Syrian Private University

Faculty of Computer & Informatics Engineering

Design and the mechanism of controlling a robotic arm

A Senior Project Presented to the Faculty of Computer and Informatics Engineering In Partial Fulfillment of the Requirements for the Degree

Of Bachelor of in Communication & Network Engineering

Under the supervision of

Prof. Ahmad Rateb Al-Najjar **Prof. Ali Skaff**

By

Noor Ali Al teef

Yousef Sofyan Jghef

Hasan Mohamed Heddeh

Mohamed Zaki Hani Ibrahim

Mohammed Ali Ismail Abdul Rahman

August 2015

CERTIFICATION OF APPROVAL

Prof. Ahmad Rateb Al-Najjar Date :

Prof. Ali Skaff Date :

CONTENT

Table of Figure

الملخص باللغة العربية

تمحورت الدراسة الحالية حول تصميم وبرمجة ذراع آلية. تم تصميم الذراع الآلية ضمن خمس درجات من الحرية وتم تزويدها بالمطلوب لتنجز المهام البسيطة بشكلٍ دقيق. وكمثالٍ على ذلك فإن المادة الخفيفة الطبيعة التي تكسو الذراع الآلية مزوّدة بمحركات سيرفو والتي تقوم بعملية الربط بين الأذرع، بالإضافة إلى تأدية حركات الذراع. إن المتحكم الذي يقود المحركات تلك لديه القدرة على تعديل الوضعية.

تتم عملية الترجمة على المتحكم من نوع/نمط ATMEGA-328p باستخدام برمجة Arduino. يتم استخدام مقاييس الجهد أيضاً للكشف عن زاوية الدورات والإشارات المُرسَلة إلى المتحكم. بالإمكان التحكم بالذراع الآلية أيضاً باستخدام جهاز أندرويد. ففي عالم اليوم، تبين أن هذه الذراع الآلية جالبة للنفع العام. بالإضافة إلى الروبوتات والأتمتة، فإن هذه الأنواع من الأذرع تمتاز بتنوع تطبيقاتها المتاحة في عدة مجالات أخرى.

CHAPTER 1: INTRODUCTION

A Robot is a virtually intelligent agent capable of carrying out tasks robotically with the help of some supervision. Practically, a robot is basically an electro-mechanical machine that is guided by means of computer and electronic programming. Robots can be classified as autonomous, semiautonomous and remotely controlled. Robots are widely used for variety of tasks such as service stations, cleaning drains, and in tasks that are considered too dangerous to be performed by humans. A robotic arm is a robotic manipulator, usually programmable, with similar functions to a human arm.

This Robotic arm is programmable in nature and it can be manipulated. 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 tasks involved in the manufacturing industry by themselves. However, a Robotic arm can be used for various tasks such as welding, drilling, spraying and many more. A self-sufficient robotic arm is fabricated by using components like micro-controllers and motors. This increases their speed of operation and reduces the complexity. It also brings about an increase in productivity which makes it easy to shift to hazardous materials. The main part of the design is ATMEGA-328p micro-controller which coordinates and controls the product's action. This specific micro controller is used in various types of embedded applications. Robotics involves elements of mechanical and electrical engineering, as well as control theory, computing and now artificial intelligence. According to the Robot Institute of America, "A robot is a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.

The robots interact with their environment, which is an important objective in the development of robots. This interaction is commonly established by means of some sort of arm and gripping device or end effectors. In the robotic arm, the arm has a few joints, similar to a human arm, in addition to shoulder, elbow, and wrist, coupled with the finger

joints; there are many joints . The design process is clearly explained in the next section with detailed information regarding the components which are used.

CHAPTER 2: DESIGNING PORTION

2.1 Design of Robotic Arm

The Robotic Arm is designed using the Microcontroller i.e. ATMEGA328p Micro-controller using Arduino programming. This process works on the principle of interfacing servos and potentiometers. This task is achieved by using Arduino board. Potentiometers play an important role The remote is fitted with potentiometers and the servos are attached to the body of the robotic arm. The potentiometer converts the mechanical motion into electrical motion. Hence, on the motion of the remote the potentiometers produce the electrical pulses, which are in route for the Arduino board. The board then processes the signals received from the potentiometers and finally, converts them into requisite digital pulses that are then sent to the servomotors. This servo will respond with regards to the pulses which results in the moment of the arm.

Figure 2.1 shows the image of a servo motor. It consists of a motor which is coupled to a sensor, used for position feedback, through a reduction gearbox. It also accompanies a relatively sophisticated controller, usually a dedicated module designed specifically for use with servo motors

In short, the micro controller interfaces all these components specified below. A short list of components include

1. Servo motors
2. Potentiometers
3. Atmega 328p.
4. Arduino Deumilanove "IDE"
5. Bloutooth module.

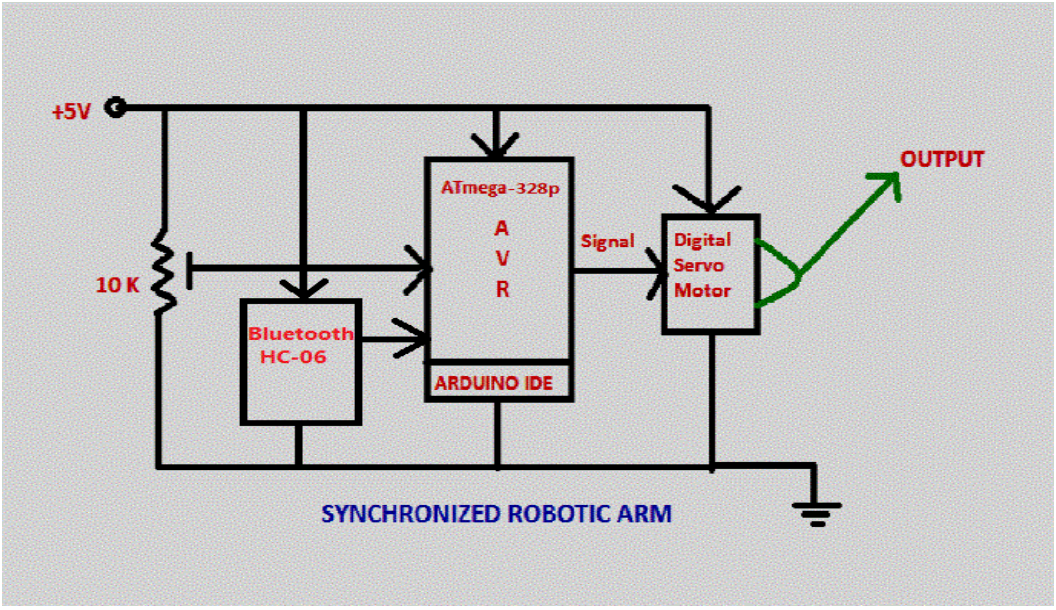


Figure2.1 shows the image of a servo motor

2.2 Five degrees of freedom

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

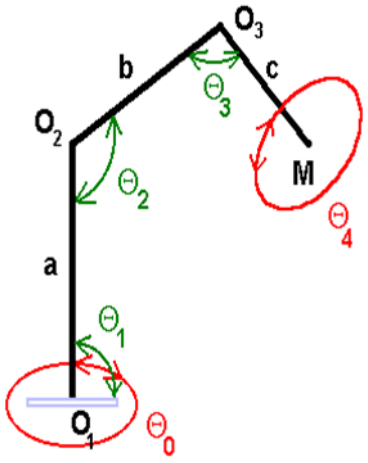


Figure2.2 Five degrees of freedom

Robot arms are described by their degrees of freedom. This number typically refers to the number of single-axis rotational joints in the arm, where higher number indicates an increased flexibility in positioning a tool. This is a practical metric, in contrast to the abstract definition of degrees of freedom which measures the aggregate positioning capability of a system.

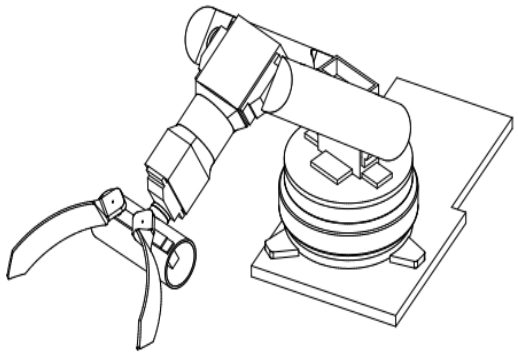


Figure2.3 Robot Arm

2.3 Positive and negative on robotic arm

2.3.1 The Positive:

- Increase productivity
- Use equipment effectively
- Reduce working costs
- Flexibility at work
- Get the job done in the shortest time
- Provide good returns on investment
- Better accuracy in performance
- Ability to work in risky ways and make it more safe

2.3.2 The negative:

- Cause unemployment for manual workers
- High initial cost
- designed Arm to perform specific tasks and not comparable to the human hand
- Difficulty programmed to perform Accurate tasks
- Needed a large number of sensors and high accuracy to perform the Complex tasks
- And other technical problems, "especially in the fields of artificial intelligence and Machine vision" .
- When the Robotic arm break down the production line will go off in the factories.

CHAPTER 3: THEORETICAL PORTION

3.1 Arduino

Arduino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

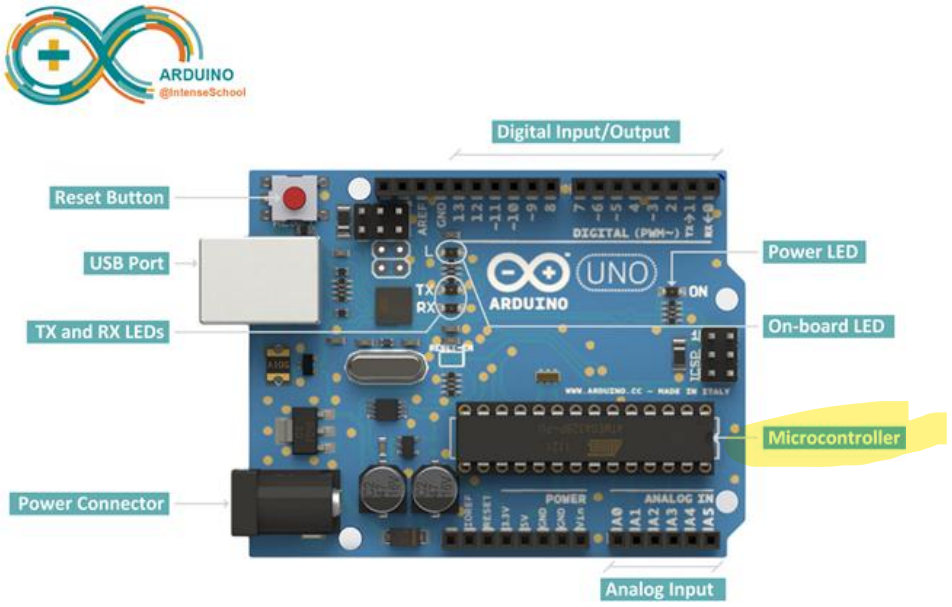


Figure3.1 Arduino Uno microcontroller board (interface)

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

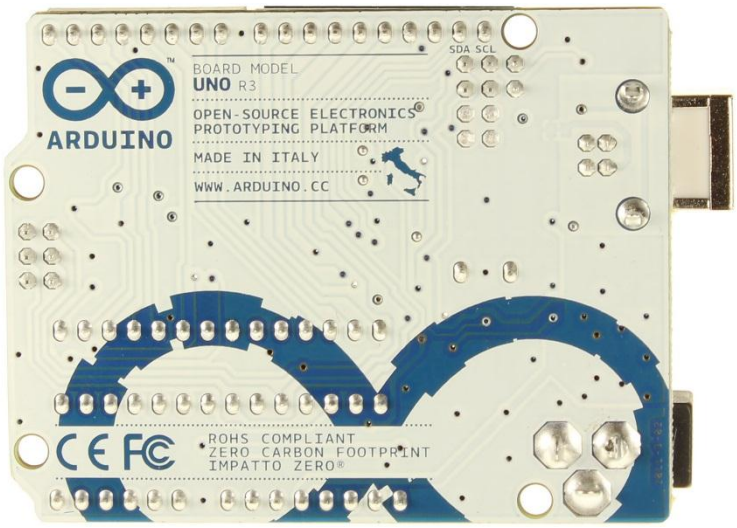


Figure 3.2 Arduino Uno microcontroller board (back view)

3.2 Specifications

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)

	of which 0.5 KB used by bootloade*r
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.3 Programming

The Arduino Uno can be programmed with the (Arduino Software (IDE)).The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

3.4 Power

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

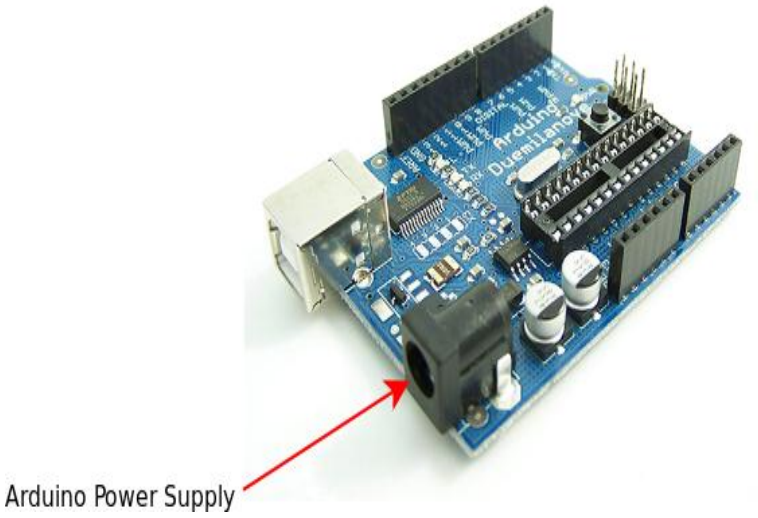


Figure 3.3 Arduino Power Supply

3.5 Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM

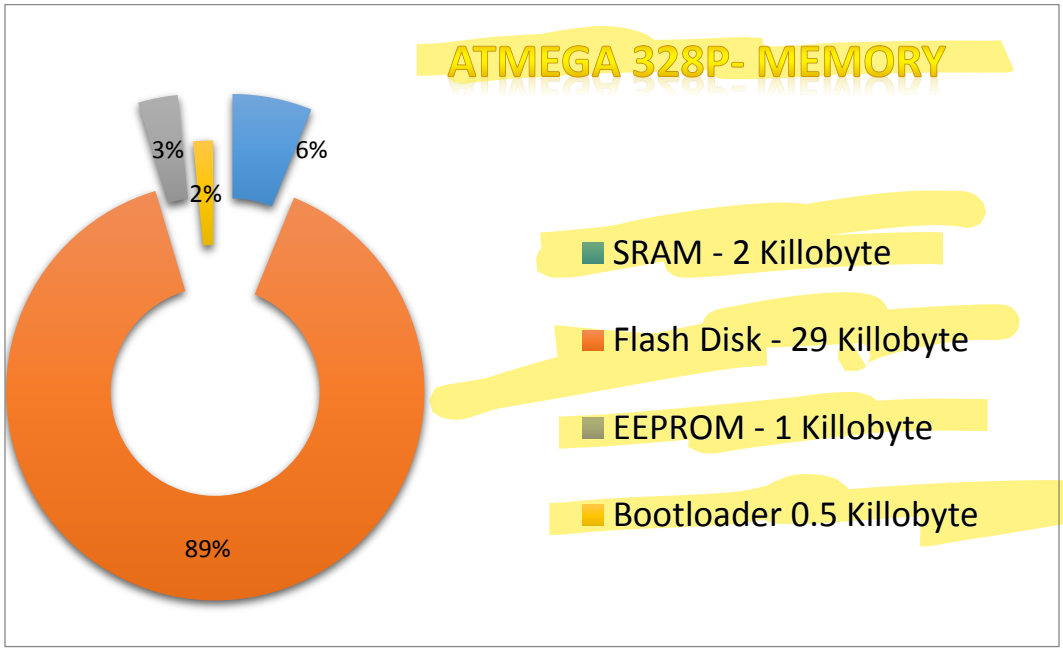


Figure 3.4 ATmega328P – Memory

3.6 Arduino development "IDE"

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface

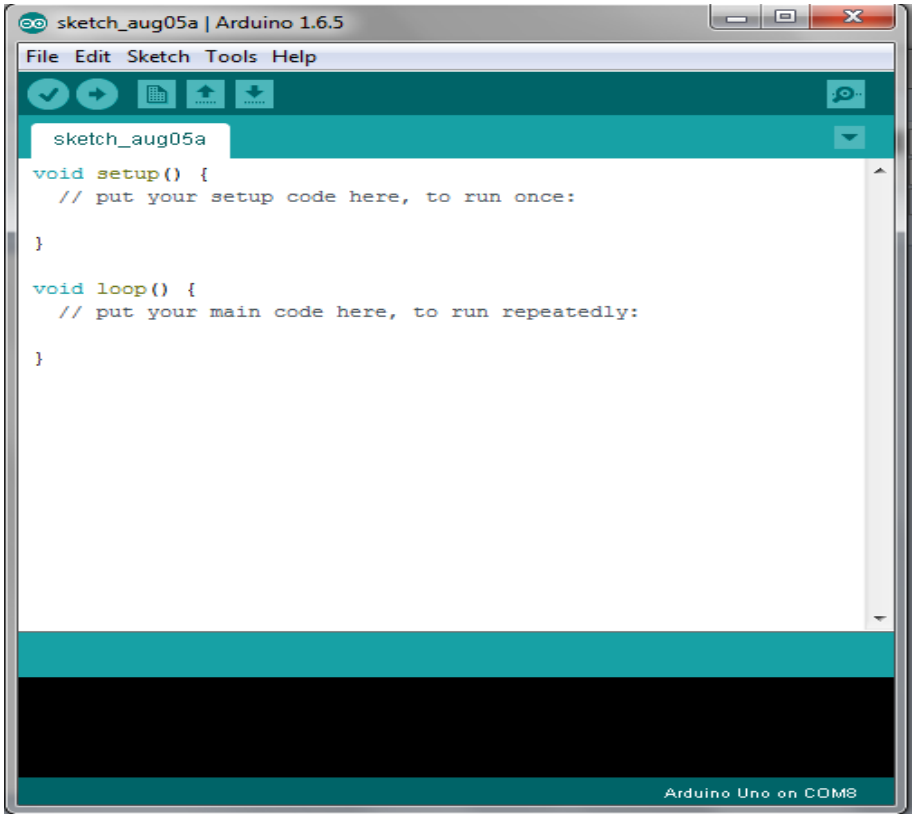


Figure 3.5 Interface Arduino Uno Program

Developer(s)	Arduino Software
Stable release	1.0.3 / December 10, 2012; 3months ago
Written in	Java, C and C++
Operating system	Cross-platform
Type	Integrated development environment
Website	arduino.cc

Arduino programs are written in C or C++ The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output.

Operations much easier. Users only need define two functions.

To make a runnable cyclic executive program:

- **Setup ():** a function run once at the start of a program that can initialize settings.
- **Loop ():** a function called repeatedly until the board powers off.

CHAPTER 4: PRACTICAL PORTION

4.1 Servo Motors

Servo refers to an error sensing feedback control which is used to correct the performance of a system. Servo or RC Servo Motors are DC motors equipped with a servo mechanism for precise control of angular position.

The RC servo motors usually have a rotation limit from 90° to 180°. But servos do not rotate continually. Their rotation is restricted in between the fixed angles.

The Servos are used for precision positioning. They are used in robotic arms and legs, sensor scanners and in RC toys like RC helicopter, airplanes and cars.

The specifications for big Servomotor used are as follows:

- Weight- 55g
- Dimension- 40.7*19.7*42.9mm
- Stall torque- 10kg/cm
- Operating speed-0.20sec/60degree(4.8v)
- Operating voltage 4.8-7.2V.
- Temperature range 0-55 degrees.

Figure 4.1 Big Servo Motor

The specifications for small Servomotor G9 used are as follows:

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm • Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC

Figure 4.2 G9 Servo Motor

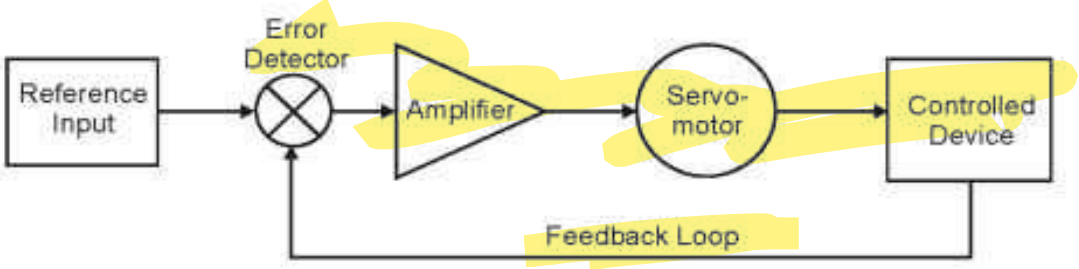


Figure 4.3 Block diagram of a Servo motor

4.2 Theory of DC Servo Motor

As we know that any electrical motor can be utilized as servo motor if it is controlled by servomechanism. Likewise, if we control a DC motor by means of servomechanism, it would be referred as **DC servo motor**. There are different types of DC motor, such shunt wound DC motor, series DC motor, Separately excited DC motor, permanent magnet DC motor, Brushless DC motor etc. Among all mainly separately excited DC motor, permanent magnet DC motor and brush less DC motor are used as servo.

4.3 Separately Excited DC Servo Motor

Figure (4.4) shows the block diagram of the separately excited Dc servo motor

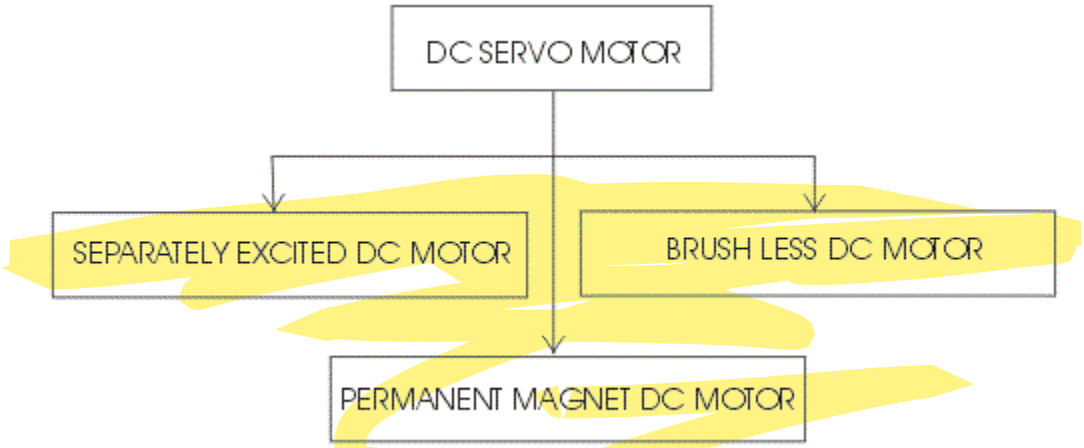


Figure 4.4 Separately Excited DC Servo Motor

4.3.1 DC Servo Motor Theory:

The motors which are utilized as **DC servo motors**, generally have separate DC source for field winding and armature winding. The control can be archived either by controlling the field current or armature current. Field control has some specific advantages over armature control and on the other hand armature control has also some specific advantages over field control. Which type of control should be applied to the **DC**

servo motor, is being decided depending upon its specific applications. Let's discus **DC servo motor working principle** for field control and armature control one by one.

Field Controlled DC Servo Motor Theory

The Figure below illustrates the schematic diagram for a field controlled DC servo motor. In this arrangement the field of DC motor is excited be the amplified error signal and armature winding is energized by a constant current source

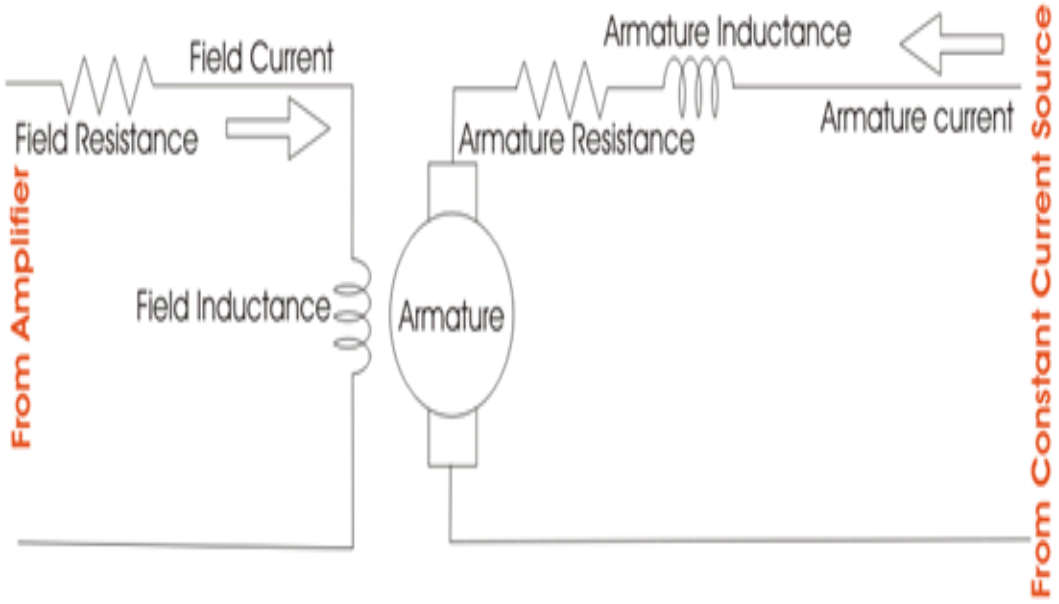


Figure 4.5 Field controlled DC servo motor

The field is controlled below the knee point of magnetizing saturation curve. At that portion of the curve the MMF linearly varies with excitation current. That means torque developed in the DC motor is directly proportional to the field current below the knee point of magnetizing saturation curve.

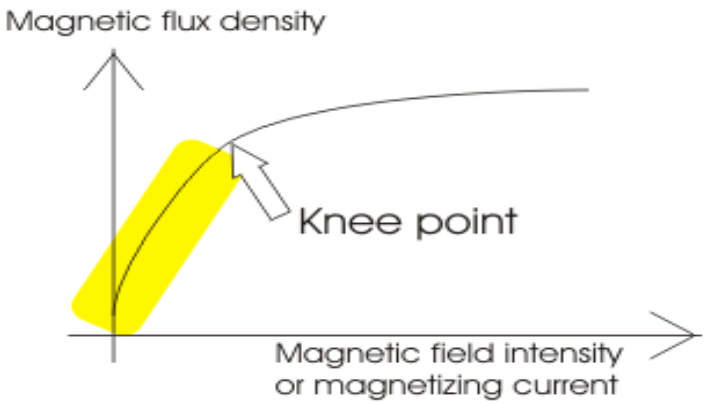


Figure 4.6 knee point of magnetizing saturation curve

From general torque equation of DC motor it is found that, torque $T \propto \varphi I_a$. Where, φ is field flux and I_a is armature current. But in field controlled DC servo motor, the armature is excited by constant current source , hence I_a is constant here. Hence, $T \propto \varphi$

As field of this DC servo motor is excited by amplified error signal, the torque of the motor i.e. rotation of the motor can be controlled by amplified error signal. If the constant armature current is large enough then, every little change in field current causes corresponding change in torque on the motor shaft.

The direction of rotation can be changed by changing polarity of the field.

The direction of rotation can also be altered by using split field DC motor, where the field winding is divided into two parts, one half of the winding is wound in clockwise direction and other half in wound in anticlockwise direction. The amplified error signal is fed to the junction point of these two halves of the field as shown below. The magnetic field of both halves of the field winding opposes each other. During operation of the motor, magnetic field strength of one half dominates other depending upon the value of amplified error signal fed between these halves. Due to this, the DC servo motor rotates in a particular direction according to the amplified error signal voltage.

The main disadvantage of field control **DC servo motors**, is that the dynamic response to the error is slower because of longer time constant of inductive field circuit. The field is an electromagnet so it is basically a highly inductive circuit hence due to sudden change in error signal voltage, the current through the field will reach to its steady state value after certain period depending upon the time constant of the field circuit. That is why field control DC servo motor arrangement is mainly used in small servo motor applications.

The main advantage of using field control scheme is that, as the motor is controlled by field. The controlling power requirement is much lower than rated power of the motor.

4.3.2 Armature Controlled DC Servo Motor Theory:

Figure (4.5) shows the schematic diagram for an armature controlled DC servo motor. Here the armature is energized by amplified error signal and field is excited by a constant current source .

The field is operated at well beyond the knee point of magnetizing saturation curve. In this portion of the curve, for huge change in magnetizing current, there is very small change in mmf in the motor field. This makes the servo motor is less sensitive to change in field current. Actually for armature controlled DC servo motor, we do not want that, the motor should response to any change of field current.

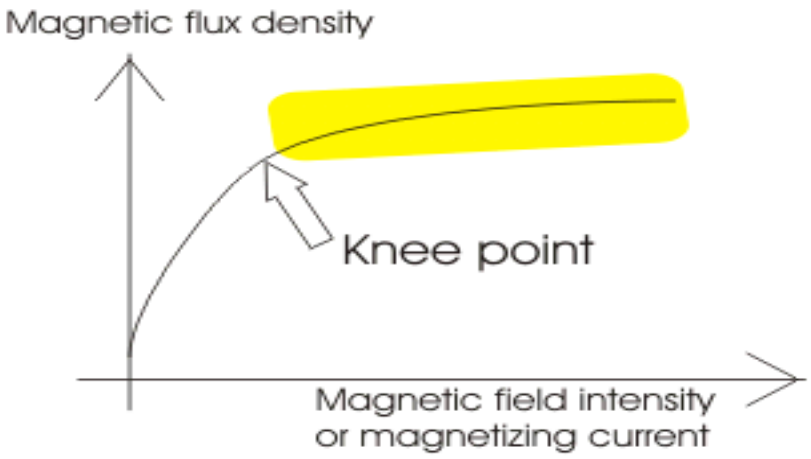


Figure 4.7 knee point of magnetizing saturation

Again, at saturation the field flux is maximum. As we said earlier, the general torque equation of DC motor is, torque $T \propto \varphi I_a$. Now if φ is large enough, for every little change in armature current I_a there will be a prominent changer in motor torque. That means servo motor becomes much sensitive to the armature current.

As the armature of DC motor is less inductive and more resistive, time constant of armature winding is small enough. This causes quick change of armature current due to sudden change in armature voltage.

That is why dynamic response of armature controlled DC servo motor is much faster than that of field controlled DC servo motor.

The direction of rotation of the motor can easily be changed by reversing the polarity of the error signal.

4.3.3 Permanent Magnet DC Servo Motor:

Field control is not possible in the case of permanent magnet DC motor as the field is a permanent magnet here. DC servo motor working principle in that case is similar to that of armature controlled motor.

4.4 Deriving State Equations for a DC Servo Motor

4.4.1System Model:

A useful component in many real control systems is a permanent magnet DC servo motor. The input signal to the motor is the armature voltage Va(t), and the output signal is the angular position θ(t). A schematic diagram for the motor is shown in Figure. (4.4.1.1). The terms Ra and La are the resistance and inductance of the armature winding in the motor, respectively. The voltage Vb is the back EMF generated internally in the motor by the angular rotation. J is the inertia of the motor and load (assumed lumped together), and B is the damping in the motor and load relative to the fixed chassis.

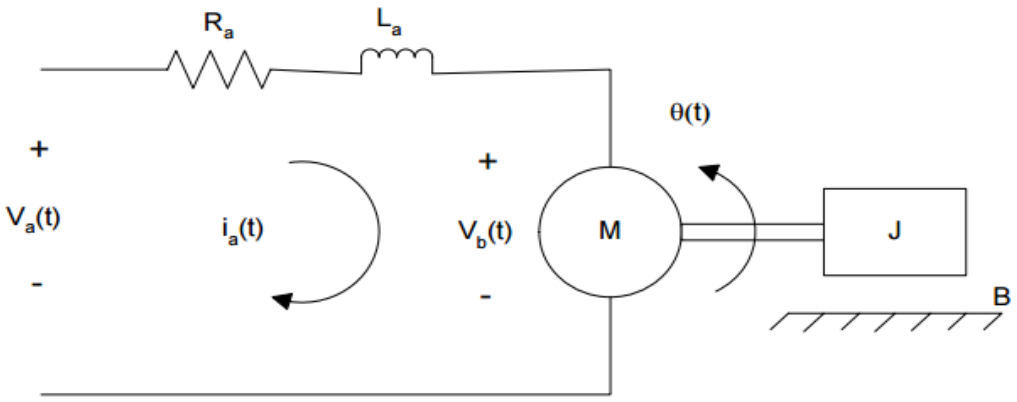


Figure 4.8 The terms Ra and La are the resistance

The equations for the electrical side of the system are

$Va(t) = Raia(t) + La\ dia(t)/dt + Vb(t)$ with $Vb(t) = Kb\ d\theta(t)/dt$ (1)

$Va(t) = Raia(t) + La\ dia(t)/dt + Kb\ d\theta(t)/dt$ (2)

Where Kb is the motor's back EMF constant. The equations for the mechanical side of the system are

$J\ d2\theta(t)/dt2 + B\ d\theta(t)/dt = Tapp(t)$ with $Tapp(t) = KT\ ia(t)$ (3)

$J\ d2\theta(t)/dt2 + B\ d\theta(t)/dt = KT\ ia(t)$ (4)

Where Tapp is the applied torque, and KT is the torque constant that relates the torque to the armature current.

4.4.2 Developing the State Equations:

Here are three derivative terms in the system model for the DC servo motor—first derivative of ia(t) in Eqn. (2) and first and second derivatives of θ(t) in Eqn. (4). Therefore, there are a total of 3 state variables in the state space model. Although the first derivative of θ(t) appears in both of those equations, it is the same variable and so does not add an additional state variable. A simulation diagram can be drawn for this system by solving Eqns. (2) and (4) for the highest derivative term in each. This yields the

expressions in (5) and (6). The simulation diagram is shown in Figure. 2. The state variables will be defined as the outputs of the integrators, with x1 being ia, x2 being θ, and x3 being dθ/dt.

dia(t) /dt = − (Ra/ La)¶ ia(t) – (Kb /La ¶) dθ(t) /dt + (1 /La)¶ Va(t) (5)

d^2θ(t)/ dt^2 = − (B/ J) ¶ dθ(t)/ dt + (KT/ J) ¶ ia(t) (6)

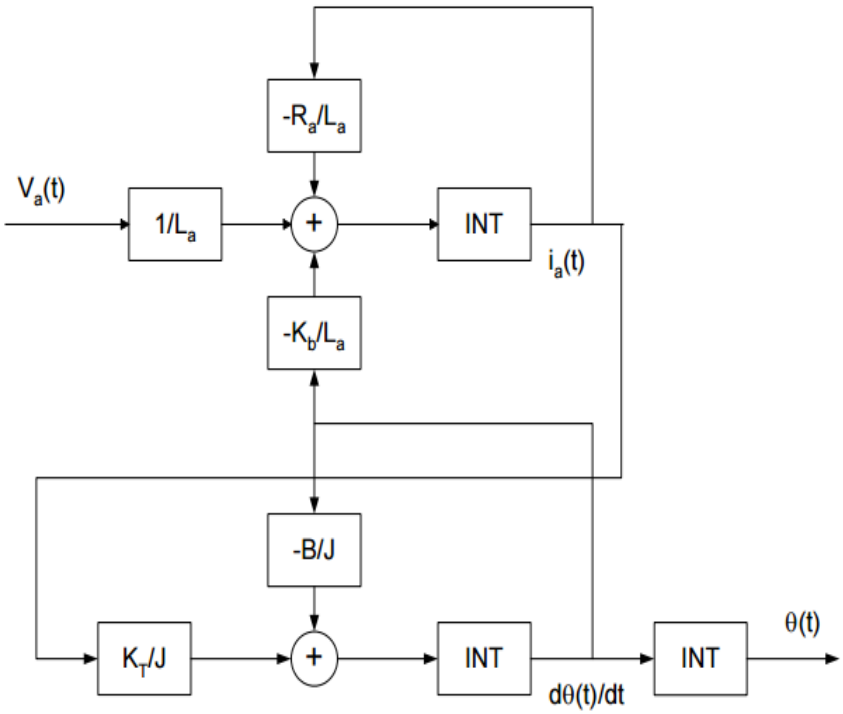


Figure 4.9 Simulation Diagram for The DC Servo Motor

Example With these definitions for the state variables, and defining

u(t) = Va(t) and y(t) = θ(t), the state and output equations are:

x1(t) = − (Ra/ La)¶ x1(t) – (Kb/ La)¶ x3(t) + (1/ La)¶ u(t) (7)

x2(t) = x3(t) (8)

x3(t) = − (B /J)¶ x3(t) + (KT /J) ¶ x1(t) (9)

$$y(t) = x2(t) \quad (10)$$

In short hand notation, the state and output equations are :

$$x(t) = Ax(t) + Bu(t), \qquad y(t) = Cx(t) \quad (11)$$

4.5 Potentiometers

A Potentiometer is a variable resistor. The two output terminals act as a fixed resistor. A movable contact called a wiper (or a slider) moves across the resistor, producing a variable resistance between the center terminals and the two sides. A potentiometer is basically a voltage divider which is used for the measurement of potential. Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers which are operated by a mechanism can also be used as position transducer.

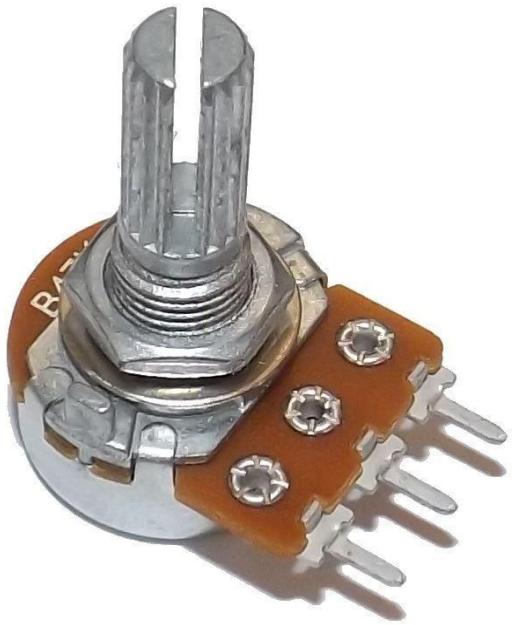


Figure 4.10 A Potentiometer

A potentiometer is a manually adjustable electrical resistor that uses three terminals. In many electrical devices, potentiometers are establishes the levels of output. For example, in case of a loudspeaker, a potentiometer is used to adjust the volume or taken the case of a television set, computer monitor or light dimmer; it can be used to control the brightness of the screen or light bulb.

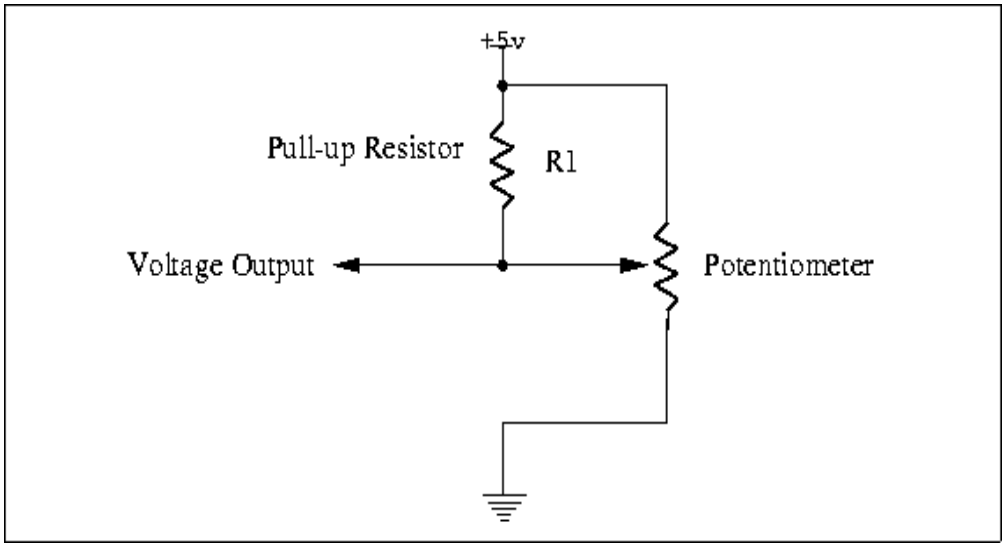


Figure 4.11 Circuit Diagram of a Potentiometer

4.6 Bluetooth HC-06 module

HC.06 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps 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). It has the footprint as small as 12.7mmx27mm.

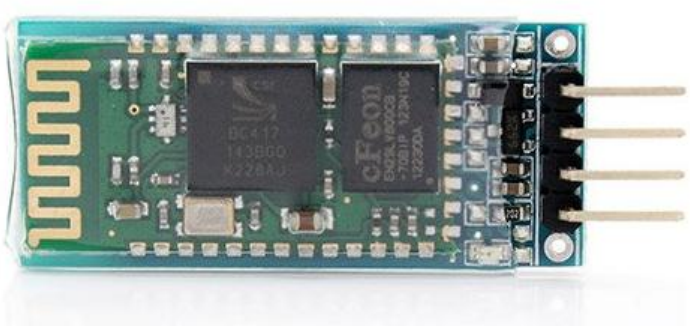


Figure 4.12 Bluetooth HC.06 module (Front View)

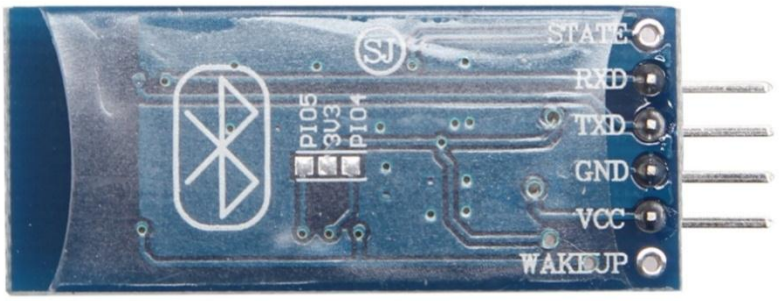


Figure 4.13 Bluetooth HC.06 module(Back View)

4.7 Specifications

4.7.1 Hardware features

- Typical .80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

4.7.2 Software features

- Default Baud rate: 38400, Data bits: 8, Stop bit: 1, Parity: No parity, Data

Control: has supported baud rate:
9600,19200,38400,57600,115200,230400,460800.

- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low disconnected, high connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto pairing PINCODE:"0000" as default
- Auto reconnect in 30 min when disconnected as a result of beyond the range of connection.

CHAPTER 5: MECHANICAL PORTION

5.1Mechanical Design

The mechanical design of the robot arm is based on a robot manipulator with similar functions to a human arm . The links of such a manipulator are connected by joints allowing rotational motion and the links of the manipulator is considered to form a kinematic chain. The business end of the kinematic chain of the manipulator is called the end effector or end of arm tooling and it is analogous to the human hand. Figure 5.1 shows the Free Body Diagram for mechanical design of the robotic arm.

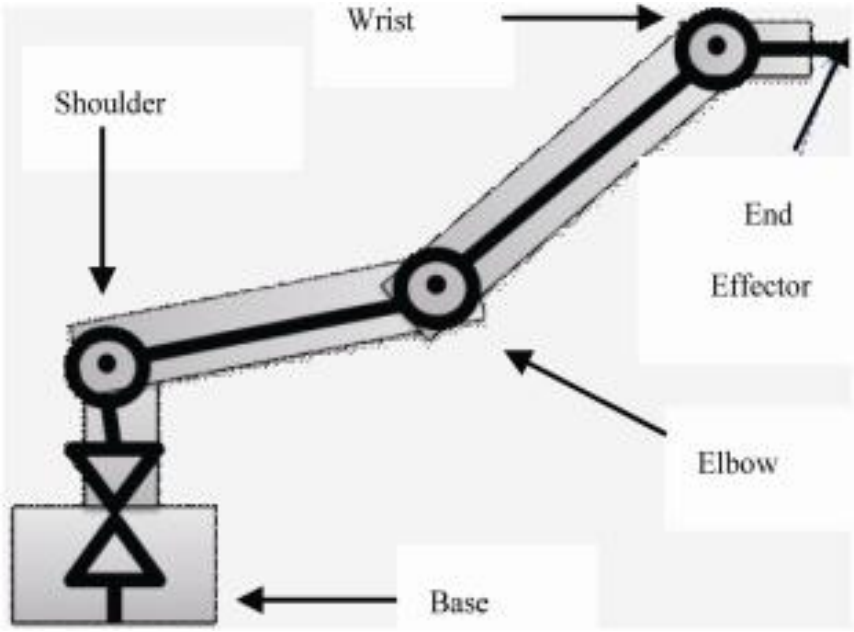
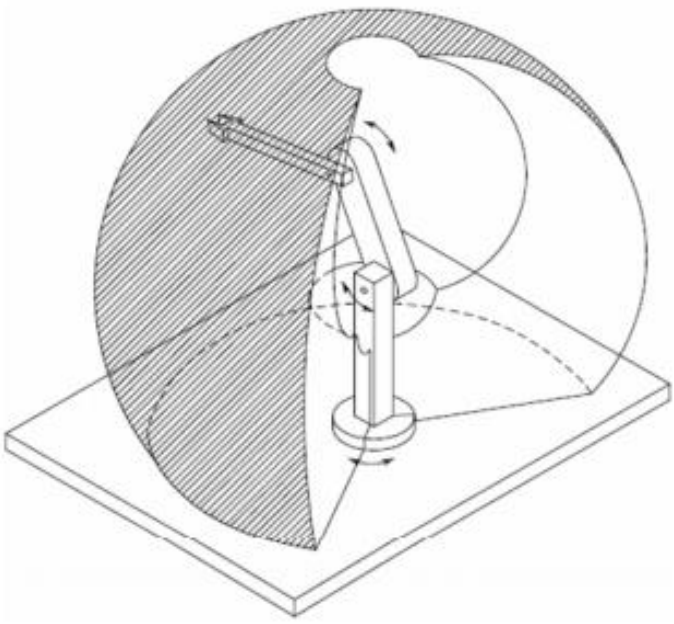


Figure 5.1 Free body diagram of the robot arm

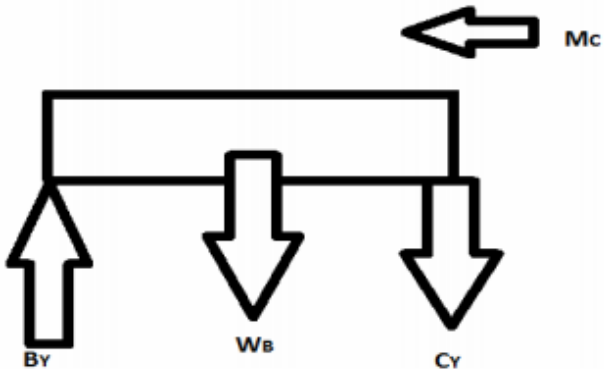
As shown, the end effector is not included in the design because a commercially available gripper is used.This is because that the end effector is one of the most complex parts of the system and, in turn, it is much easier and economical to use a commercial one than build it. Figure 5.17 shows the work region of the robotic arm. This is the typical workspace of a robot arm with four degree of freedom (6 DOF). . The mechanical design

was limited to 6 DOF mainly because that such a design allows most of the necessary movements and keeps the costs and the complexity of the robot competitively Accordingly, rotational motion of the joints is restricted where rotation is done around two axis in the shoulder and around only one in the elbow and the wrist, see Figure 5.18. The robot arm joints are typically actuated by electrical motors. The servo motors were chosen, since they include encoders which automatically provide feedback to the motors



and adjust the position accordingly.

Figure 5.2 Work region of the robotic arm



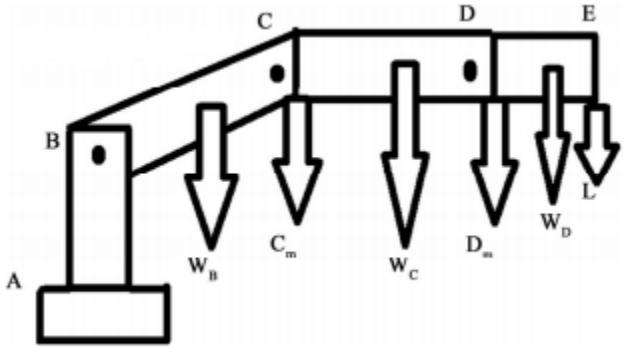


Figure 5.3 Force diagram of robot arm

Figure 5.4 Force diagram of link CB

However, the disadvantage of these motors is that rotation range is less than 180˚ span, which greatly decreases the region reached by the arm and the possible positions [9]. The qualifications of servo motors were selected based on the maximum torque required by the structure and possible loads. In the current study, the material used for the structure was acrylic.

Figure 2.16 shows the force diagram used for load calculations. The calculations were carried out only for the joints that have the largest loads, since the other joints would have the same motor, i.e. the motor can move the links without problems. The calculations considered the weight of the motors, about 50 grams, except for the weight of motor at joint B, since it is carried out by link BA. Figure 2.17 shows the force diagram on link CB, which contains the joints (B and C) with the highest load (carry the links DC and ED) and the calculations are carried out as follows.

The values used for the torque calculations:

Wd = 0.020 kg (weight of link DE)

Wc = 0.030 kg (weight of link CD)

Wb = 0.030 kg (weight of link CB)

L = 1 kg (load)

Cm = Dm = 0.050 kg (weight of motor)

LBC = 0.14 m (length of link BC)

LCD = 0.14 m (length of link CD)

LDE = 0.05 m (length of link DE)

Performing the sum of forces in the Y axis, using the loads as shown in Figure 5.20, and solving for CY and CB, see Equations (1).(4). Similarly, performing the sum of moments around point C, Equation (5), and point B, Equation (6), to obtain the torque e in C and B, Equations (7) and (8), respectively.

$$\sum F_y = \left(L + W_d + D_m + W_c + C_m\right)\mathrm{g} - C_Y = 0 \qquad (1)$$

$$C_Y = \left(1.141\,\mathrm{kg}\right) 9.8\,\mathrm{m/s^2} = 11.18\,\mathrm{N} \qquad (2)$$

$$\sum F_y = \left(L + W_d + D_m + W_c + C_m + W_B\right) g - C_B = 0 \qquad (3)$$

$$C_B = \left(1.171\,\mathrm{kg}\right) 9.8\,\mathrm{m/s^2} = 11.4758\,\mathrm{N} \qquad (4)$$

$$\sum M_c = -\left(\frac{W_c L_{CD}}{2}\right) - W_D\left(L_{CD} + \frac{L_{DE}}{2}\right) - L\left(L_{CD} + L_{DE}\right) - D_m\left(L_{CD}\right) + M_c = 0 \qquad (5)$$

$$\sum M_B = -L\left(L_{BC} + L_{CD} + L_{DE}\right) - W_D\left(L_{BC} + L_{CD} + \frac{L_{DE}}{2}\right) - D_m\left(L_{BC} + L_{CD}\right) - W_c\left(L_{BC} + \frac{L_{CD}}{2}\right) - C_m\left(L_{BC}\right) - W_B\left(\frac{L_{BC}}{2}\right) + M_B = 0 \qquad (6)$$

$$M_c = 1.968\,\mathrm{Nm} = 278.6\,\mathrm{oz/in} \qquad (7)$$

$$M_B = 3.554\,\mathrm{Nm} = 503.38\,\mathrm{oz/in} \qquad (8)$$

The servo motor that was selected, based on the calculations, is 7 small servo with torque: 1.8 kg/cm and one big servo with torque 10kg/cm

. This motors was recommended because it is much cheaper than any other motor with same specifications. Since we need more torque at joint B, see Equation (8), we used two motors at point B to comply with the torque requirements; however, one motor is enough for the other joints. Using two motors at joint B is much cheaper than using one big motor with 10kg/cm. Other relevant characteristics of the motors, which can be shown in Figureure 5, are that they can turn 60 degrees in 130 milliseconds and they have a weight of 47.9 grams each. Once the initial dimensions for the robot arm and the motor

were defined, the design were carried out using the SolidWorks platform; design should

Figure 5.5 Two Type of Servo Motor

carefully take into account the thickness of the acrylic sheet and the way that the pieces would be attached to each other. The acrylic sheet used to make the robot is 1/8 thickness

And that thin sheet was chosen because it easier for machining and less weight with a good resistance. During design, we faced some difficulties due to the way of joining thin acrylic parts strongly. It was needed tools to burn and join the acrylic parts and that weren't available and the team considered that a mechanical junction based on screws and nuts would be much strong than other alternatives, such as glue for example. In order to accomplish this, a small feature was designed which allowed to fasten the

bolts with the nuts without having to screw in the thin acrylic layer. The result of this process was the tridimensional design shown in Figure 6.21 By end of design, each part was printed in full scale in cardboard paper and then we verified all the dimensions and the interfaces of the assembly. In turn, we built the first prototype of the robot arm. Next, parts of the robot arm were machined from the acrylic sheet using a circular saw and Dermal tools. The detailing on the parts was done in a professional workshop since the parts of robot arm were too small and it is not an easy for accomplishing such small and accurate cuts. During assembling the robot parts with the motors, few problems pop up. There were critical points that did not resist the fastening and, in turn, may break down; hence, reinforcements in these points were considered. The final result of the robot arm is shown in Figure 5.21

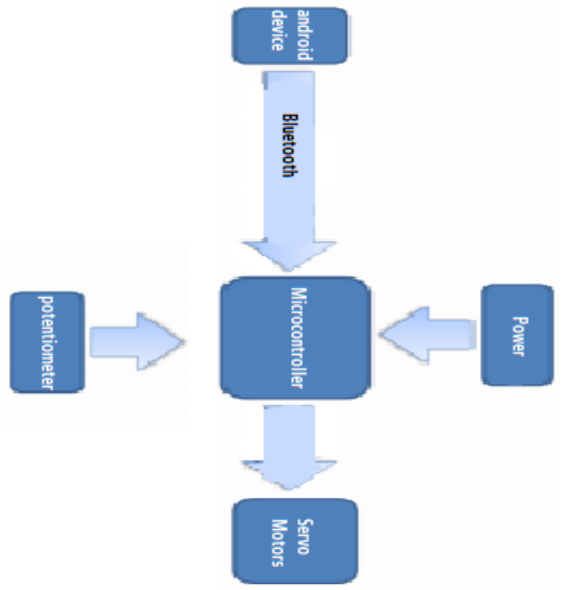
5.2 Driving engines using complementary angle

We use this Operation to control tow servo motors In opposite direction for each other's. The small servo G9 we use have a rotation limit 180 degree so we command the first servo motor to move to angle X and command the opposite servo motor to move to angle 180.X then the tow motors well move in Synchronous movement that meets our needs.

5.3 Robot Arm Control

The robot arms can be autonomous or controlled manually. In manual mode, a trained operator (programmer) typically uses a portable control device (a teach pendant) to teach a robot to do its task manually. Robot speeds during these programming sessions are slow. In the current work we enclosed the both modes. The control for the presented robot arm consists basically of three levels: a microcontroller, a driver, and a computer based user interface. This system has unique characteristics that allow flexibility in programming and controlling method, which was implemented using inverse kinematics; besides it could also be implemented in a full manual mode. The electronic design of control is shown in Figure 11. The microcontroller used is an Atmega 368p which comes with a development/programming board named “Arduino”, as shown in Figure 12. The programming language is very similar to C but includes several libraries that help in the

control of the I/O ports, timers, and serial communication. This microcontroller was chosen because it has a low price, it is very easy to reprogram, the programming language is simple, and interrupts are available for this particular chip. The driver used is a eight .channel for servo controller board. It supports tow control methods: Bluetooth for direct connection to an android device or direct control using variable resistors. This

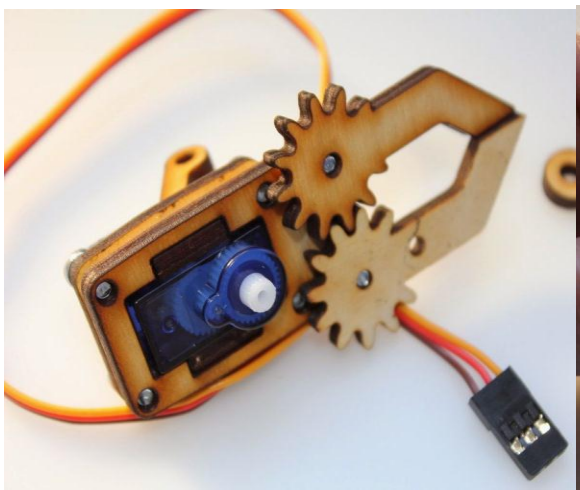


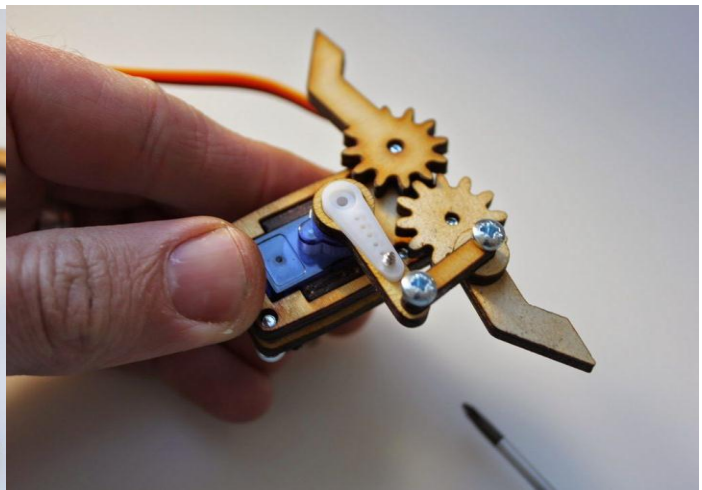
controller, as shown in Figure 5.6.

Figure 5.6 Electronic scheme of control

5.4 End-Effector Selection

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 robot arm accomplishes for; it can be pneumatic, electric or hydraulic. Since our robot arm is based on an electric system, we may choose electric basis of end effector. Besides, the main application of our system is handling, accordingly, the recommended type of our end effector is a gripper, as shown in Figurer 5.22 and Figurer 5.23. Please note that the end effector is controlled by a servo motor .

Figure 5.7 gripper Closed , gripper Open

5.5 MIT App Inventor

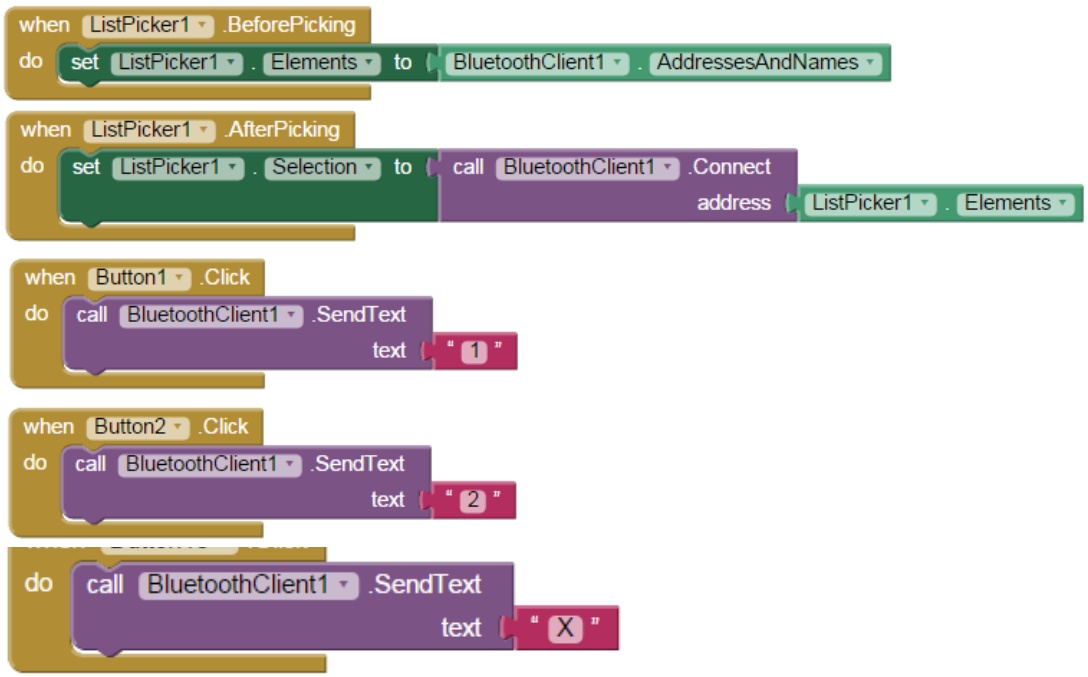
MIT App Inventor is a blocks based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices. Newcomers to App Inventor can have their first app up and running in an hour or less, and can program more complex apps in significantly less time than with more traditional, text based languages. Google's Mark Friedman and MIT Professor Hal Abelson co-led the development of App Inventor while Hal was on sabbatical at Google. Other early Google engineer contributors were Sharon Perl, Liz Looney, and Ellen Spertus . App Inventor runs as a Web service administered by staff at MIT's Center for Mobile Learning. A collaboration of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) and the MIT Media Lab. MIT App Inventor supports a worldwide community of nearly 3 million users representing 195 countries worldwide. The tool's more than 100 thousand active weekly users have built more than 7 million android apps. An open source tool that seeks to make both programming and app creation accessible to a wide range of audiences

Figure 5.8 MIB Structure

- **Formal and informal educators** who have used MIT App Inventor to introduce programming to their Computer Science students, science club members, after school programs attendees, and summer campers. Many educators have also started to use MIT App Inventor to develop apps in support of their own instructional objectives.
- **Government and civic employees and volunteers** who have harnessed the power of MIT App Inventor to develop custom, often hyper local apps in response to natural disasters and community based needs Designers and product managers who have seen the potential that MIT App Inventor has to support the iterative design process via rapid prototyping, testing and iteration.
- **Researchers** who use MIT App Inventor to create custom app in support to meet their data collection and analysis requirements in support of their research in a wide variety of fields from medical to social.
- **Hobbyists and Entrepreneurs** who have an idea they want to quickly turn into an app without the cost or learning curve that more traditional app creation entails.

5.6 Robotic arm app using MIT app inventor

We use the app inventor to create an app on android device to control on robotic arm using android device and Bluetooth module



5.6.1 Interface designer :

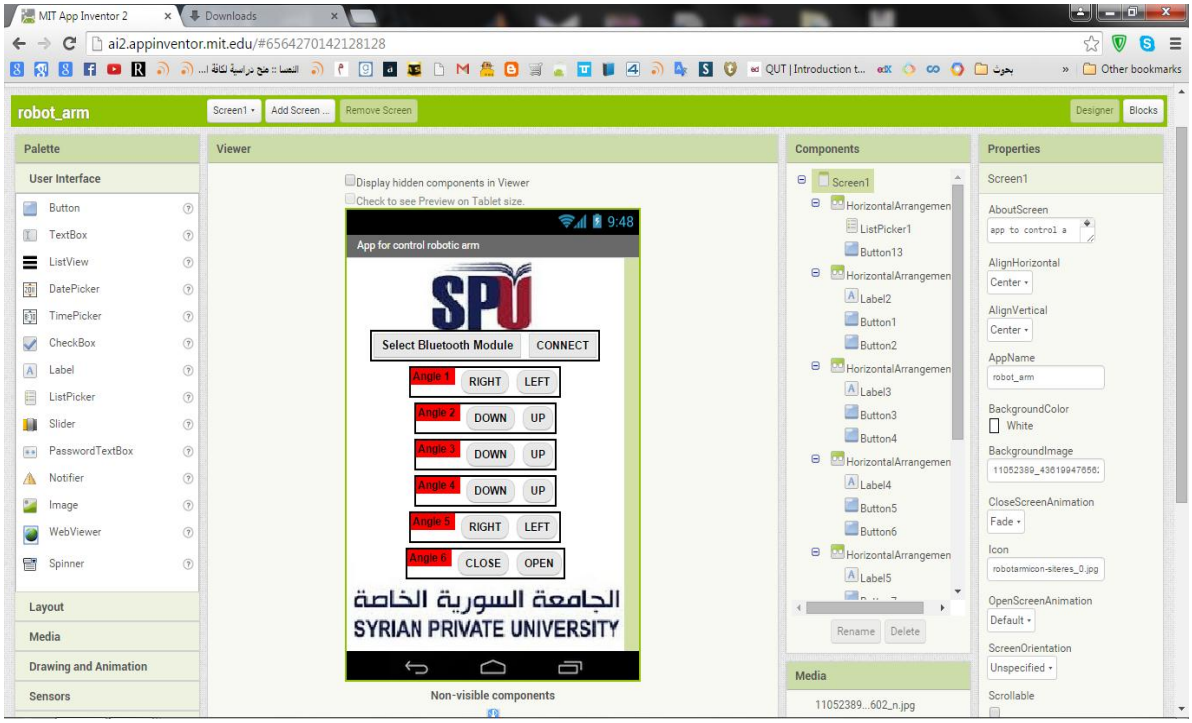


Figure5.12 Interface designer

5.6.2 Interface Blocks:

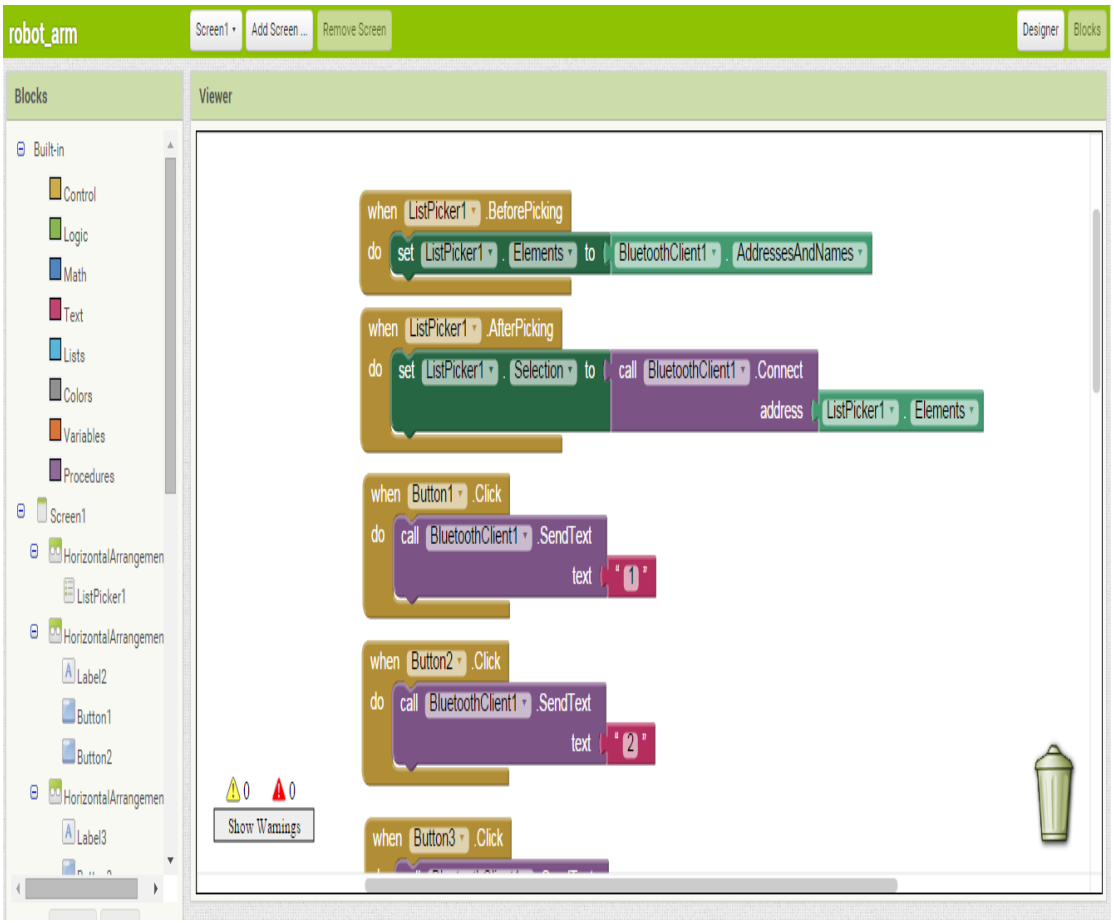


Figure5.13 Interface Blocks

5.6.3 App Designer:

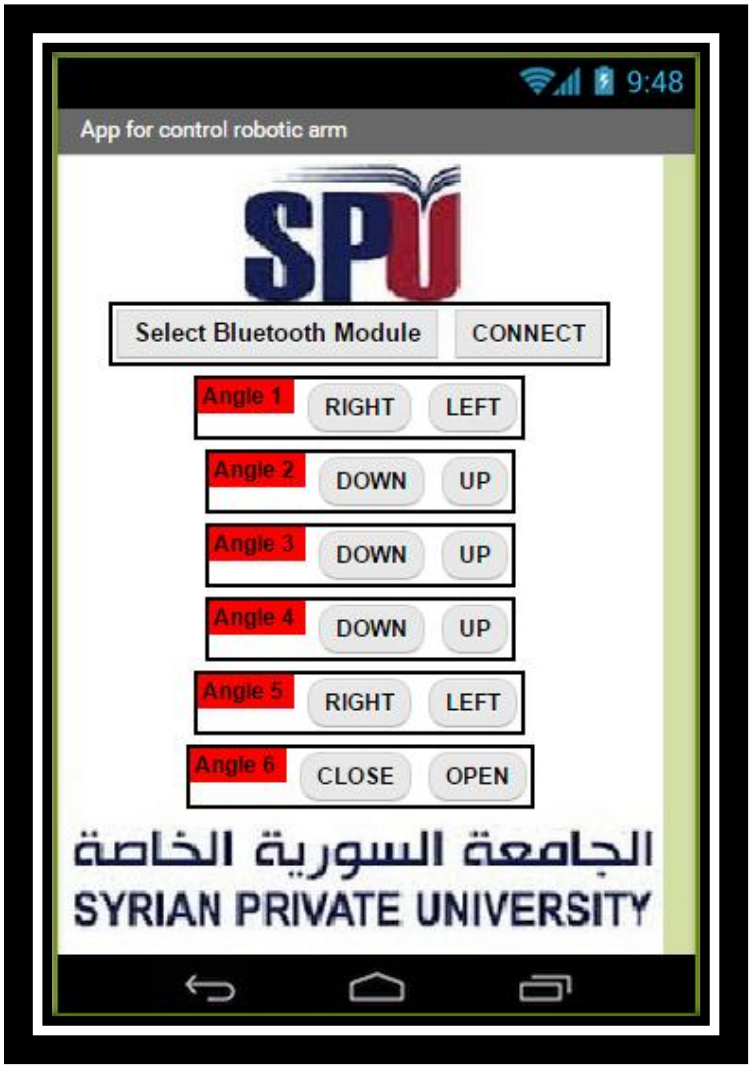


Figure 5.14 App Designer

5.6.4 Arm Block Diagram:

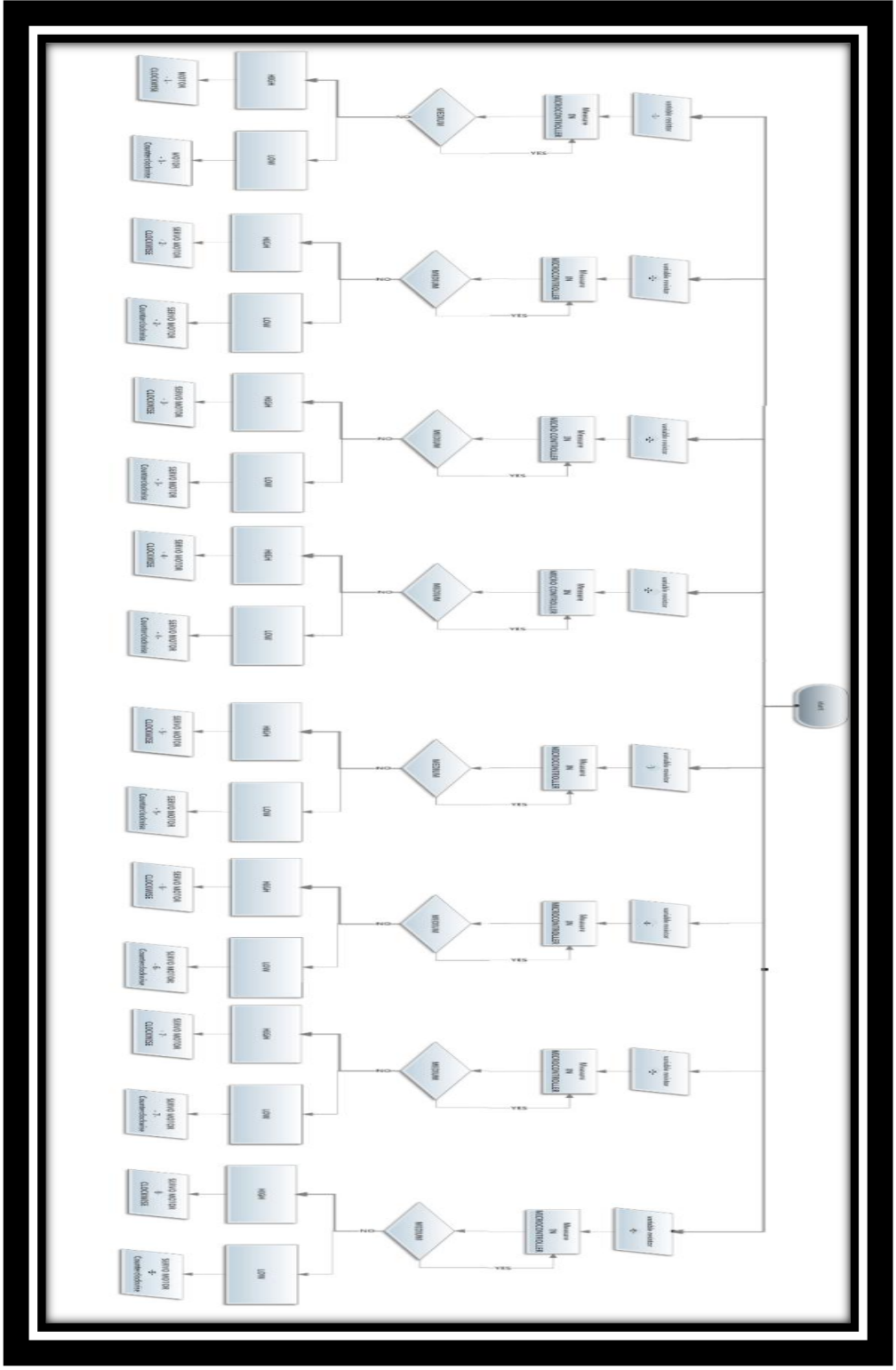


Figure 5.15 Arm Block Diagram

CONCLUSION

The main focus of this work was to design, and programme robotic arm the robot arm was designed with five degrees of freedom and talented to accomplish accurately simple tasks, such as light material handling the robot arm is equipped with several servo

motors which do links between arms and perform arm movements. A microcontroller that drives the servo motors with the capability of modifying position

The programming is done on ATMEGA-328p Microcontroller using Arduino programming. The potentiometers are also used to detect the angle of rotation and the signals are then sent to the microcontroller. And you can control the robotic arm also using android device, in today's world, this Robotic arm has turned out very benevolent. Besides Robotics and Automation, these kinds of arms have applications in other fields also.

REFERENCE

[1] C. F. Olson, Lab. JP, Technol. CIo, Pasadena, “Probabilistic self-localization for mobile robots”; IEEE Transactions on Robotics and Automation 16,1, 55-66, 2000.

[2] R.B. Gillespie, J. E. Colgate, M. A. Peshkin, “A general framework for robot control”; IEEE Transactions on Robotics and Automation, 17,4, 391-401, 2001. [3] Devendra P. Garg and Manish Kumar, “Optimization Techniques applied to multiple manipulators for path planning and torque minimization”; Engineering Applications of Artificial Intelligence 15, 3-4, 241-252, 2002.

[4] J. C. Trinkle and R. James Milgram, “Complete Path Planning for Closed Kinematics Chains with Spherical Joints”; SAGE International Journal of Robotic Research 21, 9, 773- 789, 2002.

[5] M. Gemeinder and M. Gerke, “GA-based Path Planning for Mobile Robot Systems employing an active Search Algorithm”; Journal of Applied Soft Computing 3, 2, 149-158, 2003. [2]. “Six-servo Robot Arm” from www.arexx.com.cn on 13-11-2011 pg: 1-18

[6]. Jegede Olawale, Awodele Oludele, Ajayi Ayodele, “Development of a Microcontroller Based Robotic Arm”, in Proceedings of the 2007 Computer Science and IT Education Conference pg: 549-557.

[7]. “Six-servo Robot Arm” from www.arexx.com.cn on 13- 11-2011.

[8]. “Robotic Arm” - www. NASA explores. com

[9]. “Robot software” from Wikipedia, the free encyclopedia on 04-3-2012

[10]. “ATMEL – short notes”, www.atmel.com

[11]. “Arduino MCU”, - www.arduino.com

[12].“Servomotors & system design components”, - www.bhashaelectronics. Com

[13] Robert J. Shilling "Fundamentals of robotics analysis and control ".

[14] GORDON McCOMB" The robot builders' bonanza ".

[15] www.wikipedia.com/robotic.

[16] G. J. Monkman" Robot Gripper ".

[17] http://www.instructables.com/

[18] J. M. Selig "Introductory Robotics"

[19] Andrease Wolf "Grippers in Motion"

[20] www.SocietyofRobots.com

Appendix A

```
#include <Servo.h>

#include <SoftwareSerial.h>

 SoftwareSerial Bluetooth(10, 11);

 char Serial_Data;

 Servo myservo0;

 Servo myservo1;  // create servo object to control a servo

 Servo myservo2;// twelve servo objects can be created on most boards

 Servo myservo3;

 Servo myservo4;

 Servo myservo5;

 Servo myservo6;

 Servo myservo7;

 const int sensor1=A0;

 const int sensor2=A1;

 const int sensor3=A2;

 const int sensor4=A3;

 const int sensor5=A4;

 const int sensor6=A5;

int  pos0 = 180;     // variable to store the servo position
```

```
int  pos1=90;    // variable to store the servo position
int  pos2=90;    // variable to store the servo position
int  pos3=90;    // variable to store the servo position
int  pos4=90;    // variable to store the servo position
int  pos5=90;    // variable to store the servo position
int  pos6=90;    // variable to store the servo position
int  pos7=90;    // variable to store the servo position
void setup()
{
 Bluetooth.begin(9600);
  myservo0.attach(2);
  myservo1.attach(3);
  myservo2.attach(4);
  myservo3.attach(5);
  myservo4.attach(6);
  myservo5.attach(7);
  myservo6.attach(8);
  myservo7.attach(9);// attaches the servo on pin 9 to the servo object
  ////////////////////////////////////////////////////////////////////


 }
```

```
void loop() {


 pos0=analogRead(sensor1);
pos0=map(pos0,0,1023,0,180);
myservo0.write(pos0);
pos1=analogRead(sensor2);
pos1=map(pos1,0,1023,70,110);
pos2=analogRead(sensor2);
pos2=180-map(pos2,0,1023,70,110);
myservo1.write(pos1);
myservo2.write(pos2);
pos3=analogRead(sensor3);
pos3=map(pos3,0,1023,0,180);
pos4=analogRead(sensor3);
pos4=180-map(pos4,0,1023,0,180);
myservo3.write(pos3);
myservo4.write(pos4);
pos5=analogRead(sensor4);
pos5=map(pos5,0,1023,0,180);
myservo5.write(pos5);
```

```
pos6=analogRead(sensor5);
pos6=map(pos6,0,1023,0,180);
myservo6.write(pos6);
pos7=analogRead(sensor6);
pos7=map(pos7,0,1023,90,180);
myservo7.write(pos7);

if (Bluetooth.available()){
    Serial_Data=Bluetooth.read();


 if (Serial_Data=='X'){
  for (;;){
        Serial_Data=Bluetooth.read();

////////////////////////////////////////////////
    if (Serial_Data=='1'){
   if (pos0<177){
    pos0=4+pos0;
   }
  myservo0.write(pos0);
```

```
  Serial_Data=0;
    }

    if (Serial_Data=='2'){
    if(pos0>3){
    pos0=pos0-4;
    }
  myservo0.write(pos0);
  Serial_Data=0;
    }
////////////////////////////////////////////
     if (Serial_Data=='3'){
      if (pos1<109){
pos1=pos1+2;
pos2=180-pos1;
      }
  myservo1.write(pos1);
  myservo2.write(pos2);

  Serial_Data=0;
  }
```

```
    if (Serial_Data=='4'){


 if (pos1>71){
 pos1=pos1-2;
pos2=180-pos1;
 }
  myservo1.write(pos1);
  myservo2.write(pos2);
  Serial_Data=0;
      }
     ////////////////////////////////////////
    if (Serial_Data=='5'){
    if(pos3<175){
        pos3=pos3+6;
pos4=180-pos3;
    }
  myservo3.write(pos3);
  myservo4.write(pos4);
  Serial_Data=0;
          }
```

```
    if (Serial_Data=='6'){
  if(pos3>5){
        pos3=pos3-6;
pos4=180-pos3;
  }
  myservo3.write(pos3);
  myservo4.write(pos4);
  Serial_Data=0;
          }
          ////////////////////////////////
     if (Serial_Data=='7'){
   if(pos5>5){
   pos5=pos5-6;
   }
  myservo5.write(pos5);
  Serial_Data=0;
   }

   if (Serial_Data=='8'){
  if (pos5<175){
   pos5=pos5+6;
```

```
  }
  myservo5.write(pos5);
  Serial_Data=0;
  }
  ////////////////////////////////////////////
   if (Serial_Data=='9'){
  if(pos6<175){
   pos6=pos6+6;
  }
  myservo6.write(pos6);
  Serial_Data=0;
   }

   if (Serial_Data=='A'){
if(pos6>5){
   pos6=pos6-6;
}
  myservo6.write(pos6);
  Serial_Data=0;
   }
   ////////////////////////////////////////
```

```
  if (Serial_Data=='B'){
 if (pos7<175){
  pos7=pos7+6;
 }
myservo7.write(pos7);
Serial_Data=0;
 }
 if (Serial_Data=='C'){
if (pos7>95){
  pos7=pos7-6;
}
myservo7.write(pos7);
  }
}}}}}
```