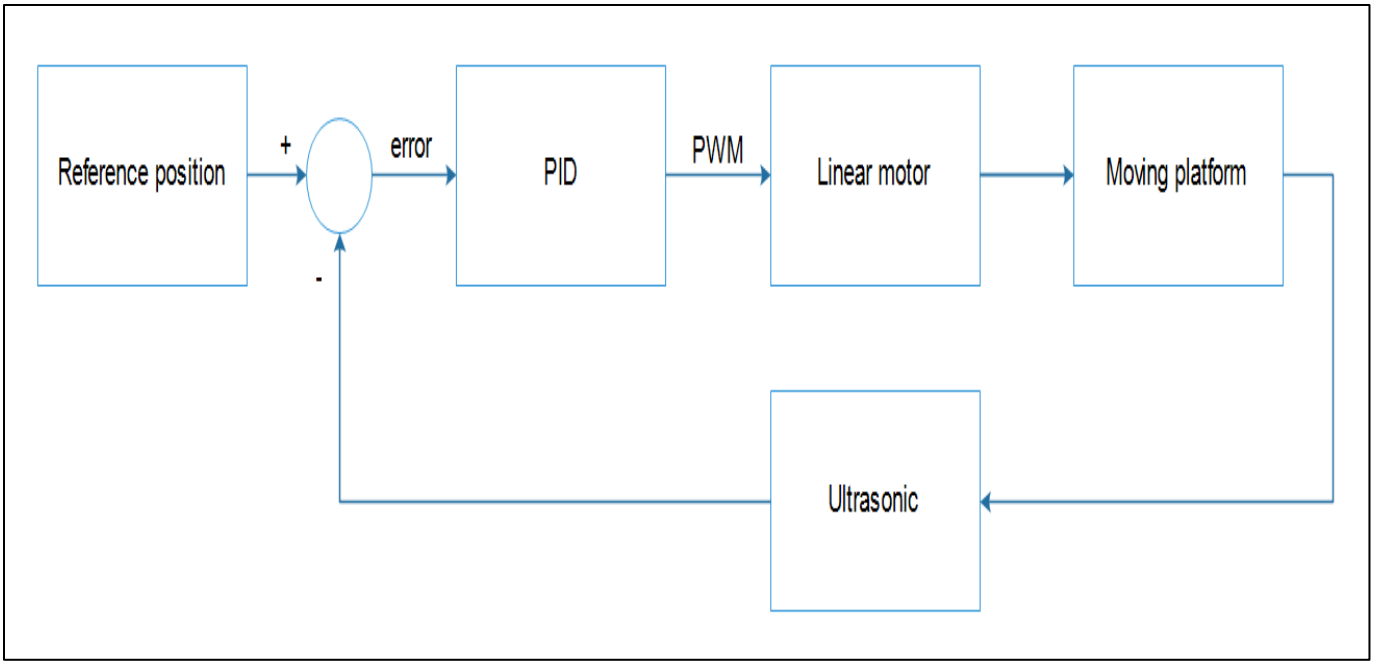

Figure 21: Control block diagram.

3.2.2 Hardware algorithm

As the Arduino receives the reference signals for both moving platforms, it measures the angular position of each moving platform using the attached ultrasonic sensor. Then it computes the control law as shown in Figure 22. The result of the control law computation is delivered to the linear actuator by the mean of PWM through the IBT2 driver.

The Arduino will continuously compare the current position of each moving platform with its reference signal, and will compute control law continuously as the control error is larger than zero. At the same time, it will measure the room and the patient temperature continuously and ill create an encoded packet containing these measurements and transmits them to the android application by the mean of Serial Bluetooth protocol. These steps are shown in the flowchart in Figure 23. The Algorithm contains two parallel branches. Both of them are executed continuously

during running interval. The fist branch (at the left) is responsible to read reference signal received from the mobile application, compute control law and apply it to the actuators. While the other branch (right one) is responsible to read the temperature sensors continuously and send it to the mobile application.

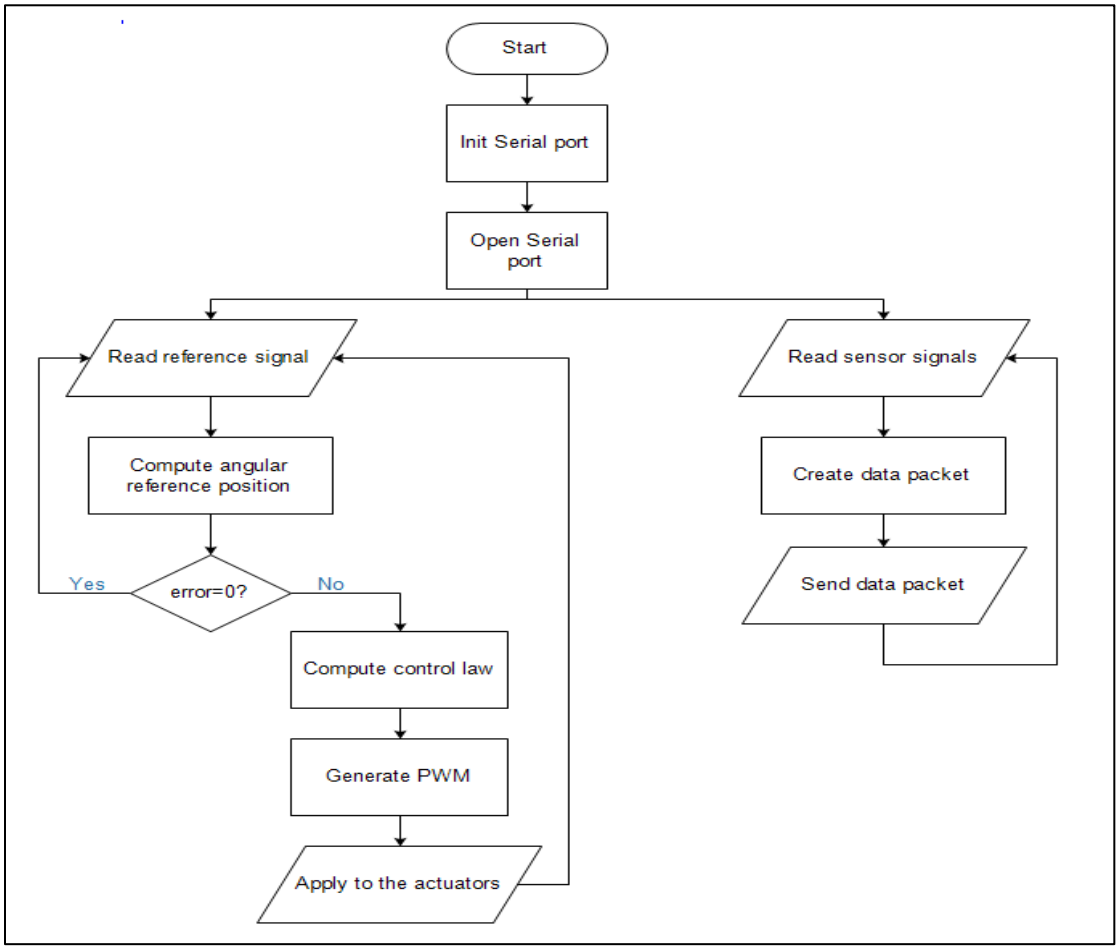

Figure 22: Hardware sequence flow chart.

Chapter 4
Conclusion and Future Work

Chapter 4: Conclusion and future work

4.1 Conclusion.

In this project, a hardware prototype of 2D controlled chair was designed and implemented as shown in Figure 23 and Figure 24. The prototype was manipulated and controlled using Arduino UNO microcontroller, and the reference signal was generated by the user using a GUI control and monitor panel which was designed for this project using Android studio software. The prototype provides multiple advantages such as,

1. Simple control panel.
2. Accurate angular position control.
3. Low cost.
4. Providing safety factors.
5. Efficient alarming function.
6. Remote control using mobile.
7. Flexibility and ability to improve and to add new functions upon user request.

Figure 23: Hardware prototype.

Figure 24: Transmission mechanism.

4.2. Future work

This project represent a solid base for future developments and researches. It is very flexible to be developed and improved by adding new functions and services to help the users and to save their efforts and money. The following recommendations can be suggested as a future work,

1. Add more sensors to measure the patient's vital signs.

   Heart rate measurement, ECG signal, blood pressure, SPO2 measurement and diabetes measurement.

2. Add new degree of freedoms to the system.

   Such that the chair can totally rotate around Z axis.

3. Perform level control functions.

   Where the user can move all the chair up or down.

4. Add mass sensor

   Mass sensor can provide critical measurement about the mass of the patient which can be used to improve the efficiency of the control algorithm.

5. Provide reports service.

   Which enable the user to print/ monitor the vital signs of the user along specific period.

References

[1] merriam-webster. (2021). *Hospital bed*. Available: https://www.merriam-webster.com/medical/hospital%20bed

[2] wikipedia. (2021). *Hospital bed*. Available: https://en.wikipedia.org/wiki/Hospital_bed

[3] K.-K. S. Y.-J. C. P.-L. L. M.-H. L. J.-J. S. C.-H. W. Y.-T. W. P.-C. Tung, "Total Design of an FPGA-Based Brain–Computer Interface Control Hospital Bed Nursing System," *IEEE Transactions on Industrial Electronics,* vol. 60, no. 7, pp. 2731 - 2739, 2013.

[4] K. s. Fund, "The hospital bed: on its way out?," *BMJ,* vol. 346, 2013.

[5] M. A. Ripon Kumar Chakrabortty, Md. Nuruzzaman, "Fuzzy and AHP approaches for designing a hospital bed: a case study in Bangladesh," *International Journal of Industrial and Systems Engineering,* vol. 17, no. 3, 2014.

[6] H. K. S. Abedian, H. Riazi, E. Bitaraf, "Cross Hospital Bed Management System," *IOS,* vol. 205, pp. 126-130.

[7] N. G. K. M. N. Mohammed, M. A. Abdelgnei, "A New Design of Multi-Functional Portable Patient Bed," *UTM,* vol. 59, no. 2, 2012.

[8] (2021). *Wight of human body parts*. Available: http://robslink.com/SAS/democd79/body_part_weights.htm

Appendix A

A.1 XML GUI control panel code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/lg1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Low alarm:"
            android:textColor="@color/black"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/top_range"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="25"
            android:textColor="@color/black"
            android:textSize="20sp" />

        <Button
            android:id="@+id/add1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="5dp"
            android:text="+" />

        <Button
            android:id="@+id/sub1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:text="-" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/lt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="High alarm:"
        android:textColor="@color/black"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/btn_range"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="35"
        android:textColor="@color/black"
        android:textSize="20sp" />

    <Button
        android:id="@+id/add2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:text="+" />

    <Button
        android:id="@+id/sub2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:text="-" />
```

```
</LinearLayout>

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:orientation="horizontal">

   <TextView
      android:id="@+id/l1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Room Temperature : "
      android:textColor="@color/black"
      android:textSize="20sp" />

   <TextView
      android:id="@+id/l2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="20"
      android:textColor="@color/black"
      android:textSize="20sp" />

   <TextView
      android:id="@+id/l3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="  C "
      android:textColor="@color/black"
      android:textSize="20sp" />

</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:orientation="horizontal">

   <TextView
      android:id="@+id/l11"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Patient Temperature : "
      android:textColor="@color/black"
      android:textSize="20sp" />
```

```
    <TextView
        android:id="@+id/l4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="20"
        android:textColor="@color/black"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/l33"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="  C "
        android:textColor="@color/black"
        android:textSize="20sp" />

</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Head"
            android:textColor="@color/black"
            android:textSize="18sp" />

        <Button
            android:id="@+id/top1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:layout_marginBottom="5dp"
            android:background="#FFC107"
```

```
        android:text="Top"
        app:backgroundTint="#FFC107" />

    <Button
        android:id="@+id/mid1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:background="#FFC107"
        android:text="Mid"
        app:backgroundTint="#FFC107" />

    <Button
        android:id="@+id/low1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:background="#FFC107"
        android:text="Low"
        app:backgroundTint="#FFC107" />

    <ImageView
        android:id="@+id/up1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/a1" />

    <ImageView
        android:id="@+id/down1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/a2" />
</LinearLayout>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="0.4">

    <ImageView
        android:id="@+id/body"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:src="@drawable/body_img" />

    <ImageView
        android:id="@+id/top_high"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/top_high_na" />

    <ImageView
        android:id="@+id/top_mid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/top_mid" />

    <ImageView
        android:id="@+id/top_low"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/top_low_na" />

    <ImageView
        android:id="@+id/btn_high"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/btn_high_na" />

    <ImageView
        android:id="@+id/btn_mid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/btn_mid" />

    <ImageView
        android:id="@+id/btn_low"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/btn_low_na" />

</FrameLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:gravity="center"
```

```
android:orientation="vertical">

<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Buttom"
    android:textColor="@color/black"
    android:textSize="18sp" />

<Button
    android:id="@+id/top2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"
    android:background="#FEA0A1"
    android:text="Top"
    app:backgroundTint="#FEA0A1" />

<Button
    android:id="@+id/mid2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"
    android:background="#FEA0A1"
    android:text="Mid"
    app:backgroundTint="#FEA0A1" />

<Button
    android:id="@+id/low2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"
    android:background="#FEA0A1"
    android:text="Low"
    app:backgroundTint="#FEA0A1" />

<ImageView
    android:id="@+id/up2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
```

```
            android:src="@drawable/a3" />

        <ImageView
            android:id="@+id/down2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:src="@drawable/a4" />
    </LinearLayout>
  </LinearLayout>

</LinearLayout>
```

A.2 Android Java code

```
package com.example.chaircontroller;
import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.lang.reflect.Method;
import java.util.UUID;

public class MainActivity extends Activity {
   final int RECIEVE_MESSAGE = 1;        // Status  for Handler
   private ConnectedThread mConnectedThread;
   private BluetoothAdapter btAdapter = null;
   private BluetoothSocket btSocket = null;
   private StringBuilder sb = new StringBuilder();
   private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
```

```
    private static String address = "00:19:09:03:6C:DE";
    TextView top_range,btn_range;
    Button add1,add2,sub1,sub2;
    Handler h;
ImageView top_high,top_mid,top_low,btn_high,btn_mid,btn_low,up1,up2,down1,down2;
Button top1,mid1,low1,top2,mid2,low2;
TextView l1,l2,l3,l4;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btAdapter=BluetoothAdapter.getDefaultAdapter();
        checkBtstate();
        h = new Handler() {
            public void handleMessage(android.os.Message msg) {
                switch (msg.what) {
                    case RECIEVE_MESSAGE:                                    // if receive massage
                        byte[] readBuf = (byte[]) msg.obj;
                        String strIncom = new String(readBuf, 0, msg.arg1);                 // create string
from bytes array
                        sb.append(strIncom);                                // append string
                        int endOfLineIndex = sb.indexOf("#");                    // determine the end-of-line
                        if (endOfLineIndex > 0 ) {                                           // if end-of-line,
                            String sbprint = sb.substring(0, endOfLineIndex);           // extract string
                            sb.delete(0, sb.length());                      // and clear
                            try{
                                System.out.println(sbprint);
l2.setText(sbprint.split(",")[0]);
l4.setText(sbprint.split(",")[1]);
try{
    int t = Integer.parseInt(l2.getText().toString().trim());
    int tl=Integer.parseInt(top_range.getText().toString().trim());
    int th=Integer.parseInt(btn_range.getText().toString().trim());
    if (t>th || t<tl){
        l1.setTextColor(Color.RED);
        l2.setTextColor(Color.RED);
        l3.setTextColor(Color.RED);
    }
    else {
        l1.setTextColor(Color.BLACK);
        l2.setTextColor(Color.BLACK);
        l3.setTextColor(Color.BLACK);
    }
}catch(Exception e){}

                        }catch (Exception e){}
```

```
                }
                break;
            }
        };
    };
    l1=findViewById(R.id.l1);
    l2=findViewById(R.id.l2);
    l3=findViewById(R.id.l3);
    l4=findViewById(R.id.l4);
    top_high=findViewById(R.id.top_high);
    top_mid=findViewById(R.id.top_mid);
    top_low=findViewById(R.id.top_low);

    btn_high=findViewById(R.id.btn_high);
    btn_mid=findViewById(R.id.btn_mid);
    btn_low=findViewById(R.id.btn_low);
    top1=findViewById(R.id.top1);
    top2=findViewById(R.id.top2);
    mid1=findViewById(R.id.mid1);
    mid2=findViewById(R.id.mid2);

    low1=findViewById(R.id.low1);
    low2=findViewById(R.id.low2);
    up1=findViewById(R.id.up1);
    up2=findViewById(R.id.up2);
    down1=findViewById(R.id.down1);
    down2=findViewById(R.id.down2);
top_range=findViewById(R.id.top_range);
btn_range=findViewById(R.id.btn_range);
add1=findViewById(R.id.add1);
add2=findViewById(R.id.add2);
sub1=findViewById(R.id.sub1);
sub2=findViewById(R.id.sub2);

add1.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    int th=Integer.parseInt(top_range.getText().toString().trim());
    th++;
    top_range.setText(th+"");
  }
});

    sub1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
```

```
        int th=Integer.parseInt(top_range.getText().toString().trim());
        th--;
        top_range.setText(th+"");
    }
});



add2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        int tl=Integer.parseInt(btn_range.getText().toString().trim());
        tl++;
        btn_range.setText(tl+"");
    }
});

sub2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        int tl=Integer.parseInt(btn_range.getText().toString().trim());
        tl--;
        btn_range.setText(tl+"");
    }
});




up1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("g");
    }
});
down1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("h");
    }
});
up2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("i");
    }
```

```
});
down2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("j");
    }
});
top1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("a");
        top_high.setImageResource(R.drawable.top_high);
        top_mid.setImageResource(R.drawable.top_mid_na);
        top_low.setImageResource(R.drawable.top_low_na);

    }
});

mid1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("b");
        top_high.setImageResource(R.drawable.top_high_na);
        top_mid.setImageResource(R.drawable.top_mid);
        top_low.setImageResource(R.drawable.top_low_na);
    }
});

low1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("c");
        top_high.setImageResource(R.drawable.top_high_na);
        top_mid.setImageResource(R.drawable.top_mid_na);
        top_low.setImageResource(R.drawable.top_low);
    }
});


top2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mConnectedThread.write("d");
        btn_high.setImageResource(R.drawable.btn_high);
        btn_mid.setImageResource(R.drawable.btn_mid_na);
        btn_low.setImageResource(R.drawable.btn_low_na);
```

```
        }
    });

    mid2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mConnectedThread.write("e");
            btn_high.setImageResource(R.drawable.btn_high_na);
            btn_mid.setImageResource(R.drawable.btn_mid);
            btn_low.setImageResource(R.drawable.btn_low_na);
        }
    });

    low2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mConnectedThread.write("f");
            btn_high.setImageResource(R.drawable.btn_high_na);
            btn_mid.setImageResource(R.drawable.btn_mid_na);
            btn_low.setImageResource(R.drawable.btn_low);
        }
    });


  }
  public void onResume() {
    super.onResume();
    if (btSocket==null) {
        try {
            BluetoothDevice device = btAdapter.getRemoteDevice(address);
            try {
                btSocket = createBluetoothSocket(device);
            } catch (Exception e) {
                Toast.makeText(getBaseContext(), "البرنامج تشغيل اعادة يرجى , الاتصال اثناء خطأ حدث",
Toast.LENGTH_LONG).show();
            }

            btAdapter.cancelDiscovery();
            try {
                btSocket.connect();
            } catch (Exception e) {
                Toast.makeText(getBaseContext(), "البرنامج تشغيل اعادة يرجى , الاتصال اثناء خطأ حدث",
Toast.LENGTH_LONG).show();
                try {

                    btSocket.close();
```

```
            } catch (Exception e2) {
            }
        }

        mConnectedThread = new ConnectedThread(btSocket);
        mConnectedThread.start();
      } catch (Exception e) {
        Toast.makeText(getBaseContext(), "البرنامج تشغيل اعادة يرجى , الاتصال اثناء خطأ حدث",
Toast.LENGTH_LONG).show();
      }
    }

  }
  public void onDestroy() {
    super.onDestroy();
    try {
      btSocket.close();
    } catch (Exception e2) {
    }
  }
  public void checkBtstate(){
    if (btAdapter !=null && !btAdapter.isEnabled()){
      Intent btintent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(btintent,1);
    }

  }
  private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws Exception {
    if(Build.VERSION.SDK_INT >= 10){
      try {
        final Method m =
device.getClass().getMethod("createInsecureRfcommSocketToServiceRecord", new Class[] {
UUID.class });
        return (BluetoothSocket) m.invoke(device, MY_UUID);
      } catch (Exception e) {
      }
    }
    return  device.createRfcommSocketToServiceRecord(MY_UUID);
  }
  private class ConnectedThread extends Thread {
    private final InputStream mmInStream;
    private final DataOutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
      InputStream tmpIn = null;
      DataOutputStream tmpOut = null;
```

```
        try {
            tmpIn = socket.getInputStream();
            tmpOut = new DataOutputStream(socket.getOutputStream())  ;
        } catch (Exception e) { }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }

    public void run() {
        byte[] buffer = new byte[256];  // buffer store for the stream
        int bytes; // bytes returned from read()

        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);      // Get number of bytes and message in
"buffer"
                h.obtainMessage(RECIEVE_MESSAGE, bytes, -1, buffer).sendToTarget();      //
Send to message queue Handler
            } catch (Exception e) {
                break;
            }
        }
    }

    public void write(String message) {
        byte[] msgBuffer = message.getBytes();
        try {
            // mmOutStream.write(data1);
            mmOutStream.write(msgBuffer);
            // mmOutStream.write(data2);
        } catch (Exception e) {
        }
    }
  }
}
```

A.3 Arduino C code

```
#include <HCSR04.h>

#include <SoftwareSerial.h>
```

```
SoftwareSerial bt(3, 4);
const int f1=5;
const int r1=6;
const int f2=10;
const int r2=11;

const int e1 = A4;
const int t1 = A5;
const int e2 = A2;
const int t2 = A3;

const int u1 = 2;
const int d1 = 7;
const int u2 = 8;
const int d2 = 9;


HCSR04 hc2(t1,e1);
HCSR04 hc1(t2,e2);
unsigned long previousMillis = 0;
const long interval = 1000;
int ref1 = 5;
int ref2 = 5;
int auto_control=1;
void setup() {
```

```
Serial.begin(9600);
bt.begin(9600);
pinMode(f1,OUTPUT);
pinMode(f2,OUTPUT);
pinMode(r1,OUTPUT);
pinMode(r2,OUTPUT);
pinMode (t1,OUTPUT);
pinMode (e1,INPUT);
pinMode(u1,INPUT_PULLUP);
pinMode(u2,INPUT_PULLUP);
pinMode(d1,INPUT_PULLUP);
pinMode(d2,INPUT_PULLUP);
}

void loop() {
int a=digitalRead(u1);
int b=digitalRead(d1);
int c=not(digitalRead(u2));
int d=not(digitalRead(d2));

if(a==1){auto_control=0; read_serial_data('g');}
else if(b==1){auto_control=0; read_serial_data('h');}
else if(c==1){auto_control=0; read_serial_data('i');}
else if(d==1){auto_control=0; read_serial_data('j');}
```

```
 float vout=analogRead(A0);
 int tempc=(vout*500)/1023;

float vout1=analogRead(A1);
 int temp1=(vout1*151)/1023;

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis;

  String l=(String)tempc+","+(String)temp1+"#";
  bt.println(l);
  Serial.println(l);
  }


 int d1 = hc1.dist();
 delay(10);
 int d2 = hc2.dist();
 delay(10);
// String line =(String)d1+","+(String)d2;
if(auto_control==1){
if(ref1>d1)down1(200);
```

```
else if(ref1<d1)up1(200);
else stop1();

if(ref2>d2)down2(127);
else if(ref2<d2)up2(127);
else stop2();
}

if(Serial.available()){read_serial_data(Serial.read());}
if(bt.available()){read_serial_data(bt.read());}



}

void read_serial_data(int x){


      if (x=='a'){ref1=4;auto_control=1;}
  else if (x=='b'){ref1=8;auto_control=1;}
  else if (x=='c'){ref1=11;auto_control=1;}

  else if (x=='d'){ref2=2;auto_control=1;}
  else if (x=='e'){ref2=7;auto_control=1;}
  else if (x=='f'){ref2=12;auto_control=1;}
```

```
  else if (x=='g'){up1(255);delay(300);stop1();auto_control=0;}
  else if (x=='h'){down1(255);delay(300);stop1();auto_control=0;}

  else if (x=='i'){up2(255);delay(300);stop2();auto_control=0;}
  else if (x=='j'){down2(255);delay(300);stop2();auto_control=0;}

  }

long get_dest(int t,int e){
  long duration1, cm1;
  pinMode(t, OUTPUT);
  digitalWrite(t, LOW);
  delayMicroseconds(2);
  digitalWrite(t, HIGH);
  delayMicroseconds(10);
  digitalWrite(t, LOW);
  pinMode(e, INPUT);
  duration1 = pulseIn(e, HIGH);
  cm1 = duration1/ 29 / 2;
 return cm1 ;

  }

void down1(int s){
```

```
digitalWrite(r1,LOW);
analogWrite(f1,s);
}
void up2(int s){
digitalWrite(f2,LOW);
analogWrite(r2,s);
}
void up1(int s){
digitalWrite(f1,LOW);
analogWrite(r1,s);
}
void down2(int s){
digitalWrite(r2,LOW);
analogWrite(f2,s);
}

void stop1(){
digitalWrite(r1,LOW);
digitalWrite(f1,LOW);
}
void stop2(){
digitalWrite(r2,LOW);
digitalWrite(f2,LOW);
}
```