TECHNOLOGICAL UNIVERSITY (KALAY)

DEPARTMENT OF ELECTRONIC ENGINEERING

DESIGN AND IMPLEMENTATION OF ROBOTIC ARM (ANDROID SOFTWARE IMPLEMENTATION)

BY

MAUNG AUNG YE HTET

B.E. MINI-THESIS

OCTOBER, 2018

KALAY

TECHNOLOGICAL UNIVERSITY (KALAY)

DEPARTMENT OF ELECTRONIC ENGINEERING

DESIGN AND IMPLEMENTATION OF ROBOTIC ARM (ANDROID SOFTWARE IMPLEMENTATION)

BY

MAUNG AUNG YE HTET

A MINI-THESIS

SUBMITTED TO THE DEPARTMENT OF

ELECTRONIC ENGINEERING IN

PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF BACHELOR OF ENGINEERING

(ELECTRONICS)

OCTOBER, 2018

KALAY

TECHNOLOGICAL UNIVERSITY (KALAY)

DEPARTMENT OF ELECTRONIC ENGINEERING

We certify that we have examined, and recommend to the University Steering Committee for Post Graduate Studies for acceptance of the BE. Mini-thesis entitled: **"DESIGN AND IMPLEMENTATION OF ROBOTIC ARM"** submitted by **Maung Aung Ye Htet, Roll No. B.E EcE-47 (October, 2018)** to the Department of Electronic Engineering in partial fulfillment of the requirements for the degree of B.E. (Electronics).

4. Daw Bawi Nun Thiam

   B.E. (Electronics), TU (Kalay);

   Assistant Lecturer

   Department of Electronic Engineering ..……..…………………

   Technology University (Kalay) (Member)

ACKNOWLEDGEMENT

First and foremost, the author is grateful to Prof. Dr. Thant Zaw Oo, Principal of Technological University (Kalay), for giving the permission to submit the thesis for the Bachelor Degree at Technological University (Kalay).

The author would like to express grateful thanks to Daw Naing, Lecturer and Head, Department of Electronic Engineering, Technological University (Kalay), for permission to carry out this thesis and helps, suggestions and supervisions for this thesis.

The author especially thanks his supervisor, Daw Cho Thet Nwe, Assistant Lecturer, Department of Electronic Engineering from Technological University (Kalay), for her teaching, advices, supervisions, and supports during the preparation of this thesis.

The author deeply appreciates U Than Htun Aung, Assistant Lecturer, Department of Electronic Engineering from Technological University (Kalay), for his excellent guidance and valuable suggestions.

The author wishes to express his gratitude to, Daw Bawi Nun Thiam, Department of Electronic Engineering, Technological University (Kalay), for her helpful comments and suggestions.

Last but not the least, the author is deeply grateful to his parents and brothers, and to all teachers who shared their knowledge and to all of his friends who heartily helped him.

ABSTRACT

A robotic arm is a mechatronic arm which is usually programmable the functions similar to human arm on a mounted stand. Robotic Arm which can be used in industries to do repetitive task such as moving the things from a conveyor to another place, in research for dangerous chemical test. This robotic arm is a human based control system using the controller application on android phone by driving servos from Arduino. Arduino Uno board is used to communicate with android phone and to drive the servo motors. This robotic arm is five degrees of freedom robotic arm which can move 180 degree and in 3 directions and can do almost all the works of gripping. First of all, this robotic arm is able to move a specific point of its workspace through x, y and z axis vales automatically if the points are given. Secondly, Bluetooth module support in robotic arm as a wireless interface between the robotic arm and the android phone so that the robotic arm can be controlled ten meter away from it. Finally, the commands can change immediately as the controller application stores the algorithms that run the robotic arm servos and Arduino receives the command by a unit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1. Introduction to Robotic Arm

Indeed the earth is vastly revolving all over and together with this are the different changes, which range from having a simple life to be more complex. On the other hand, electronics are one of the technologies being created today which are widely used all over the world. Likely the term mechatronics, a combination of electronics and mechanics are used in industries. Robotic arm is one mechatronic application that can operate autonomously to handle various industrial processes that need more physical effort. A Robotic arm is a robotic manipulator, usually programmable, with similar functions to a human arm; the arm may be part of a more complex robot. The robotic arm will be connected to the mobile phone through Bluetooth and can be controlled by using an android controller application. The robotic arm is also sometimes referred to as anthropomorphic as it is very similar to that of a human hand. Humans today do all the task involved in the manufacturing industry themselves. However, a robotic arm can be used for various task such as welding, drilling and spraying and many more. The main part of the design is ATmega328 microcontroller which coordinates and controls the product action.

Researcher have been done and implemented in order to have knowledge about mechanics and software during the operations carried out by the robotic arm which is designed to fulfill the tasks determined in accordance with predetermined commands. It was determined what function the robotic arm would be and what movements it could make. Robotic arm made of stepper motor; it can carry the desired material, mix it up and perform the commands previously determined by a user. If this project is also a designated task; the robotic arm takes a piece of material and brings it to the desired position and then records its movements and lets it do the same action until it is stopped. The servo motor is preferred in order to be able to precisely and must be at high torque. The robotic arm is composed of seven servo motors and can move in five axis directions with motors.

In the thesis, Arduino IDE (Integrated Development Environment) software written in Java language is programmed and servo motor driven is provided. Thus it is possible to perform the desired operations by means of the elements located on the Arduino without any circuit construction other than the circuit where the servo motor inputs are located. For mechanical part, the robotic arm frames are drawn with the CorelDraw software and the dimensions of the robotic arm are specified. A 5V power supply is also preferred for the robot to work.

Figure 1.1. Robotic arm

1.2. Aim and Objectives

The general objective of this thesis is to develop an algorithm to make a robotic arm capable of accomplishing pick-and-place operations. Such operations involve moving an objective from an initial to a final given position using android controller application. After reviewing the literature of previous thesis done about manipulator motion, specific objectives were established.

The specific objectives include:

- To define the entire workspace of the manipulator in order to design paths with reachable configurations
- To select components for working long time operations of robotic arm.
- To design the algorithms for controller and driver to perfectly accomplish robotic arm’s operations.
- To evaluate the effectiveness of the proposed method by comparing the theory and experimental results.

1.3. Scope of Thesis

The motion of this robotic arm is limited to a circular path that means the robotic arm can move in an autonomy of degree. For example, the volume and the area of sphere is $4/3\pi r^2$ and $4\pi r^2$ respectively, so the movement and motion of the robotic arm was restricted according to the formula. Any duty which can be done by the movement along the circulator shape of sphere can be performed by this robotic arm if the written code is matched. Arduino Uno is used as driver, generally the reset circuitry and power setup will be ready especially the circuitry of programming and Bluetooth communication. The body and some special part of robotic arms can be done with fireboard, servo motors, bearings, screws and jumper wire. For the wireless control device, android phone is used with Robotic Arm Controller application.

1.4. Implementation Program

The Implementation procedure are following:

- To find the design fundamental for robotic arm
- To select the best material used to build the robotic arm
- To design the circuit for driver and the program for controller
- To test the implemented robotic arm

1.5. System Overview

In this thesis, Robotic arm is designed with scrap material and servos. The arm has been built with fiber and the individual parts have been locked to servo motors. Arduino Uno is programmed to control servo motors. Servo are serving as joints of robotic arm. This setup also looks as a robotic crane or it can be converted into a crane by some easy tweaks.

These servos can be moved by rotating the arms to pick some object, with some practice the object can be easily picked and moved from one place to another. Low torque servos can be used here but more powerful servos can be used to pick heavy objects. Robotic arm which can be controlled remotely by an Android phone. It also can receive commands via Bluetooth and work accordingly. Develop an Android application which allows the user to send commands via Bluetooth. Commands received by Bluetooth modem connected to Arduino microcontroller. Microcontroller drives servo motors which allows the movement of robotic arm in all directions and gripper to handling physical objects.

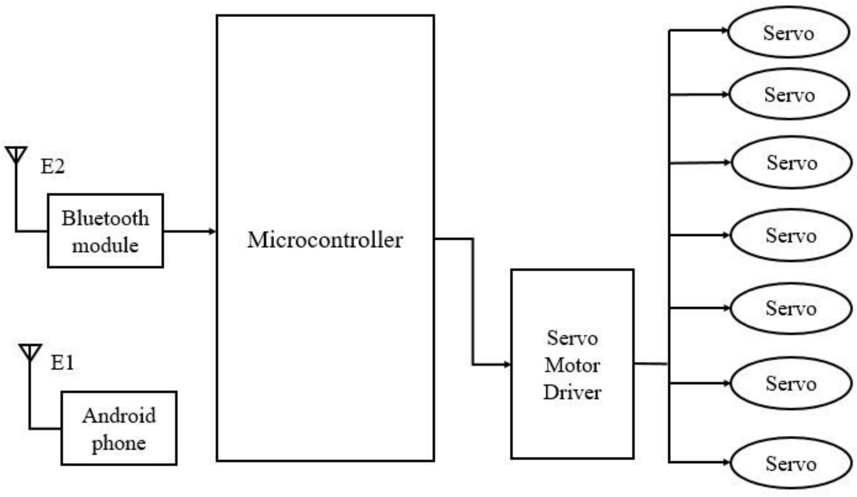


Figure 1.2. Overall System Block Diagram

1.6. Outline of the Thesis

During the course of this thesis, certain issues had to be addressed which was critical to the overall success of the thesis. These issues can be categorized into four separate sections. They are android software implementation, Arduino software implementation, circuit implementation and hardware implementation.

1.7. Layout of the Thesis

In this thesis organized as follow, chapter two describes the overview of the robotic arm. In chapter three, the design of the robotic arm followed by mechanical parts. Android software implementation of the robotic arm will be mention in chapter four. In chapter five, the test and result of the robotic arm will be described. In the end, a short conclusion is drawn in chapter six.

CHAPTER 2
LITERATURE REVIEW

2.1. Robotic Arm

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm. With the rise in manufacturing industrial activities, a robotic arm is invented to help various industries to perform a task or work instead of using manpower. Industrial robotic arms are designed to do exactly the same thing, in a controlled environment, over and over again. For example, a robot might twist the caps onto peanut butter jars coming down an assembly line. To teach a robotic arm how to do is job, the programmer guides the arm through the motions using a handled controller. The robotic arm stores the exact sequence of movements in is memory, and does it again and again every time a new unit comes down the assembly line. The light material lifting task can be de done by robotic arm efficiently and time saving because it is restricted by fatigue or health ricks which man might experience. It handle tasks that are difficult, dangerous or boring to human beings.

The robotic arm which is usually made up of seven metal segments, joined by six joints. The computer controls the robot by rotating individual step motors connected to each joint. The links of such a manipulator are connected by joints allowing either rotational motion or translational displacement. The links of the manipulator can be considered to form a kinematic chain. The terminus of the kinematic chain of the manipulator is called the end effector and it is analogous to the human hand. The end effector or robotic arm can be designed to perform any desired tasks such as welding, drilling and spraying etc.

The robotic arm can be autonomous or controlled manually and can be used to perform a variety of tasks with great accuracy. Robotic arm can do a lot of work more efficiently than human beings because they are so precise. The robotic arm can be fixed or mobile and can be designed for industrial or home application. The wireless mobile robotic arm also have been developing in previous year.

2.2. History of Robotic Arm

The foundation of surgical robotics is in the development of the robotic arm. This is a thorough review of the literature on the nature and development of this device with emphasis on surgical applications. The published literature and classified robotic arms have been reviewed by their application: show, industrial application, medical application, etc. There is a definite trend in the manufacture of robotic arms toward more dexterous devices, more degrees-of-freedom, and capabilities beyond the human arm. Da Vinci designed the first sophisticated robotic arm in 1495 with four degrees-of-freedom and an analog on-board controller supplying power and programmability. Von Kemplen's chess-playing automaton left arm was quite sophisticated. Unimate introduced the first industrial robotic arm in 1961, it has subsequently evolved into the PUMA arm. In 1963 the Rancho arm was designed: Minsky's Tentacle arm appeared in 1968, Scheinma's Stanford arm in 1969, and MIT's Sliver arm in 1974. Aird became the first cyborg human with a robotic arm in 1993.

In 2000 Miguel Nicolalis redefined possible man machine capacity in his work on cerebral implantation in owl-monkeys directly interfacing with robotic arms both locally and at a distance. The robotic arm is the end effector of robotic systems and the hallmark feature of the Vinci Surgical System making its entrance into surgical application. But, despite the potential advantages of this computer controlled master-slave system, robotic arm have definite limitations. Ongoing work in robotics has many potential solutions to the drawbacks of current robotics surgical systems. Although surgical robotics is in its infancy, the rapid proliferation of surgical systems attests to the fact that this technology is here to stay and that urologists should brace themselves for the next wave of technology that will yet again change the way they work. Many in practice are rather started by the rapid insurgence of this sophistication technology into the armamentarium of clinical practice.

The approach in this historical review will be a bit different from that in other published accounts of robotic technology that is increasingly proliferating. The robotic arm will be the slope topic of this investigation and will be the dissected rather like the human arm. Some context will be added for literary interest but the focus will be on a sequential timeline of development and how we arrived at a piano-wire based, seven degrees-of-freedom surgical system for urology that is now sweeping across the United States.

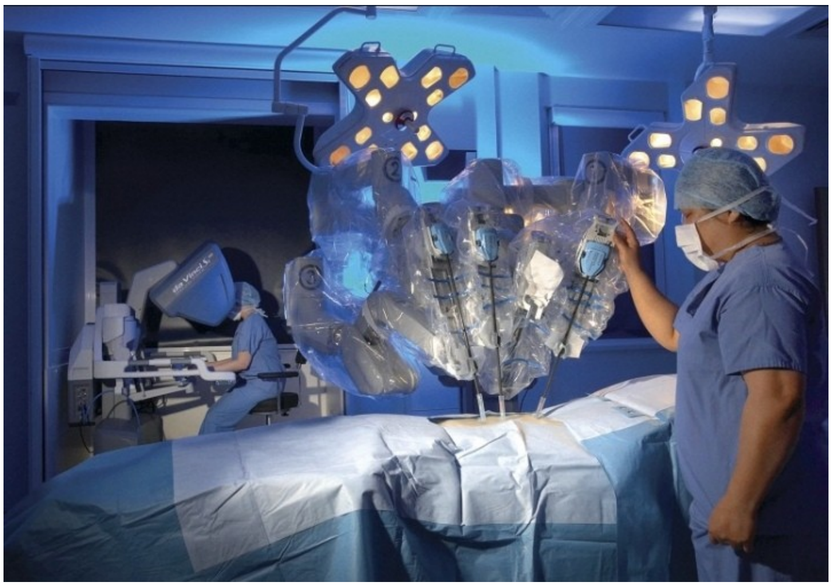


Figure 2.1. Da Vinci surgical Robotic Arm

2.3. Various type of Robotic Arm

There are many different types of robotic arms, but most can be characterized into one of five major categories by their mechanical structure.

- Cartesian Robotic arm
- Cylindrical Robotic arm
- Parallel Robotic arm
- SCARA Robotic arm
- Articulated Robotic arm

2.3.1. Cartesian Robotic Arm

A robotic arm which has linear actuators cooperating with linear motors kinked to a linear axis is known as a linear robotic arm. This link can be fixed of flexible connections between the actuators and the robot. The linear motor is directly attached to the linear axis. Robotic arm which use two motors in controlling a linear axis defined gantry robotic arm. Each motor has a limited distance orthogonal to the linear axis. Ball screws follow the same principles which either use linear motor or rotary motors. This kind of robots usually achieve tasks. The manipulator of the linear robots is connected in an overhead way that allows the robotic arm to move along the horizontal plane easily, where each of these movements are perpendicular to each other and are basically defined as “x,y” for horizontal axis and sometimes “z” in case of having a vertical axis.

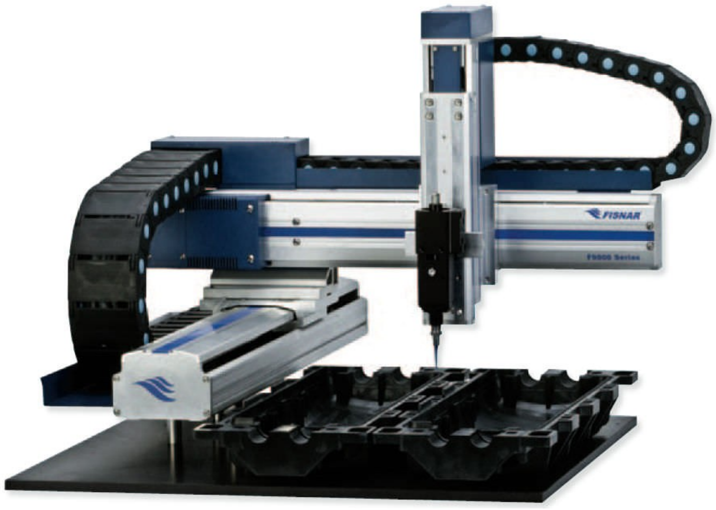


Figure 2.2. Linear Robotic Arm

2.3.2. Cylindrical Robotic Arm

Cylindrical robotic arm have two prismatic joints: one rotary joint for positioning task and the end-effector of the robot forms a cylindrical workspace. The main idea of the cylindrical robotic arm is to mount a horizontal arm which moves in forward and backward directions. The horizontal arm is linked to a carriage which goes up and down and is connected to the rotary base. Schematic cylindrical robot and its symbol figure. Since both of the units move on the base, the workspace is annular space of the cylinder. .

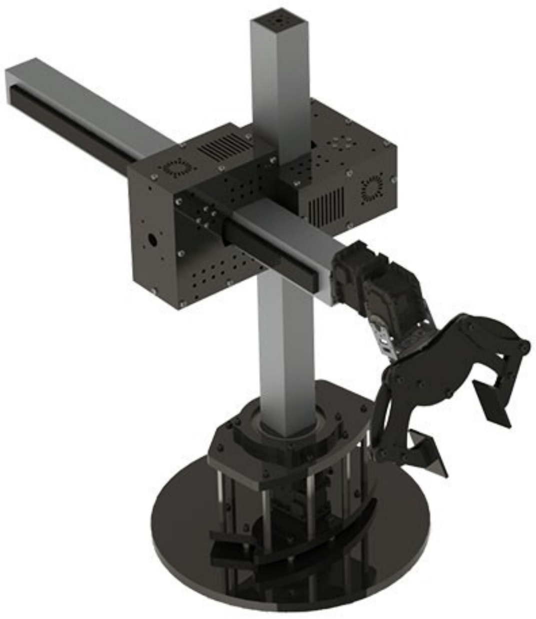


Figure 2.3. Cylindrical Robotic Arm

2.3.3. Parallel Robotic Arm

A parallel robot has an end-effector with DOF which is connected to a fixed base. The connection is done by at least two independent kinematic chains which provide the movements of the robot. A generalized parallel manipulator has a base. There exists different definitions and types of a parallel robotic arm yet the most common properties. There should not exist any mobility of manipulator when the motors are locked. There are different parallel robotic arm configurations but two kinematic design have become popular. First design is parallel robotic arm, which has tripod with three axis connection the end-effector to the movable platform and the base, and has a wrist with 2 or 3 DOF.

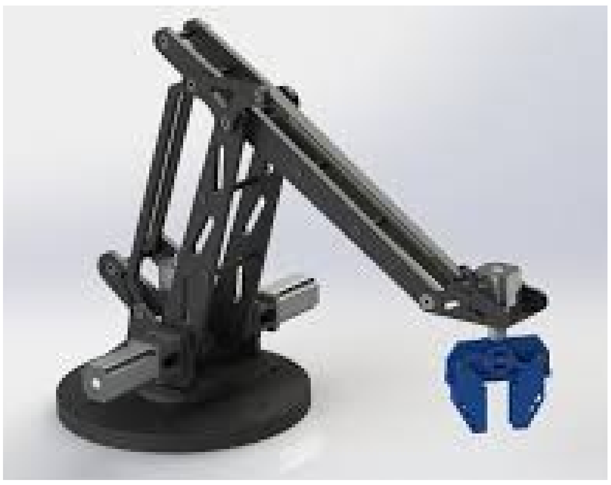


Figure 2.4. Parallel Robotic Arm

2.3.4. SCARA Robotic Arm

Selective Compliance Assembly Robotic Arm (SCARA) was first designed and invented in early 1960s in Japan. SCARA robot is perfect for the applications which require high speed and repetitive point to point movements. This is why SCARA is widely used in assembly operation. Special end-effector movement makes SCARA ideal for the tasks which require uniform motion and accelerations in a circular form. SCARA consists of two parallel rotary joints and a prismatic joint. The rotary joints can move along the horizontal plane and the prismatic joint moves along the vertical plane. One of the special characteristic of SCARA is that the robotics smooth while operating an x and y axis but very strong versus the z axis. SCARA arm is able to pick up a part vertically from a horizontally placed table and move along the assembly task by lowering the arm and placing the part at its proper location.

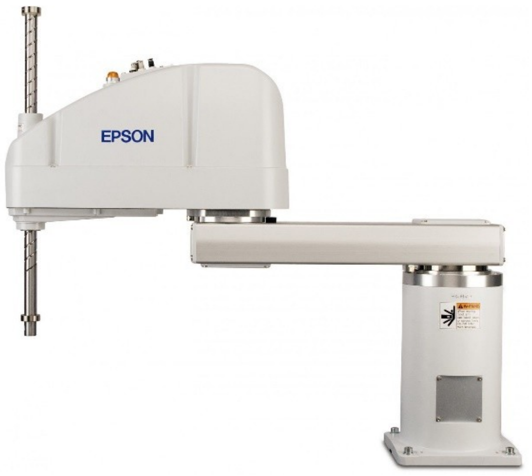


Figure 2.5. SCARA Robotic Arm

2.3.5. Articulated Robotic Arm

Articulated robot have three fixed axis connected to two revolute base. All joints of an articulated arm are revolute and most likely represent the human arm. The moving right objects are called links, revolute joints are called sliding joints. Each joint defines the relative motion of the other two object it links which determines the subset of the whole configuration space. Each configuration subset is a different position for each links. These subsets are simple to measure by considering a distance or an angle with each joint. A robotic arm can be said to be a typical example for articulated robot. An important matter which should be considered is that the dimension of the configuration space increases with the number of joints however the operation speed is limited due to the different payloads at the manipulator and nonlinear environment.

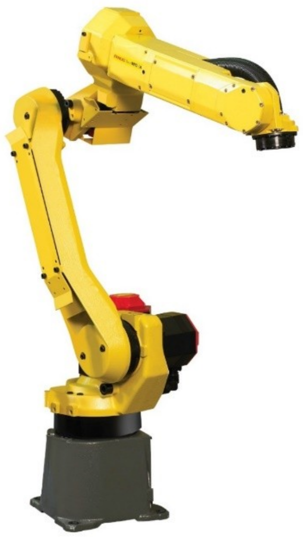


Figure 2.6. Articulated Robotic Arm

2.4. Summary

The introduction of the robotic arm review is described in this chapter. The design of the robotic arm will be discussed in the next chapter.

CHAPTER 3
SYSTEM DESIGN

3.1. Robotic Arm Design

The design part is divided into two parts; the mechanical part design and the mechanical part installation. In the design of the mechanical part, the Corel Draw drawings of the parts to be used in robotic arm construction were made through the help program. In the installation of the mechanical part, the naming of the servo motors used in the robotic arm and the tasks during the operation of the robotic arm are explained. The construction of the several steps. These steps are; Determination of the mechanical materials required for the production of the thesis,

- Determination of microcontroller and software to be used in the thesis
- Search and selection of servo motors that will run the robotic arm in a proper way
- Proper selection of mechanical parts
- Implementation of robotic arm assembly
- Testing the system to see if it works properly with the microcontroller and controller application
- Possible faults have been given in the form of restructuring the system by passing through the eye.

These steps have been completed and the Robotic arm design.

3.1.1. Five degrees of freedom

Serial and parallel manipulator system are generally designed to position end-effector with five-degrees of freedom, consisting of three in translation and two in orientation. This provides a direct relationship between actuator positions and the configuration of the manipulator.

Robotic arms are desired by five degrees of freedom. This number typically refers to the number of single-axis rotational points in the arm, where higher number indicates an increased flexibility in positioning a tool. The grabbing mechanism is not considered as such. The shoulder is bowing and rotating, the elbow and the wrist are only bowing and the hand itself is rotating.

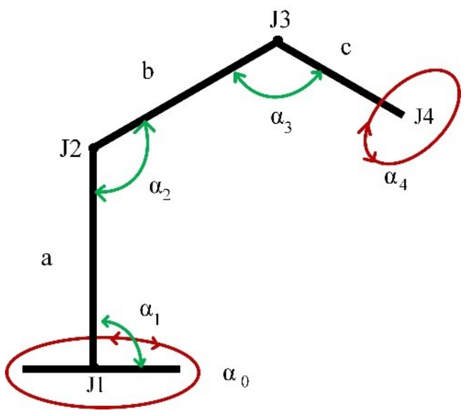


Figure 3.1. Five degree of freedom

3.2. Hardware components

This section will include components selection, their specifications, software developments, consideration and the detail of about them.

3.2.1. Component selection

- Arduino Uno
- Servo Motor
- Bluetooth module
- Gripper

3.2.1.1. Microcontroller

The microcontroller has played an importance role which is acted as computer with most of necessary support chips on board because each of the computer contains CPU, RAM and input/output. In the other side, microcontroller is a system of self-contained with peripherals, memory and a processor that can be used as an embedded system. Most of the embedded in other consumer products or machinery are using programmable microcontroller such as peripherals, automobiles, phones and household appliances for computer systems. Since microcontroller contain embedded inside it so there is a new name was added to the microcontroller which is embedded controller that means microcontroller can control the functionality, action, feature and movement of the thesis and product. Every mission, duty or task was dedicated by microcontroller and it is able to run the specific program, since the feature of the microcontroller were always be there. The feature or function of a microcontroller is to send a signal to specific device or component by taking an input from the controlled device.

3.2.1.2. Arduino Uno Microcontroller

Although microcontroller type PIC is usually used in programming and software field. Arduino has become very popular in the world in recent times. It is based on Arduino's past wiring and processing projects. Processing is written for non-programming users. Arduino wiring is produced on the basis of the programming language. The common feature of both is that it provides an environment where even the basic knowledge of electronics and programming can easily design. Arduino is now becoming more and more common nowadays. Even unmanned aerial vehicles made with Arduino, which is used almost every field, are visible.

The causes of the spread of Arduino at such a rapid rate are;

- ➢ It can be used on all platforms due to the simplicity of the development environment with driver usage.
- ➢ With the help of the advanced library, even complex operations can be easily solved.
- ➢ Program written in Arduino can run fast because they are not run on any other platform.
- ➢ There is a lot of hardware support that is compatible with Arduino and can work together.
- ➢ Communication with the environment is easy because it is open source.
- ➢ If there are any problem due to a large number of Arduino users, the solution can be easily reached.

The Arduino Uno is a small, full and breadboard friendly Arduino card that houses a microcontroller or ATmega328 microcontroller. It has almost same functions as the Arduino Duemilanove. Arduino is designed and used by Uno Gravitech. The Arduino Uno offers a variety of possibilities for communication with a computer, another Arduino, or other microcontrollers. The ATmega328 supports UART TIL serial communication, accessible via the RX and TX pins. An FTDI on the card channels the FT232RL serial communications via USB and the FTDI drivers and the writing on the computer appear as a virtual com port. The RX and TX LEDs on the card flash on the FTDI chip while the USB den serial cable and the USB is transmitting data. The Software Serial library allows serial communication over any of the digital pins of the Arduino Uno. The ATmega328 microcontroller also supports 12C (TWI) and SPI communications.

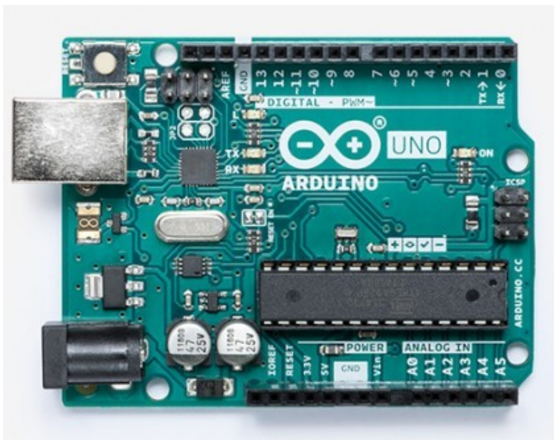


Figure 3.2. Arduino Uno Microcontroller

3.2.1.3. Servo Motor

A servo is a small device that has an output shift. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio controlled cars, puppets, and course, robots.

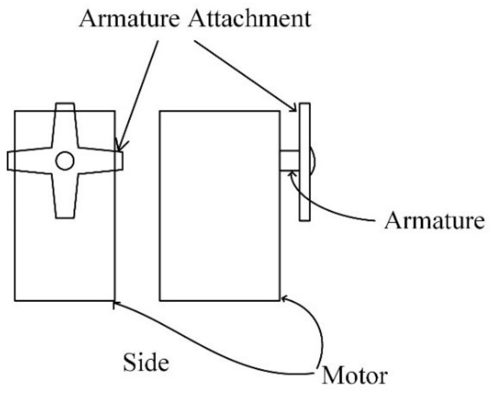


Figure 3.3. Servo notations

Servos are extremely useful in robotics. The motors are small, as shown in the picture below, have built in control circuitry, and are extremely powerful for their size. A standard servo such as the Futaba S-148 has 42oz/inches of torque, which is pretty strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, doesn't consume much energy. The guts of a servo motor are shown in the picture below.

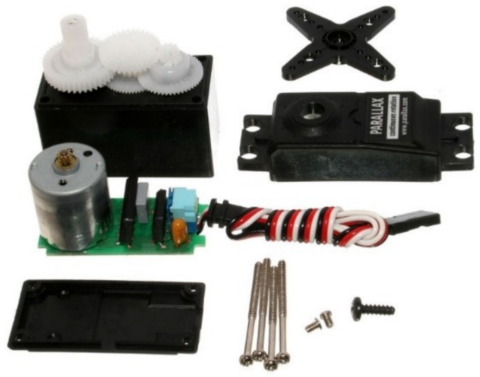


Figure 3.4. Disassembled Servo

The servo motor has some control circuits and a potentiometer that is connected to the output shaft. In the picture below, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shutoff. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, it's somewhere in the 210 degree range, but it varies by manufacture. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any father due to a mechanical stop built on to the main output gear.

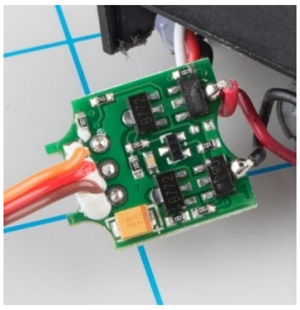


Figure 3.5. Servo circuit

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional.

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulsed Width Modulation. The servo expects to see a pulse every 20 milliseconds. The length of the pulse will determine how far the motor turns. A 1.5 milliseconds pulse, for example, will make the motor turn to the 90 degree position. If the pulse is shorter than 1.5ms, then the motor will turn the shaft to closer the shift to closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees. So we generate the desired pulse can be generated with the help of microcontroller. The servo uses three wires: white carriers the control signal, red carriers power and black is ground.

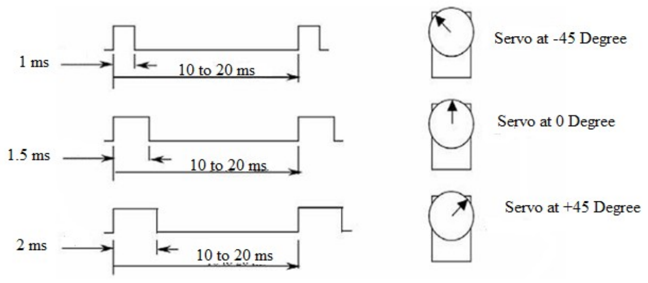


Figure 3.6. Servo pulses

3.2.1.4. MG995 Servo Motor

This is the most famous servo made by Tower Pro. MG995 is a digital metal gear high torque servo for airplane, helicopter, RC-car from 10 to 6-th Scale Truggy and Monster and many RC model.

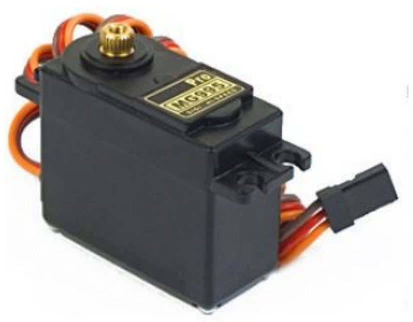


Figure 3.7. MG995 Servomotor

Characteristics	Specification
Weight	55g
Dimension	40.7×19.7×42.9mm
Stall torque	9.4kg/cm (4.8v); 11kg/cm (6v)
Operating speed	0.2sec/60degree
Operating voltage	4.8~6.6v
Gear type	Metal gear
Temperature range	0-55degree
Dead bandwidth	1µs

Table 3.1. Specification of MG995 Servo Motor

3.2.1.5. SG90 Servo Motor

Micro servo motor SG90 is a tiny and lightweight server motor with high output power. Servo can rotate approximately 180 degree (90 in each direction) and works just like the standard kinds be smaller. To control theses servos, any servo code, hardware or library can be used to control these servos.

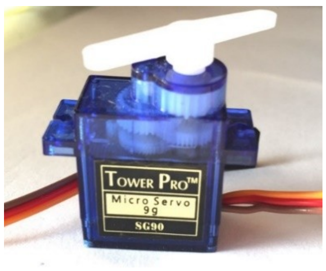


Figure 3.8. SG90 Servomotor

Table 3.2. Specification of SG90 Servo motor

Characteristic	Specification

Weight	9g
Dimension	22.2×11.8×31mm
Stall torque	2.8kg/cm
Speed	0.1s/60degree
Gear Type	plastic
Rotation	0-180 degree
Voltage	4.8V

3.2.1.6. MG996R Servo Motor

This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwidth and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

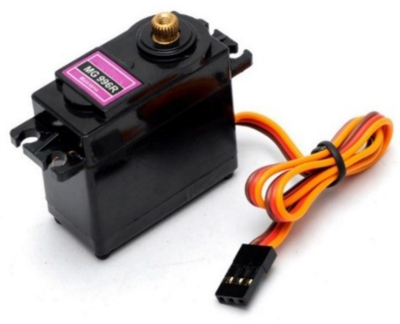


Figure 3.9. MG996R Servo Motor

Table 3.3. Specifications of MG996R Servo Motor

Characteristic	Specification
Weight	55g
Dimension	40.7 x 19.7 x 42.9 mm approx.
Stall torque	9.4 kg-cm (4.8 V), 11 kg-cm (6 V)
Operating speed	0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
Operating voltage	4.8 V a 7.2 V
Running current	500 mA

Stall current	2.5 A (6V)
Dead bandwidth	5 µs
Temperature range	0-55 degree

3.2.1.7. Bluetooth Module HC-05

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR 3Mpbs modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Blue core 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The role of the module can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.

Hardware Feature:

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- 3.3 to 5V I/O
- PIO (Programmable Input/Output) control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software Feature:

- Slave default Baud rate:9600, Data bits:8,Data bit:8
- Stop bit 1, Parity: No parity
- Auto-connect to the last device on power as default
- Permit pairing device to connect as default.
- Auto–pairing PINCODE ‘1234’ as default

Pin Description:

The HC-05 Bluetooth Module has 6 pins. They are as follows:

- ENABLE: When enable is pulled LOW, the module is disabled which means the module will not turn and fail to communicate. When enable is left opened or connected to 3.3V, the module is enabled i.e. the module remains on and communication also takes place.
- VCC: Supply Voltage 3.3V to 5V.
- GND: Ground pin.
- TXD and RXD: These two pins acts an UART interface for communication.
- STATE: It acts as a status indicator. When the module is not connected to / paired with any other Bluetooth device, signal goes Low. At this low state, the led flashes continuously which denotes that the modules is a bit not paired with other device. When this module is connected to/paired with any other Bluetooth device, the signal goes high. At this high state, the led blinks with a constant delay say for example 2s delay which indicates that the module is paired.

This is used to switch the module into AT command mode. To enable AT command mode, press the button switch for a second. With the help of AT commands, the user can change the parameters of this module but only when the module is not paired with any other BT device. If the module is connected to any other Bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

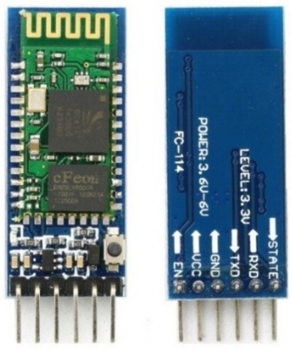


Figure 3.10. Bluetooth module HC-05

3.2.1.8. Gripper

The end effector is probably one of the most important and most complex parts of the system. The end effector varies mainly according to the application and the task that the robotic arm accomplishes for; it can be pneumatic, electric or hydraulic. Since the robotic arm is based on an electric system, electric basis offend effector may be chosen. Besides, the main application of the system is handling, according, the recommended type of the end effector is a gripper. A gripper is a device which enables the holding of an object to be manipulated. The easier way to describe a gripper is to think of the human hand.

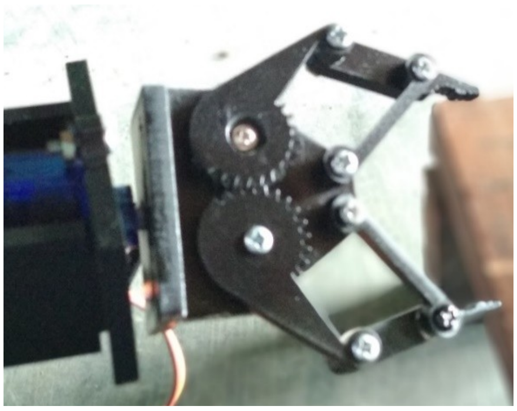


Figure 3.11. Gripper

3.3 Summary

This chapter discuss about the design of robotic arm and the next chapter will be described about the Arduino software implementation of the system.

CHAPTER 4
ANDROID SOFTWARE IMPLEMENTATION

4.1. Introduction

In this chapter, android system, java language and android software implementation of the system will be explained in some details. Android is mobile operating system developed by Google, based on a modified version of the Linux kernel. Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. Android Studio is the official integrated development environment (IDE) for Google's Android operating system. XML stands for Extensible Markup Language. Robotic Arm Controller application is the android software used to control the robotic arm via Bluetooth.

4.2. Android Platform

Android is mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics. Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013.

4.2.1. Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layer. They are

- Linux Kernel
- Android Libraries
- Android Runtime
- Application Framework
- Applications

4.2.2. Linux Kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

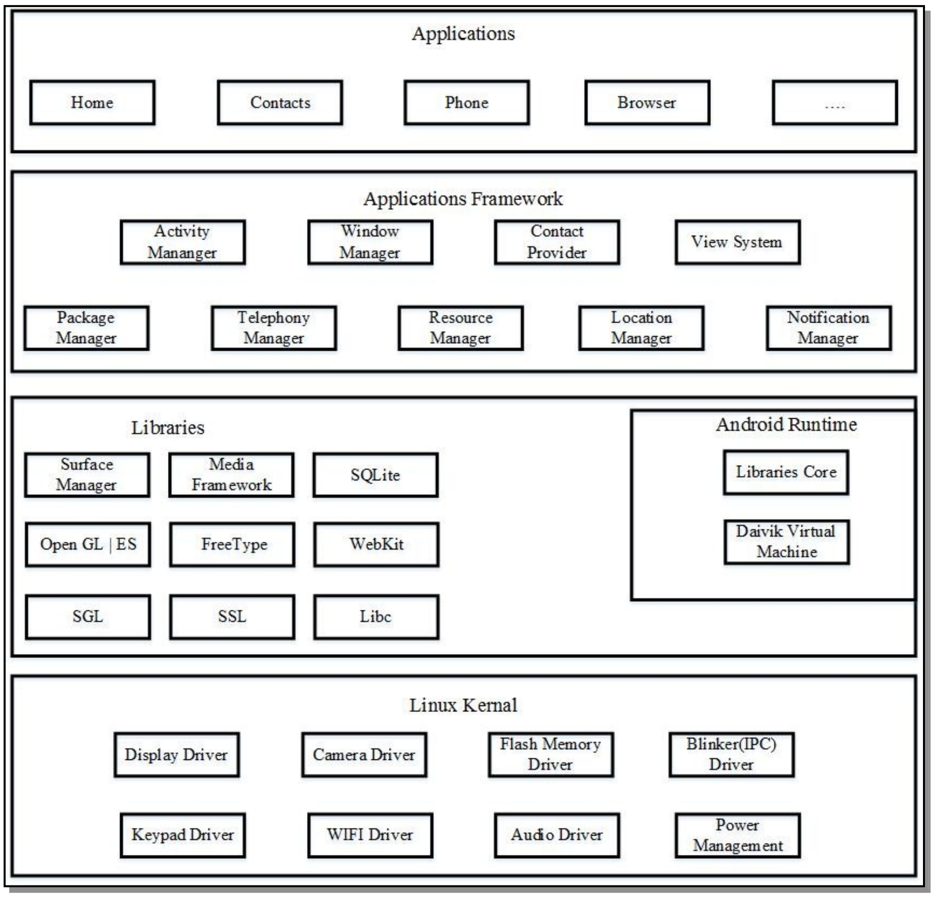


Figure 4.1. Android Architecture

4.2.3. Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows –

- android.app: Provides access to the application model and is the cornerstone of all Android applications.

- android.content: Facilitates content access, publishing and messaging between applications and application components.
- android.content: Facilitates content access, publishing and messaging between applications and application components.
- android.opengl: A Java interface to the OpenGL ES 3D graphics rendering API.
- android.os: Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- android.text: Used to render and manipulate text on a device display.
- android.view: The fundamental building blocks of application user interfaces.
- android.widget: A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

4.2.4. Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

4.2.5. Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. Application frameworks became popular with the rise of graphical user interfaces (GUIs). The Android framework includes the following key services –

- Activity Manager: Controls all aspects of the application lifecycle and activity stack.
- Content Providers: Allows applications to publish and share data with other applications.
- Resource Manager: Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- Notifications Manager: Allows applications to display alerts and notifications to the user.
- View System: An extensible set of views used to create application user interfaces.

4.2.6. Applications

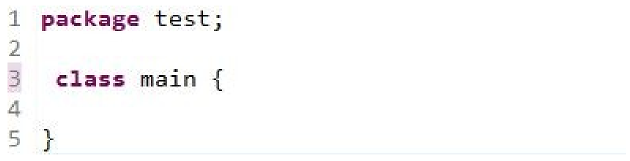
All the Android application at the top layer. Written applications are installed only on this layer. Examples of such applications are Contacts Books, Browser, and Games etc. Robotic Arm Controller application will be described on section 4.5.

4.3. Java Language

Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The target of Java is to write a program once and then run this program on multiple operating systems. The first publicly available version of Java (Java 1.0) was released in 1995. Sun Microsystems was acquired by the Oracle Corporation in 2010. Oracle has now the statesmanship for Java. In 2006 Sun started to make Java available under the GNU General Public License (GPL). Oracle continues this project called OpenJDK. The base of java programming structure are following.

4.3.1. Class

A class is a template that describes the data and behavior associated with an instance of that class. A class is defined by the class keyword and must start with a capital letter. The body of a class is surrounded by {}.



```
package test;

 class main {

}
```

Figure 4.2. Class keyword use in Java program

The data associated with a class is stored in variables; the behavior associated to a class or object is implemented with methods. A class is contained in a text file with the same name as the class plus the .java extension. It is also possible to define inner classes, these are classes defined within another class, and in this case you do not need a separate file for this class.

4.3.2. Object

An object is an instance of a class. The object is the real element which has data and can perform actions. Each object is created based on the class definition. The class can be seen as the blueprint of an object, i.e., it describes how an object is created.

4.3.3. Package

Java groups classes into functional packages. Packages are typically used to group classes into logical units. For example, all graphical views of an application might be placed in the same package called “tukaly.aungye.robotarm.views”.

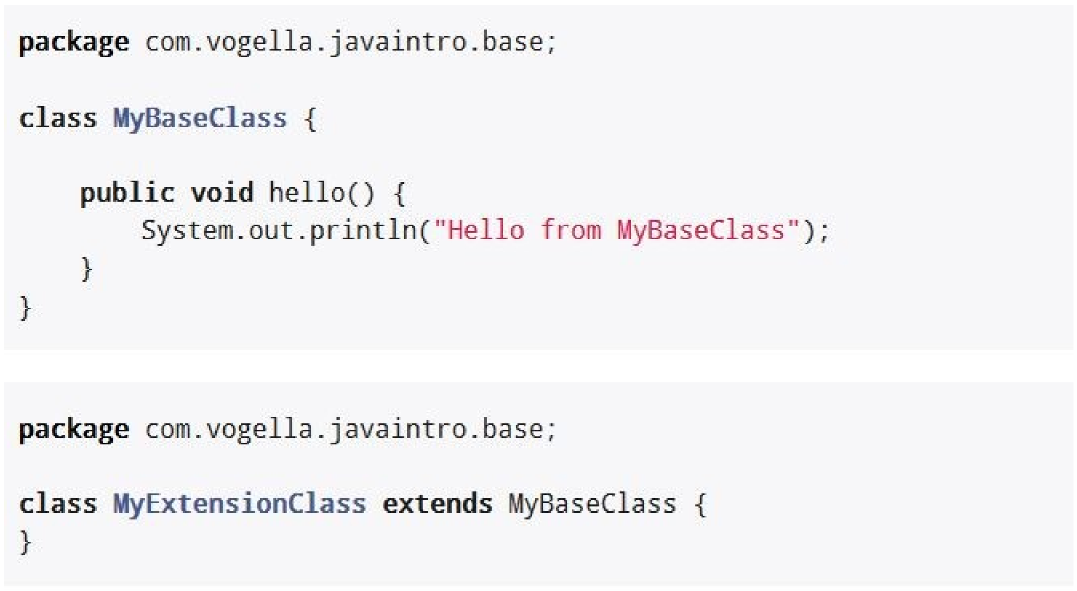
It is common practice to use the reverse domain name of the company as top level package. For example, the company might own the domain, vogella.com and in this example the Java packages of this company starts with “tukalay.aungye”.

Other main reason for the usage of packages is to avoid name collisions of classes. A name collision occurs if two programmers give the same fully qualified name to a class. The fully qualified name of a class in Java consists of the package name followed by a dot (.) and the class name.

Without packages, a programmer may create a Java class called Test. Another programmer may create a class with the same name. With the usage of packages you can tell the system which class to call. For example, if the first programmer puts the Test class into package report and the second programmer puts his class into package “xmlreader” you can distinguish between these classes by using the fully qualified name, e.g, xmlreader.Test or report.Test.

4.3.4. Inheritance

Class can be derived from another class. In this case this class is called a subclass. Another common phrase is that a class extends another class. The class from which the subclass is derived is called a superclass. Inheritance allows a class to inherit the behavior and data definitions of another class. The following codes demonstrates how a class can extend another class. In Java a class can only extend a maximum of one class.



```java
package com.vogella.javaintro.base;

class MyBaseClass {

    public void hello() {
        System.out.println("Hello from MyBaseClass");
    }
}
```

```java
package com.vogella.javaintro.base;

class MyExtensionClass extends MyBaseClass {
}
```

Figure 4.3. MyExtensionClass inherit from MyBaseClass

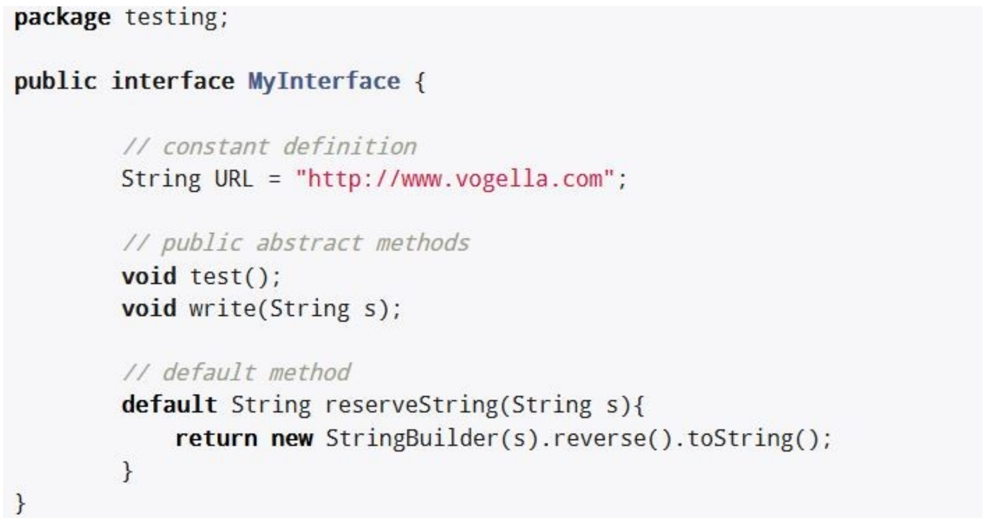
4.3.5. Exception Handling

In Java an exception is an event to indicate an error during the runtime of an application. So this disrupts the usual flow of the application's instructions. In general exceptions are thrown up in the call hierarchy until they get catched.

4.3.6. Interfaces

An interfaces is a type similar to a class and is defined via the interface keyword. Interfaces are used to define common behavior of implementing classes. If two classes implement the same interface, other code which work on the interface level, can use objects of both classes.

Like a class an interface defines methods. Classes can implement one or several interfaces. A class which implements an interface must provide an implementation for all abstract methods defined in the interface.



```
package testing;

public interface MyInterface {

        // constant definition
        String URL = "http://www.vogella.com";

        // public abstract methods
        void test();
        void write(String s);

        // default method
        default String reserveString(String s){
            return new StringBuilder(s).reverse().toString();
        }
}
```

Figure 4.4. Example implementation of an interface.

4.4. XML Design

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn't make our layout heavy.

4.4.1. Basics of User Interface

The whole concept of Android User Interface is defined using the hierarchy of View and ViewGroup objects. A ViewGroup is an invisible container that organizes child views. These child views are other widgets which are used to make the different parts of UI. One ViewGroup can have another ViewGroup as an child element as shown in the figure given below'

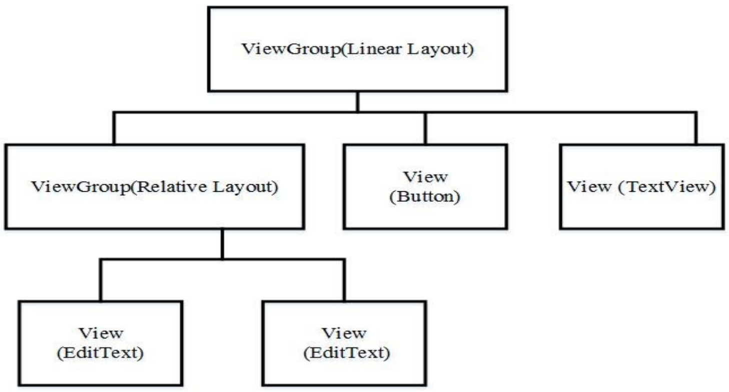


Figure 4.5. View Group in android UI

Here in above Diagram ViewGroup (Linear Layout) contains one ViewGroup (i.e. Relative Layout) and two View (Button and TextView). Further two more View (i.e. 2 EditText) are nested inside Relative Layout ViewGroup. It is important to note that one layout can be nested in another layout.

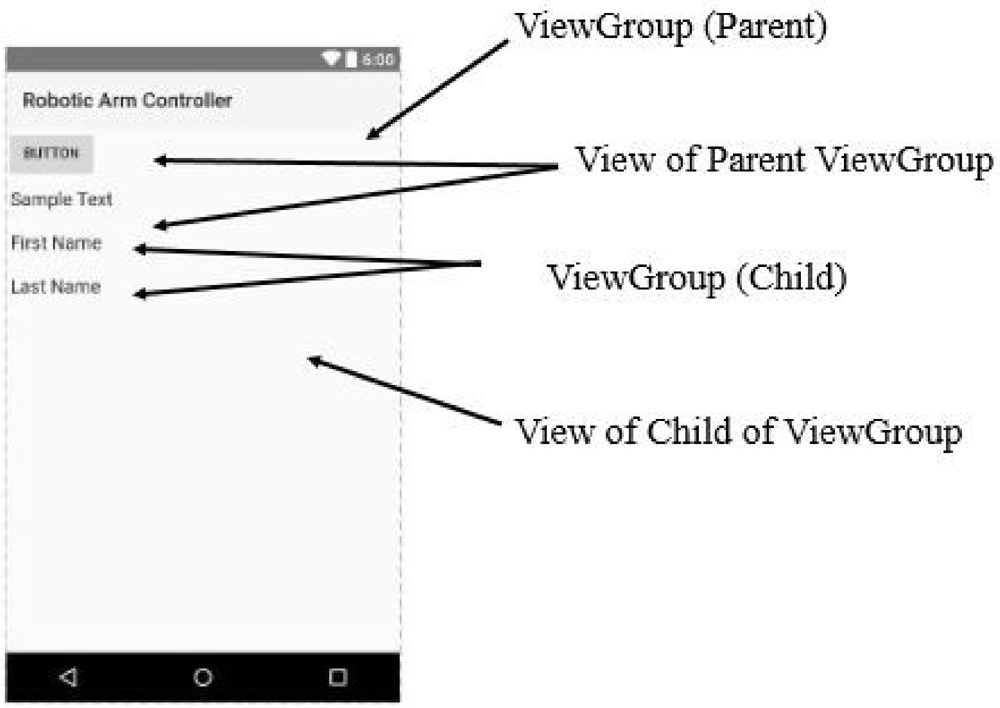


Figure 4.6. Basic UI Explanation in Android

Every Android application screen has some components like button, Text or images. These are contained inside the ViewGroup. Layouts are the best examples for ViewGroups. The different types of layout in android are Linear Layout, Relative Layout, Absolute Layout, Table Layout and Frame Layout.

4.5. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, mac OS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.1.3 released in June 2018.

4.5.1. Features

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ, and PyCharm e.g. Python, and Kotlin; and Android Studio 3.0 supports "Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features.

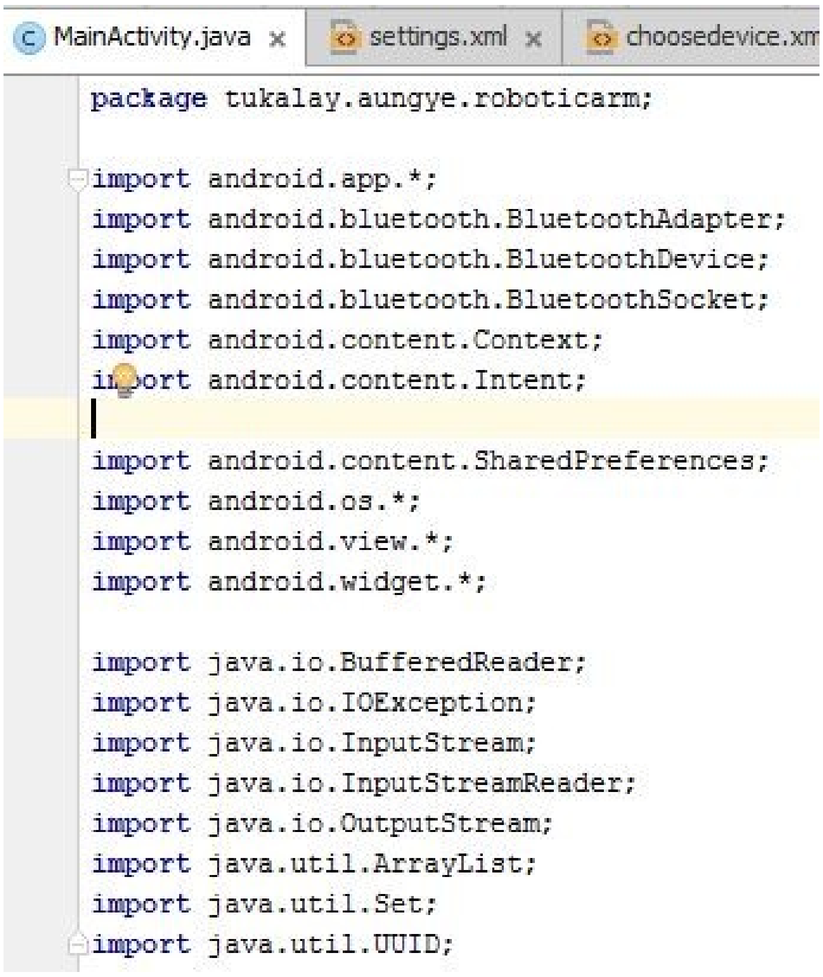
4.6. Main Activity of Robotic Arm Controller

In Mobile SDK apps, the main activity begins immediately after the user logs in. Once the main activity is running, it can launch other activities, which in turn can launch sub-activities. When the application exits, it does so by terminating the main activity. All other activities terminate in a cascade from within the main activity.

Main activity class in Robotic Arm Controller application provides the program to control the robotic arm and for UI registration. All scripts in Main Activity are as following.

4.6.1. Initialization

The program need to import packages from android and java libraries because the program use the features from this like activities, user interfaces, and mathematics.

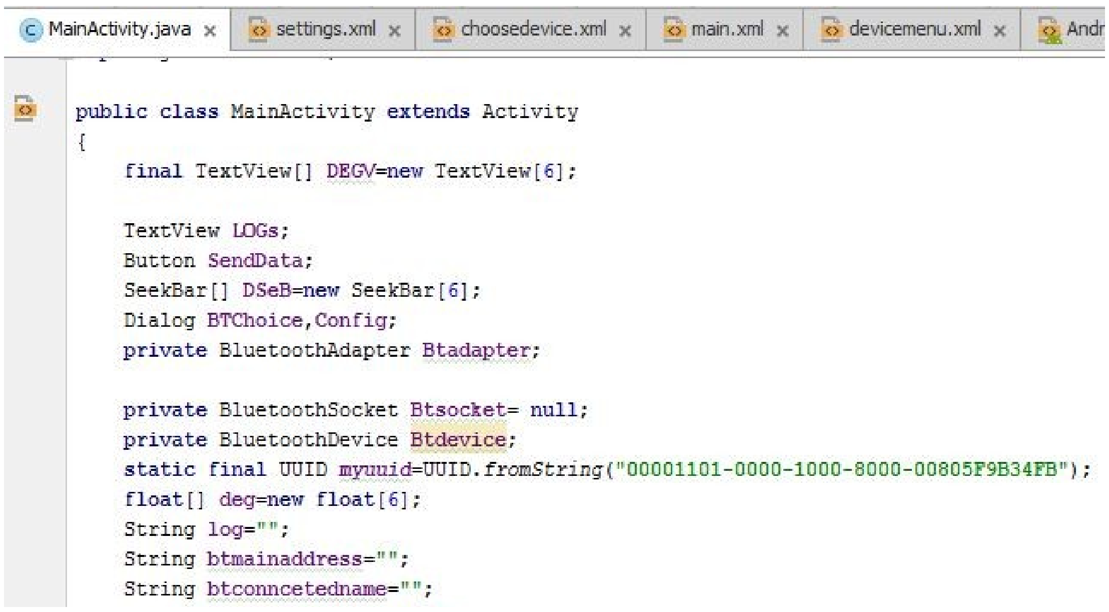


```
package tukalay.aungye.roboticarm;

import android.app.*;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;

import android.content.SharedPreferences;
import android.os.*;
import android.view.*;
import android.widget.*;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;
```

Figure 4.7. Initialization of MainActivity.java

4.6.2. Main Activity Class

The application Main Activity class extends the abstract Mobile SDK activity class, “android.os.Activity.class”. In Robotic Arm Controller application, text-views are used to show text on UI. Button widgets are also declared to be clicked from the user. String variables are used for father use of text views.

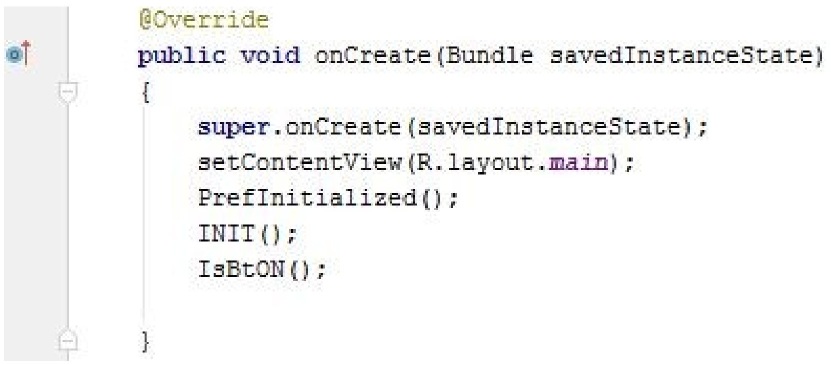


```
public class MainActivity extends Activity
{
    final TextView[] DEGV=new TextView[6];

    TextView LOGs;
    Button SendData;
    SeekBar[] DSeB=new SeekBar[6];
    Dialog BTChoice,Config;
    private BluetoothAdapter Btadapter;

    private BluetoothSocket Btsocket= null;
    private BluetoothDevice Btdevice;
    static final UUID myuuid=UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    float[] deg=new float[6];
    String log="";
    String btmainaddress="";
    String btconncetedname="";
```

Figure. 4.8. Initialization of MainActivity.Class

4.6.3. Create Methods

When an Activity first call or launched then “onCreate()” method is responsible to create the activity. Whenever orientation (i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed or when an Activity gets forcefully terminated by any Operating System then “savedInstanceState” i.e. object of Bundle Class will save the state of an Activity. After Orientation changed then “onCreate()” will call and recreate the activity and load all data from “savedInstanceState”. Basically Bundle class is used to store the data of activity whenever above condition occur in app. The “onCreate()” is not required for apps. But the reason it is used in app is because that method is the best place to put initialization code. You could also put your initialization code in “onStart()” or “onResume()” and when you app will load first, it will work same as in “onCreate()”.

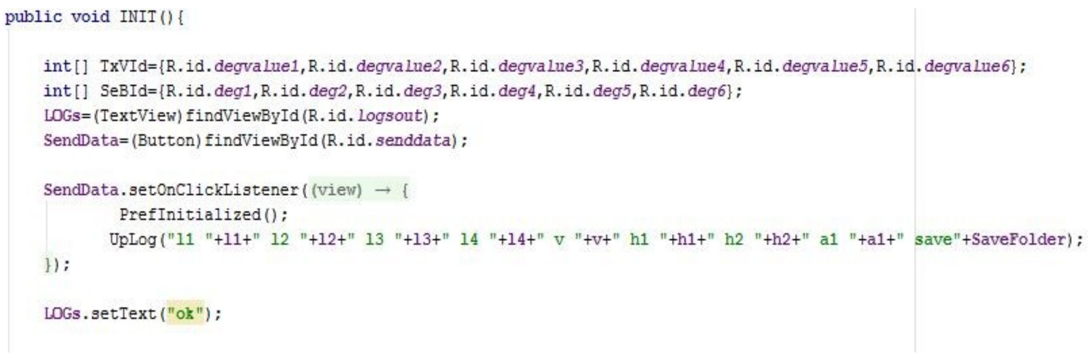


```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    PrefInitialized();
    INIT();
    IsBtON();

}
```

Figure. 4.9. “onCreate” Method in Main Activity

4.6.4. Initialization of Bluetooth

The application require the Bluetooth adapter activity so that the program need to declare "IsBtOn" function. If the adapter is disabled or empty, the program will return null.

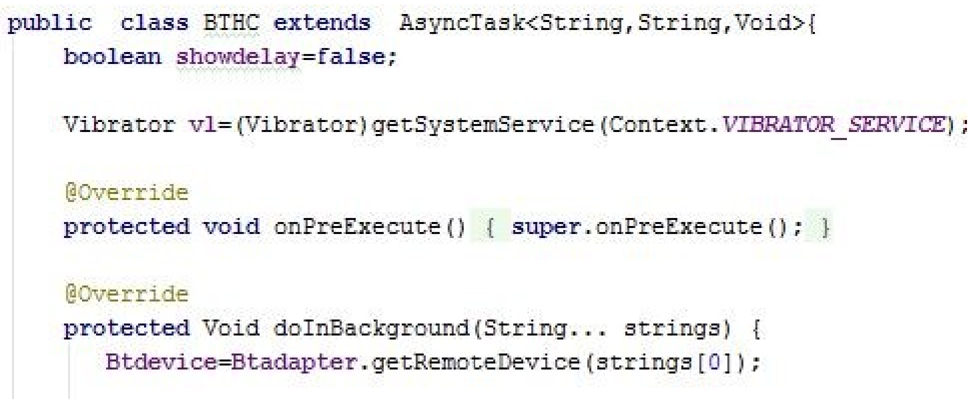


```
public void INIT(){

    int[] TxVId={R.id.degvalue1,R.id.degvalue2,R.id.degvalue3,R.id.degvalue4,R.id.degvalue5,R.id.degvalue6};
    int[] SeBId={R.id.deg1,R.id.deg2,R.id.deg3,R.id.deg4,R.id.deg5,R.id.deg6};
    LOGs=(TextView)findViewById(R.id.logsout);
    SendData=(Button)findViewById(R.id.senddata);

    SendData.setOnClickListener((view) → {
            PrefInitialized();
           UpLog("l1 "+l1+" l2 "+l2+" l3 "+l3+" l4 "+l4+" v "+v+" h1 "+h1+" h2 "+h2+" a1 "+a1+" save"+SaveFolder);
    });

    LOGs.setText("ok");
```

Figure. 4.10. Initialization program for checking Bluetooth Adapter

4.6.5. Bluetooth Device Connection

When the user want to connect the Bluetooth device of robotic arm, the program will create insecure frequency socket with a UUID by calling the "BTHC()" class. This class extends the "AsyncTask<>" class to run the scrip in background. The program will vibrate 250ms after the android phone if the Bluetooth device is connected by programming "vl.vibrate(250)" function. The function "while()" is used to receive the messages from robotic arm.

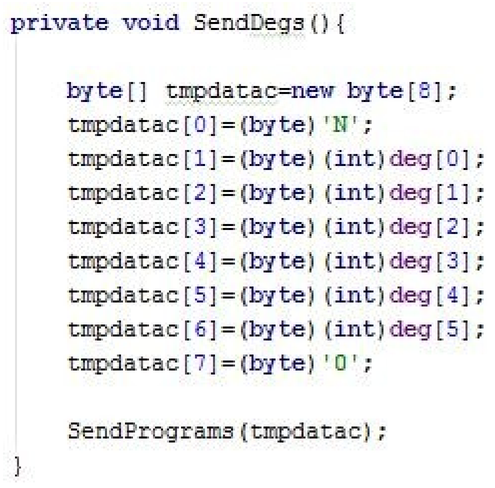


```
public  class BTHC extends  AsyncTask<String,String,Void>{
    boolean showdelay=false;

    Vibrator vl=(Vibrator)getSystemService(Context.VIBRATOR_SERVICE);

    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected Void doInBackground(String... strings) {
        Btdevice=Btadapter.getRemoteDevice(strings[0]);
```

Figure. 4.11. BTHC class for Bluetooth socket

4.6.6. Data Initialization

The program set a byte array from an array of degrees which have seven units because robotic arm uses five axes, one griper and for start and stop byte.

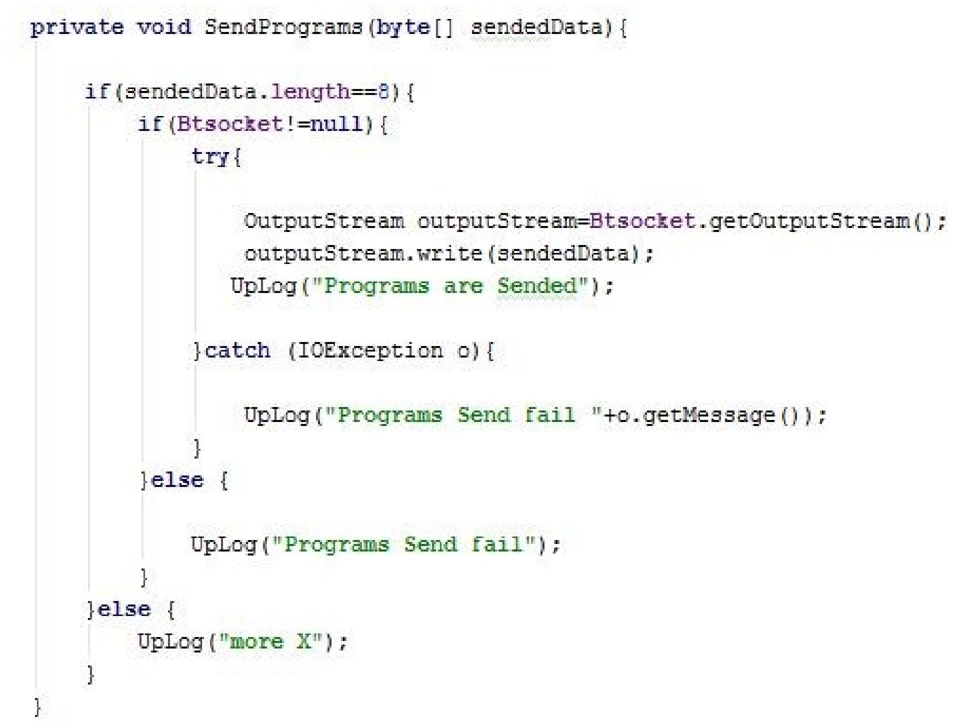


```
private void SendDegs(){

    byte[] tmpdatac=new byte[8];
    tmpdatac[0]=(byte)'N';
    tmpdatac[1]=(byte)(int)deg[0];
    tmpdatac[2]=(byte)(int)deg[1];
    tmpdatac[3]=(byte)(int)deg[2];
    tmpdatac[4]=(byte)(int)deg[3];
    tmpdatac[5]=(byte)(int)deg[4];
    tmpdatac[6]=(byte)(int)deg[5];
    tmpdatac[7]=(byte)'O';

    SendPrograms(tmpdatac);
}
```

Figure. 4.12. SendDegs() function for initialization data

4.6.7. Data Transmission Function

The microcontroller of robotic arm receives the data by byte units so that the program declare array of byte with the function of “SendPrograms(byte[])”. The program also declare the script “Btsocket.getOutputStream()” as the transmission need the output stream from the Bluetooth socket. For the error of getting output stream, the program use “try-catch” function for exception of input socket and output socket.



```
private void SendPrograms(byte[] sendedData){

    if(sendedData.length==8){
        if(Btsocket!=null){
            try{

                OutputStream outputStream=Btsocket.getOutputStream();
                outputStream.write(sendedData);
               UpLog("Programs are Sended");

            }catch (IOException o){

                UpLog("Programs Send fail "+o.getMessage());
            }
        }else {

            UpLog("Programs Send fail");
        }
    }else {
        UpLog("more X");
    }
}
```

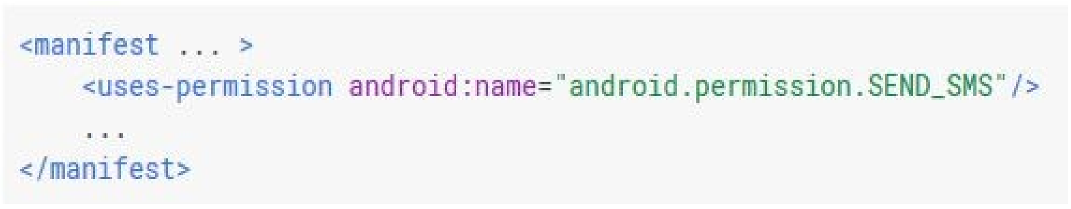
Figure. 4.13. Send program() function for data transmission

4.7. Robotic Arm Controller Application

Robotic Arm Controller application is the android software used to control the robotic arm via Bluetooth. This is developed using java program with Android Studio software. The package name of this application must be unique to avoid override installation and it is "tukalay.aungye.roboticarm".

4.7.1. Permissions

Android apps must request permission to access sensitive user data (such as contacts and SMS) or certain system features (such as the camera and internet access). Each permission is identified by a unique label. For example, an app that needs to send SMS messages must have the following line in the manifest:



```
<manifest ... >
    <uses-permission android:name="android.permission.SEND_SMS"/>
    ...
</manifest>
```

Figure 4.14. Example of request permission line in manifest

Robotic arm controller application need to request the following permission

- Bluetooth
- Read/Write External Storage
- Vibrate

4.7.2. Controller UI design

The layouts for this application is programmed using xml language. There are four UIs used in Robotic Arm Controller application. They are

- Main Interface
- Menu Interface
- Bluetooth Devices Interface
- Configuration Interface

4.7.2.1. Main Interface

Main interface is the first interface when the application is lunched. This interface contains controller joint stick and joints seek bars to control the robotic arm.

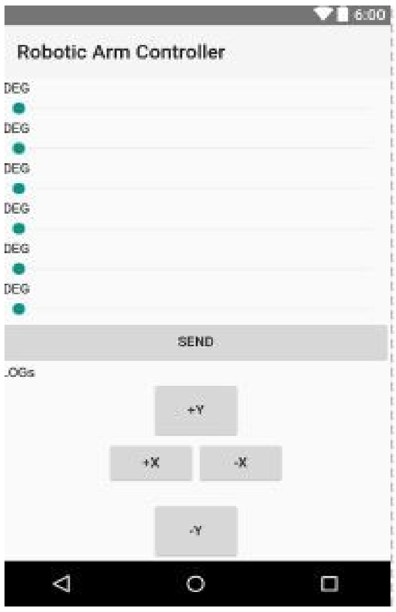


Figure 4.15. Main user interface in Robotic Arm Controller

4.7.2.2. Menu Interface

This interface will appear when the menu button is clicked in android phone running Robotic Arm Controller application. This interface has a list of option to lunch other interfaces.

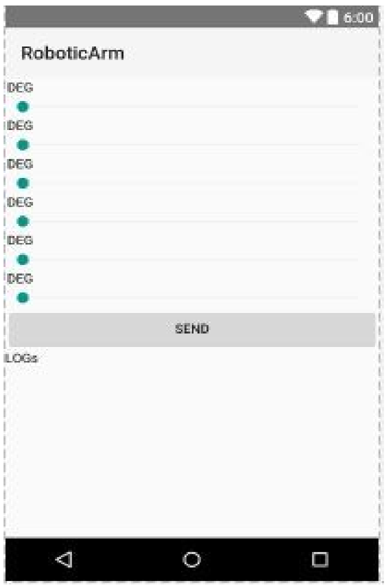
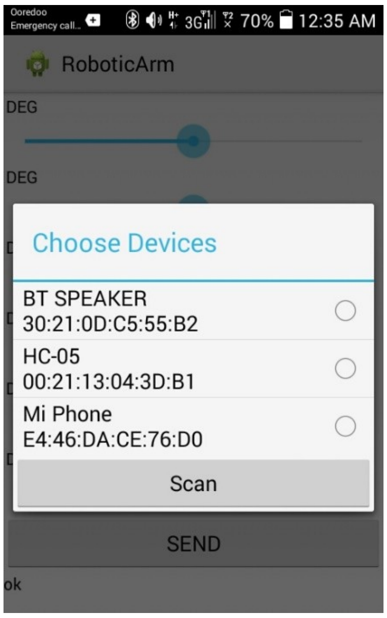


Figure 4.16. Menu UI in Robotic Arm Controller

4.7.2.3. Bluetooth Device Interface

This interface will appear when the user choose Choose Device option from the Menu Interface. List of Bluetooth device with physical address will be shown in this interface and must be choose the device (HC-05) that is used in robotic am. Scan button is used to scan when the devices are missing.



Choose Devices

BT SPEAKER
30:21:0D:C5:55:B2

HC-05
00:21:13:04:3D:B1

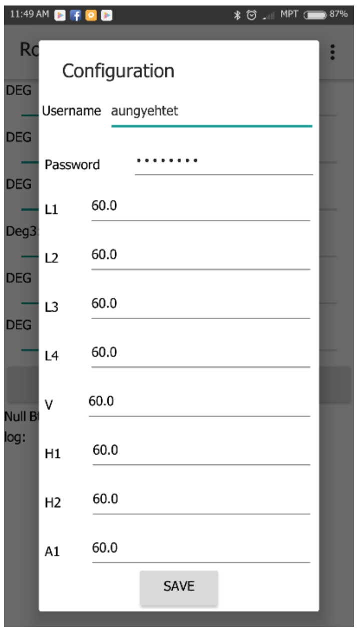
Mi Phone
E4:46:DA:CE:76:D0

Scan

Figure 4.17. Bluetooth Devices Interface

4.7.2.4. Configuration Interface

This interface is used to configure the dimensions of the robotic arm. Many edit-text box will be shown in this interface. Save button can be click when the configuration is finish or to save the configuration.


Configuration

Username	aungyehtet
Password	•••••••••
L1	60.0
L2	60.0
L3	60.0
L4	60.0
V	60.0
H1	60.0
H2	60.0
A1	60.0

Figure 4.18. Configuration Interface

4.8. Summary

The android software implementation for robotic arm is programmed using the Android Studio Software. The test and result of the system will be described in the next chapter.