Robotic Arm

A Project Presented to
the Faculty of Engineering
STI College Rosario

As a requirement for COE Elective 2
Mechatronics and Robotics

By

Dimanalata, Lenmor L.

Dones, Agustin Ace, P.

Legaspi, Jon Jon, R.

Mojillo, Chona E.

Paderna, Romeo III, B.

Portuguez, Jan Paul R.

Salatandre, Walter S.

Tolentino, Armeo A.

Mr. Paolo Roberto O. Lozada
Adviser

March 2012

I. INTRODUCTION

The term robot comes from the Czech word *robota*, generally translated as "forced labor." This describes the majority of robots fairly well. Most robots in the world are designed for heavy, repetitive manufacturing work. They handle tasks that are difficult, dangerous or boring to human beings.

The most common manufacturing robot is the robotic arm. A typical robotic arm is made up of seven metal segments, joined by six joints. The computer controls the robot by rotating individual step motors connected to each joint (some larger arms use hydraulics or pneumatics). Unlike ordinary motors, step motors move in exact increments. This allows the computer to move the arm very precisely, repeating exactly the same movement over and over again. The robot uses motion sensors to make sure it moves just the right amount.

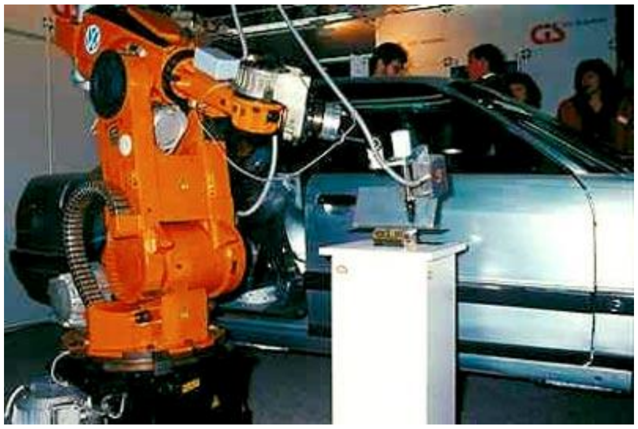


Figure 1.1 **Robotic arm used in car manufacturing**

An industrial robot with six joints closely resembles a human arm -- it has the equivalent of a shoulder, an elbow and a wrist. Typically, the shoulder is mounted to a stationary base structure rather than to a movable body. This type of robot has six *degrees of freedom*, meaning it can pivot in six different ways. A human arm, by comparison, has seven degrees of freedom.

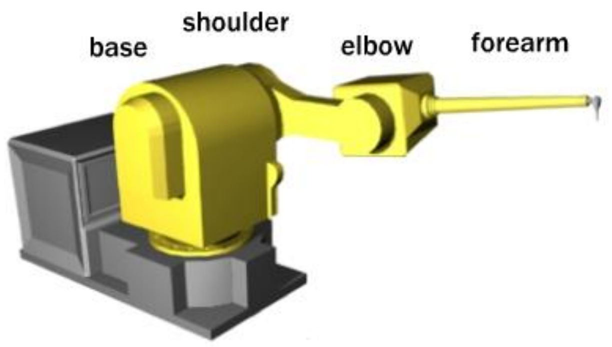


Figure 1.2 **Common parts of an arm**

Your arm's job is to move your hand from place to place. Similarly, the robotic arm's job is to move an end effector from place to place. You can outfit robotic arms with all sorts of end effectors, which are suited to a particular application. One common end effector is a simplified version of the hand, which can grasp and carry different objects. Robotic hands often have built-in pressure sensors that tell the computer how hard the robot is gripping a particular object. This keeps the robot from dropping or breaking whatever it's carrying. Other end effectors include blowtorches, drills and spray painters.

Industrial robots 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 robot how to do its job, the programmer guides the arm through the motions using a handheld controller. The robot stores the exact sequence of movements in its memory, and does it again and again every time a new unit comes down the assembly line.

Most industrial robots work in auto assembly lines, putting cars together. Robots can do a lot of this work more efficiently than human beings because they are so precise. They always drill in the exactly the same place, and they always tighten bolts with the same amount of force, no matter how many hours they've been working. Manufacturing robots are also very important in the computer industry. It takes an incredibly precise hand to put together a tiny microchip. [HARR2012]

Figure 1.2 **Assembly line robotic arms**

1.1 Statement of the Problem

1.1.1 General Problem

How to design and develop a robotic arm that can perform or simulate a task?

1.1.2 Specific Problems

1. How to design and develop the articulated jointed arm and base that will hold the end effector?

   The arm consisting of several motors, must hold the end effector effectively and must be able to be balance itself correctly in any movement.

2. How to design and develop the end effector that can do a special task?

   The end effector performs tasks, such as cleaning and getting objects.

3. How to design and develop a circuit that can control the robotic arm's movement?

   An electronic circuit is needed to provide power and control to the robotic arm.

4. How to create a microcontroller program that can efficiently utilize the robotic arm?

A program controls the motors by sending pulses. It must also be able to synchronize input to the behavior of the robotic arm.

1.1.3 Project Rationale

This project would be beneficial to:

- Students

Students from different fields, particularly those taking up engineering and information technology courses, can study and observe the project prototype and documentation to learn more about robotics and mechatronics. This can also be a basis for their projects.

- Engineering Instructors

Instructors can use the project for demonstration and simulation of industrial robots.

1.2 Current State of Technology

Robotic arms have been extensively used in the industry today. They can be found in medicine, space exploration, and commonly in manufacturing and production firms.

Canadarm2

Launched on STS-100 (assembly flight 6A) in April 2001, the next generation Canadarm is a bigger, better, smarter version of the space shuttle's robotic arm. It is 17.6 meters (57.7 feet) long when fully extended and has seven motorized joints. This arm is capable of handling large payloads and assisting with docking the space shuttle. The Space Station Remote Manipulator System, or SSRMS, is self-relocatable with a Latching End Effector, so it can be attached to complementary ports spread throughout the station's exterior surfaces.[KAUD2010]

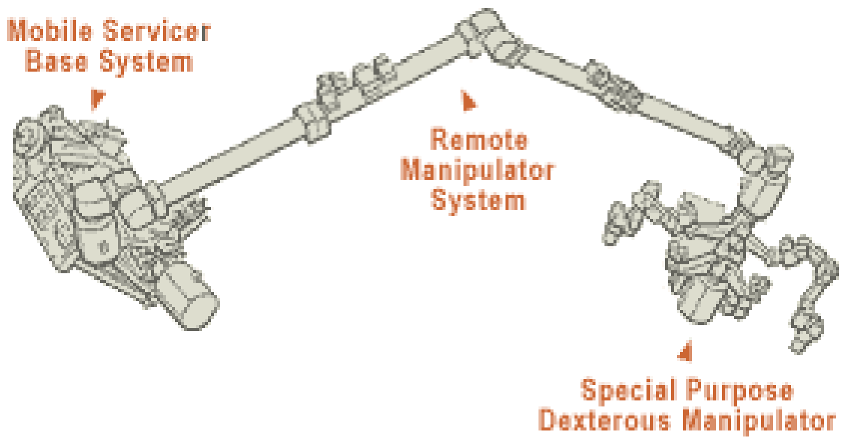


Figure 1.3 **NASA Canadarm2**

Motoman UP6

The Motoman UP6 is a high-speed robot that is compact and requires minimal installation space. Offering superior performance for welding and non-welding applications, its thin base and arm design allows for close placement for piece holding fixtures to improve part accessibility.

The UP6 features collision avoidance that allows for two or three robots to work in conjunction with each other for increased productivity. The UP6 offers the widest work envelope in its class with a 1,373mm reach.

The UP6 is also available as a shelf-mount, the UP6R robot. Shelf-mounting allows for larger reach and more applications. [ROBO2012a]

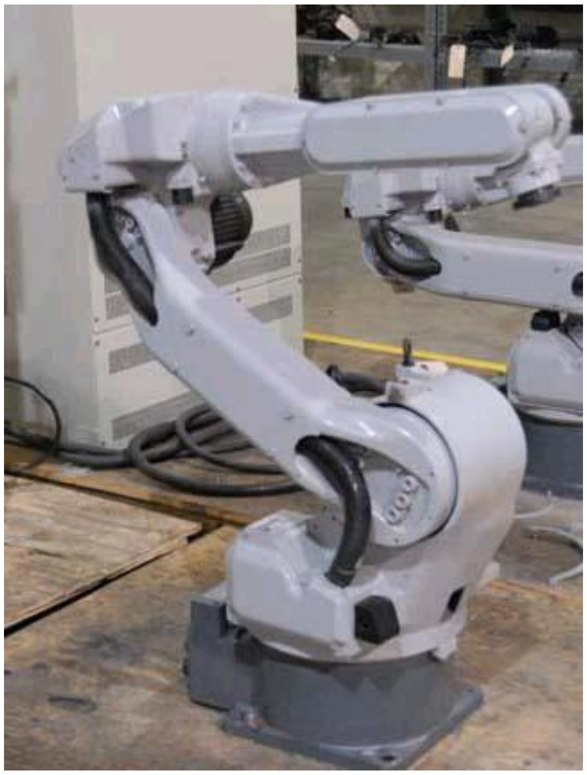


Figure 1.4 **Motoman UP6**

Fanuc ArcMate 120i Robot

The flexible Fanuc ArcMate 120i is a six-axis, electric servo-driven robot capable of precise high-speed welding and cutting. Accuracy and performance are designed into the ArcMate 120i for all your welding and cutting requirements.

Designed with integral utilities including gas/air lines and a wire feed motor cable routed inside the robot arm, the ArcMate 120i offers improved reliability, reduces setup time and eliminates external cabling requirements.

Features:

Compatible with all major brands of welding equipment

Interfaces with most types of servo driven or indexing positioners

Sealed bearings and drives provide protection and improve reliability

Integrated controller reduces footprint and eliminates exposed connection cables

Cable routing through center of axis rotation improves reliability

[ROBO2012b]

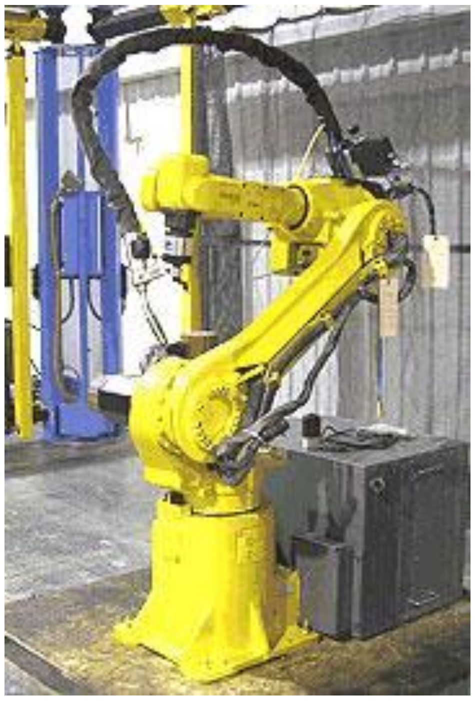


Figure 1.5 **Fanuc ArcMate 120i Robot**

Luke

Robotic arms are also used by amputees. One of the latest advancements is “Luke” by Dean Kamen, which is an impressive, mind-controlled prosthetic robot arm he’s invented at D6 in Carlsbad. “Luke” is an incredibly sophisticated bit of engineering that’s lightyears ahead of the clamping "claws" that many amputees are forced to use today. The arm is fully articulated, giving the user the same degrees of movement as a natural arm, and is sensitive enough to pick up a piece of paper, a wineglass or even a grape without mishap. [TWEN2008]

Figure 1.6 **Luke prosthetic robot arm**

1.3 Objectives

1.3.1 General Objective

This project aims to design and develop a robotic arm that can perform or simulate a task.

1.3.2 Specific Objectives

Specifically, this study aims to:

1. Design and develop the articulated jointed arm and base that will hold the end effector.

   The arm utilizes three servo motors to hold the end effector. The motors are connected by brackets.

2. Design and develop a cleaning tool as the end effector.

   The arm utilizes a DC motor as a cleaning tool.

3. Design and develop a circuit that can control the robotic arm's movement.

   A circuit, with an MCU as its main chip, contains pushbuttons that can control the movement of each of the motor bracket clockwise and counter clockwise.

4. How to create a microcontroller program that can efficiently utilize the robotic arm?

   The MCU program efficiently sends pulses to the servo motor for every movement. The program is written and compiled using PICBASIC PRO.

1.4 Scope and Limitations

Scope

1. The project demonstrates controlling of mechanical devices, such as servo motors and DC motor
2. The robotic arm provides control for the user through the push buttons
3. The robotic arm simulates some industrial and commercial tasks, such as cleaning a car, shining and polishing a car, hooking light objects
4. The project employs the use of a PIC MCU that can be reprogrammed. The button functions and motor movement can be changed by reprogramming the MCU.

Limitations

1. The power of the robotic arm to lift is limited to the strength of the motors. The robotic arm may not carry heavy objects and may cause it to topple down.
2. The robotic arm depends on a continuous 12V DC voltage with high current capability, such as a 220VAC – 12VDC power adapter.
3. The servo motor used is limited to a maximum of 180 degrees and is recommended to run it less than 180 to increase the motor’s life.
4. The elevation and reach of the robotic arm is limited by the length of its brackets.

2.0 THEORETICAL FRAMEWORK

2.1 Introduction

This chapter discusses the theories and concepts employed in the course of designing and constructing the Robotic Arm.

2.2 Degrees of Freedom (DOF) and The Robot Arm Free Body Diagram (FBD)

The degrees of freedom, or DOF, is a very important term in robotics. Each degree of freedom is a joint on the arm, a place where it can bend or rotate or translate. You can typically identify the number of degrees of freedom by the number of actuators on the robot arm. When building a robot arm you want as few degrees of freedom allowed for your application, because each degree requires a motor, often an encoder, and exponentially complicated algorithms and cost.

The Denavit-Hartenberg (DH) Convention is the accepted method of drawing robot arms in FBD's. There are only two motions a joint could make: translate and rotate. There are only three axes this could happen on: x, y, and z (out of plane). Shown below are a few robot arms with their FBDs, to demonstrate the DOF relationships and symbols. The DOF on the gripper (otherwise known as the end effector) is not counted. The gripper is often complex with multiple DOF, so for simplicity it is treated as separate in basic robot arm design.

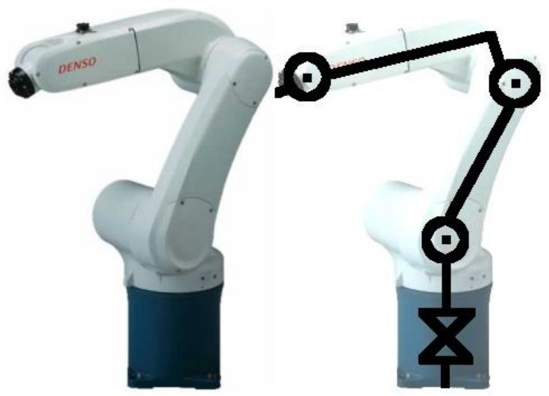


Figure 2.1 **4 DOF Robot Arm, three are out of plane**

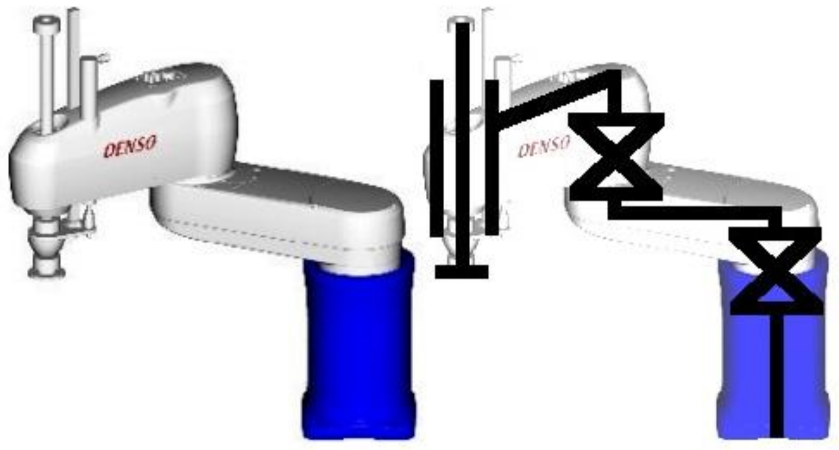


Figure 2.2 **3 DOF Robot Arm, with a translation joint**

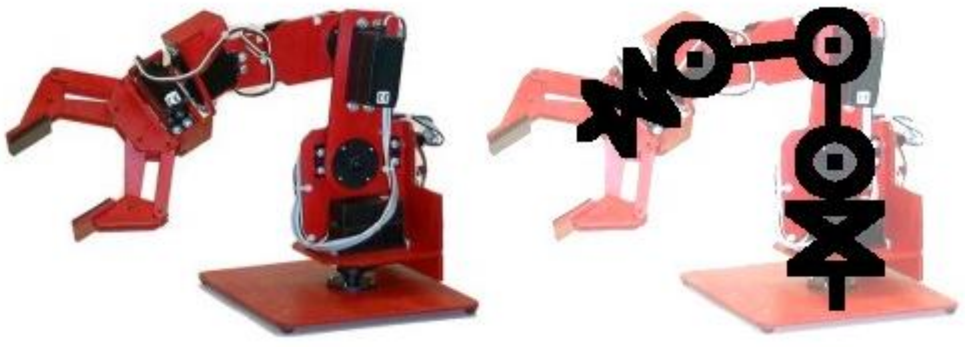


Figure 2.3 **5 DOF Robot Arm**

Notice between each DOF there is a linkage of some particular length. Sometimes a joint can have multiple DOF in the same location. An example would be the human shoulder. The shoulder actually has three coincident DOF. If you were to mathematically represent this, you would just say link length = 0.

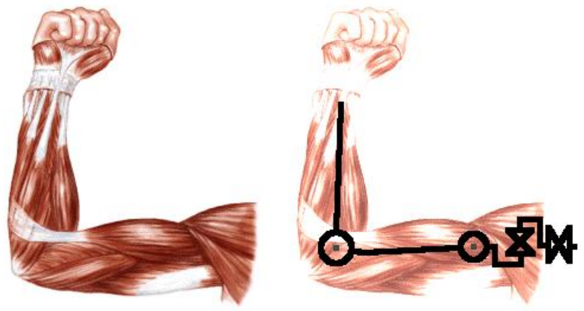


Figure 2.3 **3 DOF Human Arm**

Also note that a DOF has its limitations, known as the configuration space. Not all joints can swivel 360 degrees. A joint has some max angle restriction. For example, no human joint can rotate more than about 200 degrees. Limitations could be from wire wrapping, actuator capabilities, servo max angle, etc.

2.3 Robot Workspace

The robot workspace (sometimes known as reachable space) is all places that the end effector (gripper) can reach. The workspace is dependent on the DOF angle/translation limitations, the arm link lengths, the angle at which something must be picked up at, etc. The workspace is highly dependent on the robot configuration.

2.4 Force Calculations of Joints

The point of doing force calculations is for motor selection. You must make sure that the motor you choose can not only support the weight of the robot arm, but also what the robot arm will carry.

The first step is to label your FBD, with the robot arm stretched out to its maximum length. Choose these parameters:

- weight of each linkage
- weight of each joint
- weight of object to lift
- length of each linkage

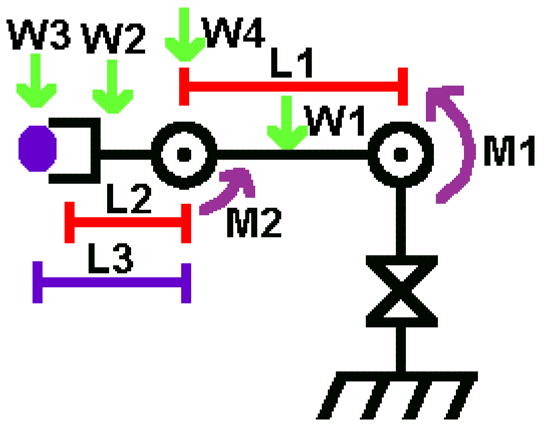


Figure 2.4 **3 DOF FBD with labels**

Next you do a moment arm calculation, multiplying downward force times the linkage lengths. This calculation must be done for each lifting actuator. This particular design has just two DOF that requires lifting, and the center of mass of each linkage is assumed to be Length/2.

Torque About Joint 1:

$M1 = L1/2 * W1 + L1 * W4 + (L1 + L2/2) * W2 + (L1 + L3) * W3$

Torque About Joint 2:

$M2 = L2/2 * W2 + L3 * W3$

[SOCI2012]

2.5 Servo Motors and Pulse Width Modulation

Servomotors are geared dc motors with a positional feedback control that allows the shaft (rotor) to be rotated and positioned accurately. When a control signal is being fed to the servomotor, the servomotor's shaft rotates to the position specified by the control signal. The positioning control is a dynamic feedback loop, meaning that if you forcibly rotate the servomotor's shaft way from its control signal command position, the servomotor circuitry will read this as a position error and will increase its torque in an attempt to rotate the shaft back to its command position.

Hobby servomotor specifications usually state that the shaf can be positioned through a minimum range of 90° (±45°). In reality this range can be extended closer to 180° (±90°) by adjusting the position control signal. There are three wire leads to a hobby servomotor. Two leads are for power 15 V (red wire) and ground (black wire). The third lead (yellow or white wire) feeds a position control signal to the motor.

The position control signal is a single variable-width pulse. The pulse width typically varies between 1 and 2 ms. The width of the pulse controls the position of the servomotor shaft. Figure 12.26 illustrates the relationship of pulse width to servomotor position. A 1-ms pulse rotates the shaft to the extreme counterclockwise (CCW) position (-45°). A 1.5-ms pulse places the shaft in a neutral midpoint position (0°). A 2-ms pulse rotates the shaft to the extreme CW position (+45°). The pulse width signal is sent to the servomotor approximately 55 times per second (55 Hz). [IOVI2004]

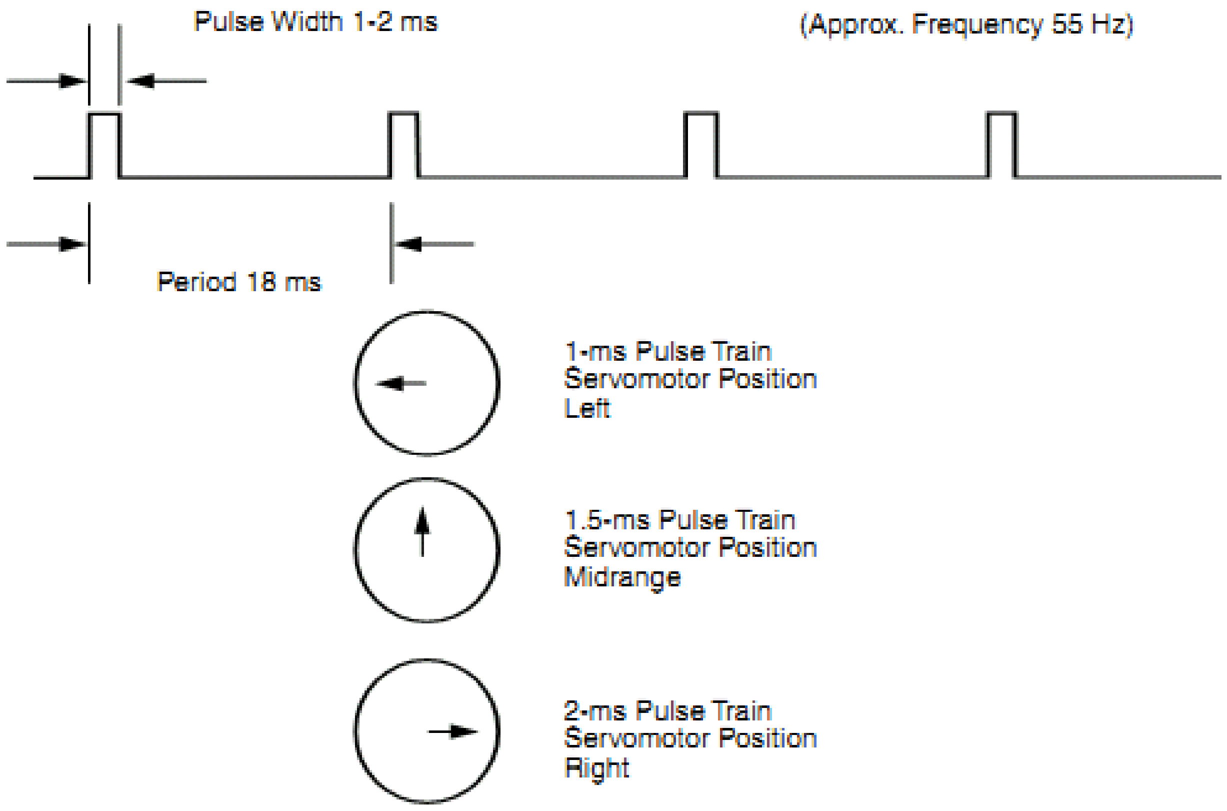


Figure 2.5 **Servomotor control signal diagram**

2.6 Microcontroller Units (MCU)

A microcontroller unit (MCU) is a single-chip computer. Micro suggests that the device is small, and controller suggests that it is used in control applications. Another term for microcontroller is embedded controller, since most of the microcontrollers are built into (or embedded in) the devices they control.

A microprocessor differs from a microcontroller in many ways. The main difference is that a microprocessor requires several other components for its operation, such as program memory and data memory, I/O interfaces, and external clock circuit. A microcontroller on the other hand has all the support chips incorporated inside the same chip. In essence, the microprocessor is only one component in the microcontroller chip.

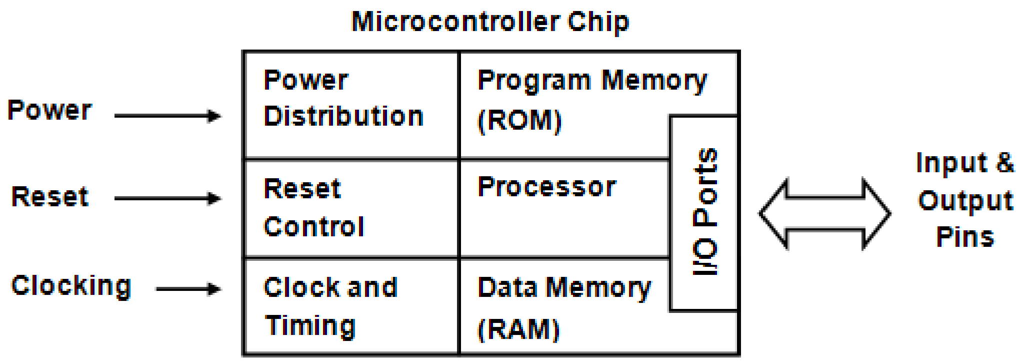


Figure 2.6 **Microcontroller Block Diagram**

The MCU is one of the most significant developments in the continuing miniaturization of electronic hardware. They are an important factor in the digitization of analog systems, such as sound systems or television. In addition, they provide an essential component of larger systems, such as automobiles, robots, and industrial systems. About 40% of microcontroller applications are in office automation, such as PCs, laser printers, fax machines, intelligent telephones, and so forth. About one-third of microcontrollers are found in consumer electronics goods. Products such as cell phones, calculators, MP3 players, hi-fi equipment, video games, washing machines and cookers fall into this category. The communications market, automotive market, and the military share the rest of the application areas. [IBRA2008]

3.0 ROBOTIC ARM

3.1 Introduction

This chapter includes the methodology, design, components, and processes involved in the development of Robotic Arm.

3.2 Project Design

3.2.1 Degrees of Freedom and Robot Workspace

The robotic arm FBD consist of 3 DOF, all rotational. The rotation of the ramp is not counted becuase it is separate from the arm. The DC motor, which serves as the end effector also cannot move in its place.

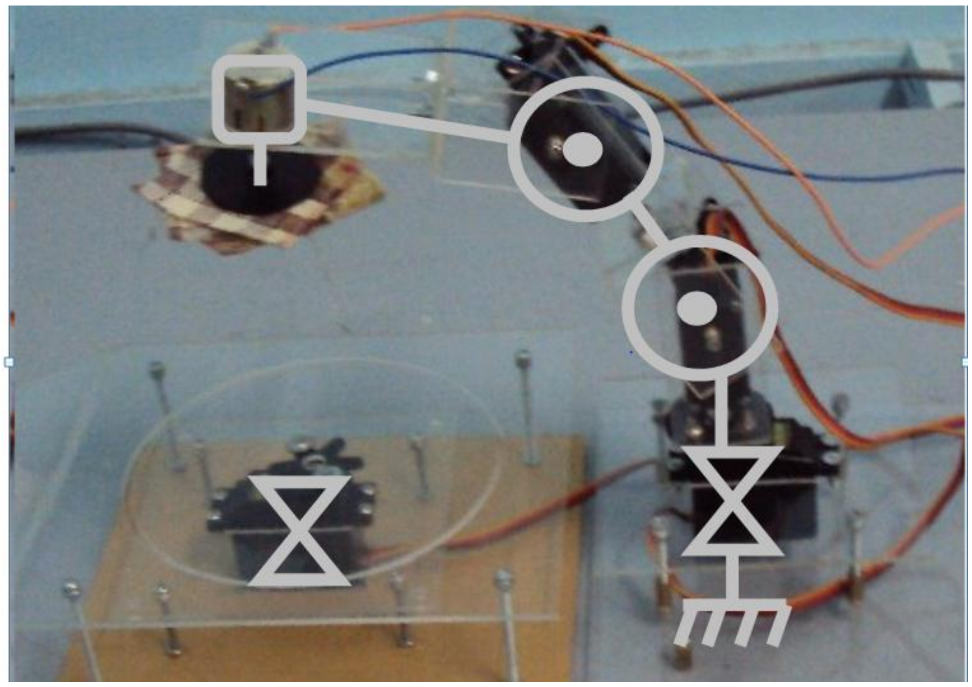


Figure 3.1 **Robotic Arm 3 DOF configuration**

Robot Workspace

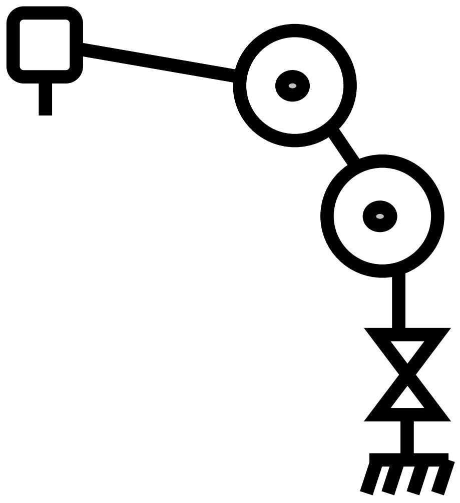


Figure 3.2 **Robotic Arm 3 DOF configuration**

The workspace is determined by tracing all the locations that the end effector can reach. The resulting image shows the extent of all joints each with a maximum of 180 degrees.

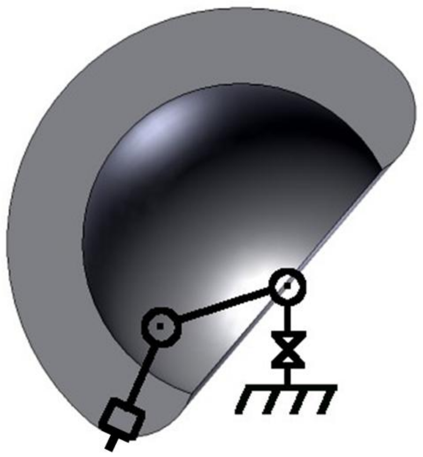


Figure 3.3 **Robotic Arm 3 DOF workspace**

Now rotating that by the base joint another 180 degrees to get 3D, we have this workspace image.. This creates a workspace of a shelled semi-sphere.

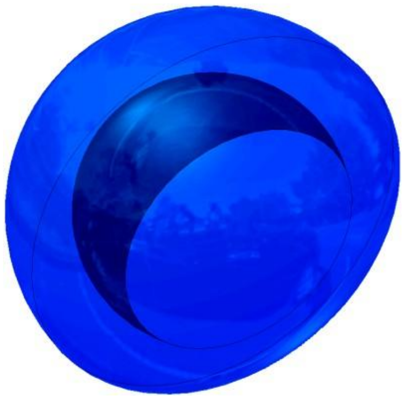


Figure 3.4 **Robotic Arm 3 DOF 3D workspace**

3.2.1 Force Calculations of Joints

Torque (T) is defined as a turning or twisting "force" and is calculated using the following relation:

T = F * L

The force (F) acts at a length (L) from a pivot point. In a vertical plane, the force acting on an object (causing it to fall) is the acceleration due to gravity ($g = 9.81\text{m/s}^2$) multiplied by its mass:

F = m* g

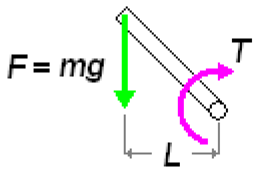
The force above is also considered the object's weight (W).

W = m* g

The torque required to hold a mass at a given distance from a pivot is therefore:

T = (m*g)*L

This can be found similarly by doing a torque balance about a point. Note that the length L is the PERPENDICULAR length from the pivot to the force.



$$\sum T = 0 = F * L - T$$

Therefore, replacing F with m*g, we find the same equation above. This method is the more accurate way to find torque (using a torque balance).

$$m * g * L = T_A$$

In order to estimate the torque required at each joint, we must choose the worst case scenario.

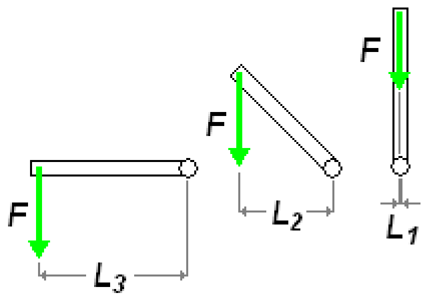


Figure 3.5 **Relationship of force and distance**

In the above image, a link of length L is rotated clockwise. Only the perpendicular component of length between the pivot and the force is taken into account. We observe that this distance decreases from L3 to L1 (L1 being zero). Since the equation for torque is length (or distance) multiplied by the force, the greatest value will be obtained using L3, since F does not change. You can similarly rotate the link counterclockwise and observe the same effect.

It can be safe to assume that the actuators in the arm will be subjected to the highest torque when the arm is stretched horizontally. Although your robot may never be designed to encounter this scenario, it should not fail under its own weight if stretched horizontally without a load.

The weight of the object (the "load") being held (**A1** in the diagram), multiplied by the distance between its center of mass and the pivot gives the torque required at the pivot. The tool takes into consideration that the links may have a significant weight (**W1, W2**..) and assumes its center of mass is located at roughly the center of its length. The torques caused by these different masses must be added:

$$T1 = L1 * A1 + \frac{1}{2} L1 * W1$$

Computation: T1 = (L1 * A1) + ((½ L1) * W1)

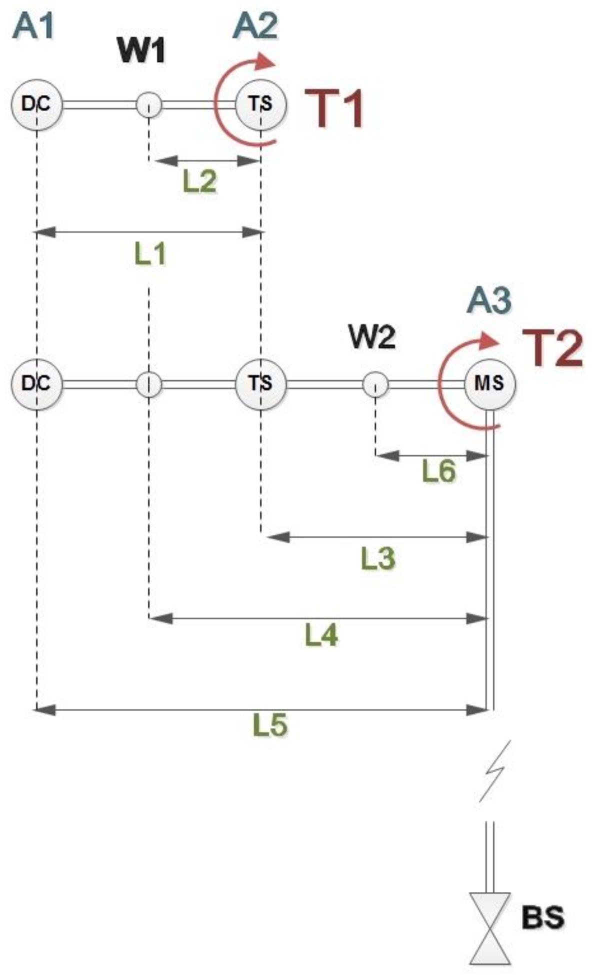
= (8cm * 0.028 kg) + ((1/2 * 8cm) * 0.02 kg)

\ = (0.224 kg.cm) + ((4 cm)* 0.02 kg)

= (0.224 kg.cm) + (0.08 kg.cm)

= 0.304 kg.cm

You may note that the actuator weight **A2** as shown in the diagram below is not included when calculating the torque at that point. This is because the length between its center of mass and the pivot point is zero. Similarly, when calculating the torque required by the actuator **A3**, its own mass is not considered. The torque required at the second joint must be re-calculated with new lengths, as shown below (applied torque shown in red):



Legend

A: weight of actuator (DC and servo motors)

T: applied torque

W: link weight (weight of brackets)

DC: DC motor actuator

TS: Top Servomotor actuator

MS: Middle Servomotor actuator

Weight Values
A1 = 28 g
A2 = 41 g
A3 = 56 g
W1 = 20 g
W2 = 20 g

Dimensions
L1 = 8 cm
L2 = 4 cm
L3 = 8 cm
L4 = 12 cm
L5 = 16 cm
L6 = 4 cm

Figure 3.6 **Robotic Arm torque analysis**

$$T2 = L5 * A1 + L4 * W1 + L3 * A2 + L6 * W2$$

Knowing that the link weight (W1, W2) are located in the center (middle) of the lengths, and the distance between actuators (L1 and L3 as in the diagram above) we re-write the equation as:

$$T2 = (L1 + L3) * A1 + (\tfrac{1}{2} L1 + L3) * W1 \\ + (L3) * A2 + (\tfrac{1}{2} L3) * W2$$

Computation: T2 = ((L1+L3) * A1) + ((1/2 L1 + L3) * W1) + ((L3) * A2) + ((1/2 L3) * W2)

= ((8cm + 8cm) * 0.028 kg) + ((1/2 * 8cm + 8cm) * 0.02 kg) + (8cm * 0.041 kg) + ((1/2 *8cm) * 0.02 kg)

= (16cm * 0.028 kg) + (12 cm * 0.02 kg) + (8cm * 0.041 kg) +

(4 cm * 0.02 kg)

= (0.448 kg.cm) + (0.24 kg.cm) + (0.328 kg.cm) + (0.08 kg.cm)

= 1.096 kg.cm

The tool only requires that the user enter the lengths of each link, which would be L1 and L3 above so the equation is shown accordingly. The torques at each subsequent joint can be found similarly, by re-calculating the lengths between each weight and each new pivot point.

The above equations only deal with the case where the robot arm is being held horizontally (not in motion). This is not necessarily the "worst case" scenario. For the arm to move from a rest position, an acceleration is required. The added torque is considered in advanced computations.

3.2.2 Dimensions

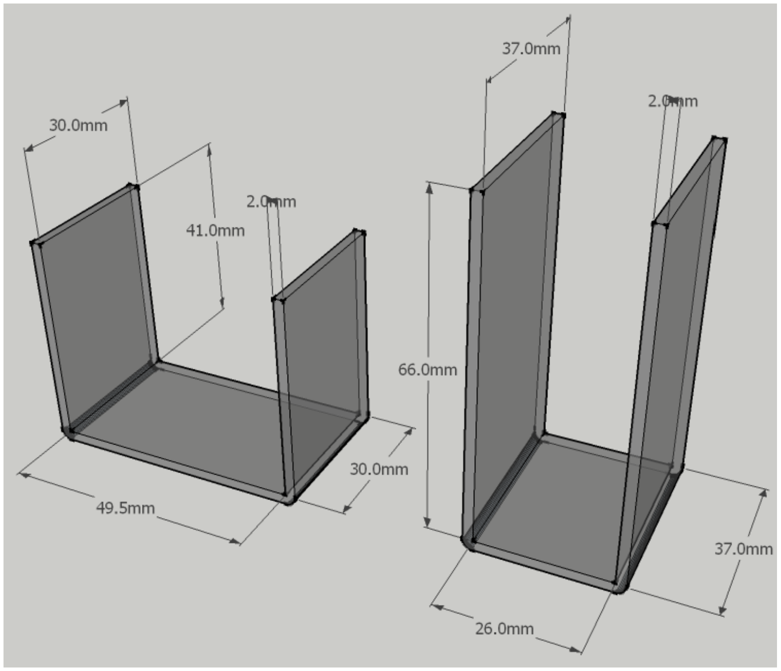


Figure 3.7 **Brackets A (right) and B (left) dimensions**

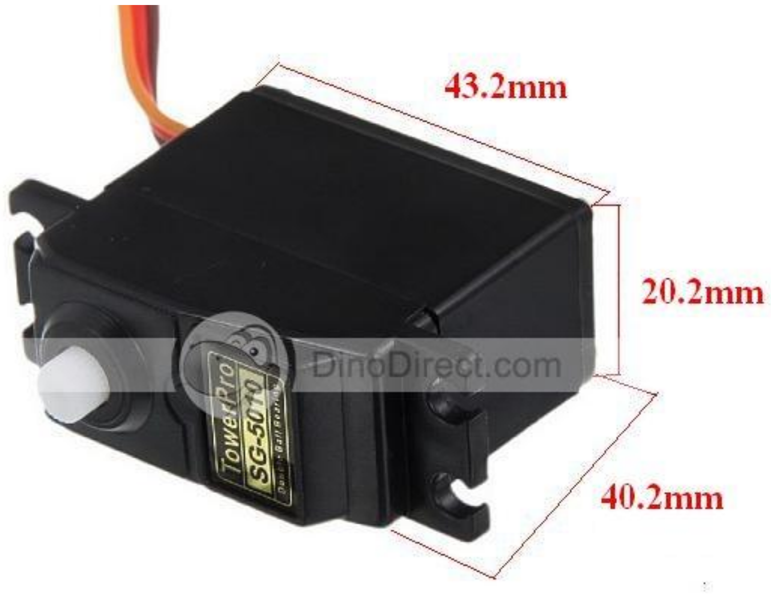


Figure 3.8 **Tower Pro SG-5010 Dimensions**

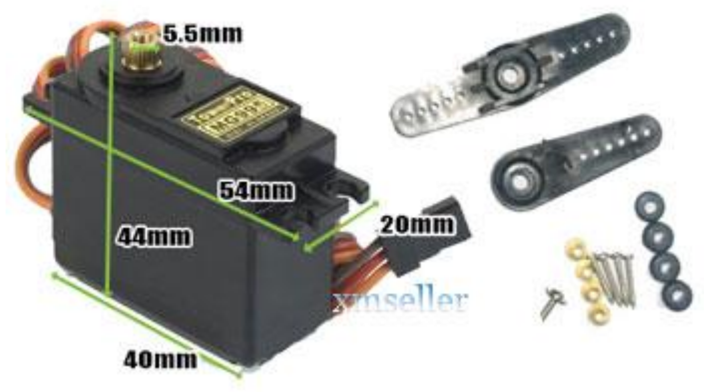


Figure 3.9 **Tower Pro MG-995 Dimensions**

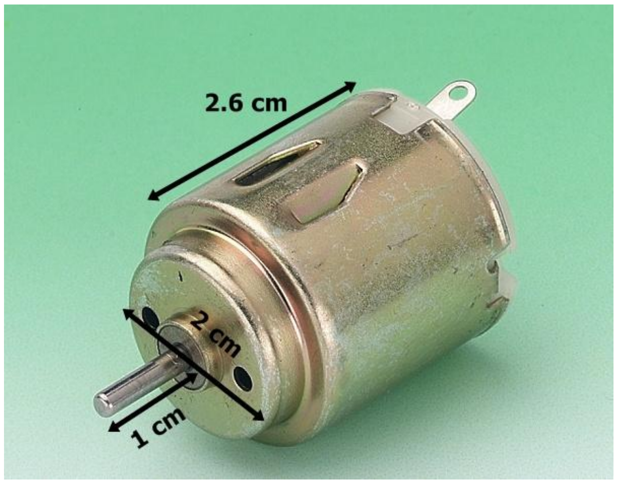


Figure 3.10 **DC Motor Dimensions**

3.2.3 Assembly

The servomotor bracket components are shown in Fig 3.11. Each of the fiber-glass U brackets that make up the assembly has multiple holes for connecting a servomotor horn as well as bottom and top holes for each servomotor bracket assembly consists of the following components: two fiber-glass U brackets, labeled A and B, pivot assembly consisting of 1.5mm x 8mm screw with nut and gold binding head, four 3mm x 6mm screws with nuts, and two sheet metal screws for mounting the servomotor horn.

When assembled with the servomotor (see Fig.3.12), the bracket becomes a modular component that may be attached to other brackets and components. The bracket allows the top and bottom components to swivel along the axis of the servomotor's shaft (see Fig. 3.13).

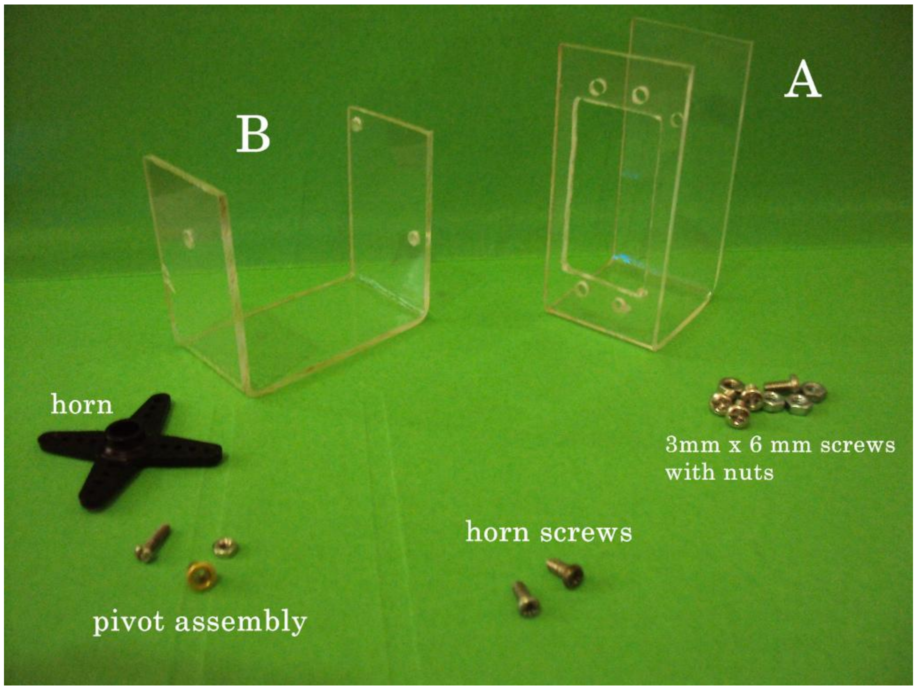


Figure 3.11 **Servomotor bracket kit**

The individual servomotor bracket kit must be assembled first. The servomotor is placed into bracket A by securing the four screws with nuts (Fig.3.15). The pivot assembly screw must also be fixed (Fig. 3.14) and the servo horn must be fixed into bracket B (Fig.3.16) using the horn screws. Brackets A and B must be joined as shown in Fig 3.17. With the design using multiple- servomotor assemblies, it is essential to preplan how the servomotors will be connected. When two or more servomotors assemblies are connected, the connecting brackets of the joints should be pre-assembled. The DC motor bracket is then fixed using the screws in Fig 3.18 to become the one shown in Figures 3.19 and 3.20. The assembled forearm is then fixed to the bottom rotating servomotor as shown in Fig. 3.21. The complete robotic arm is shown in Figures 3.22.

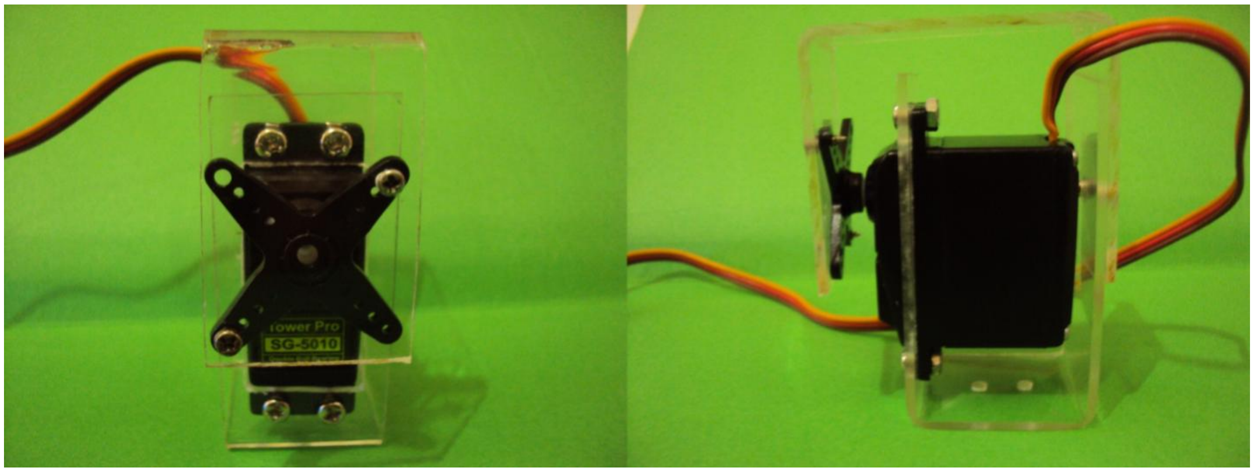


Figure 3.12 **Front and side views of servomotor bracket**

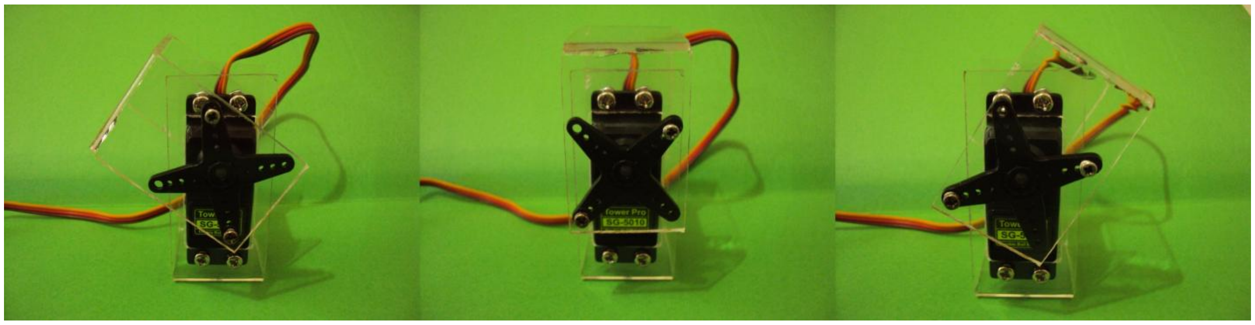


Figure 3.13 **Servomotor bracket travel (tilt left, center, tilt right)**

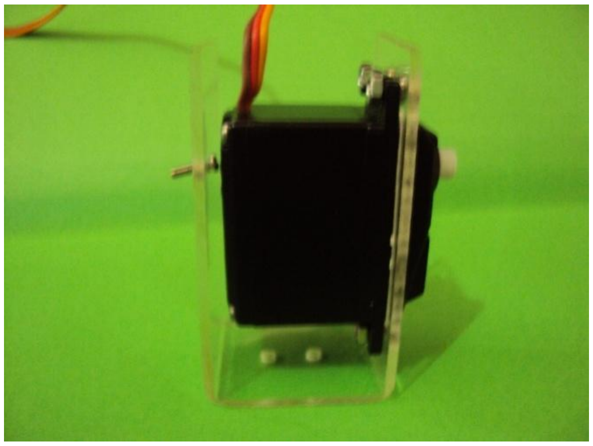


Figure 3.14 **Side view of placing servo in Bracket A**

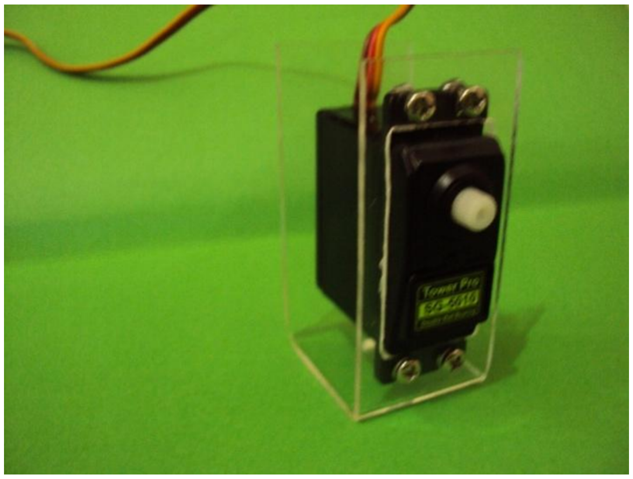


Figure 3.15 **A bracket with servo attached with screws and nuts**

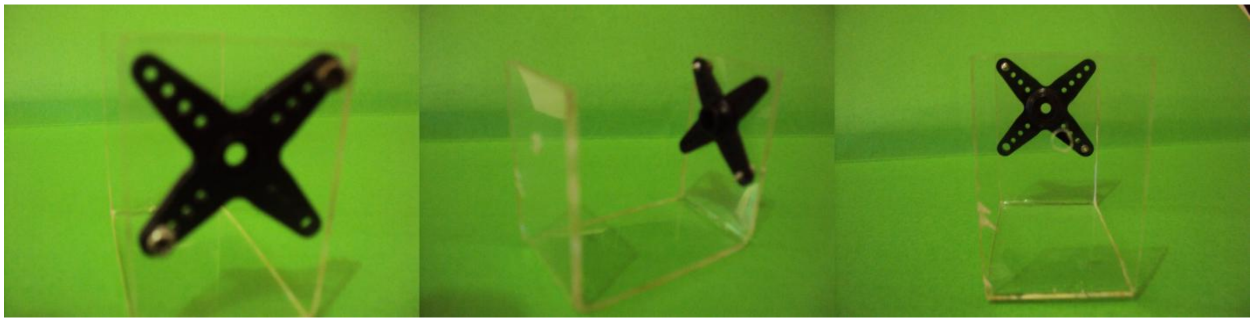


Figure 3.16 **Servomotor horn attached in bracket B**

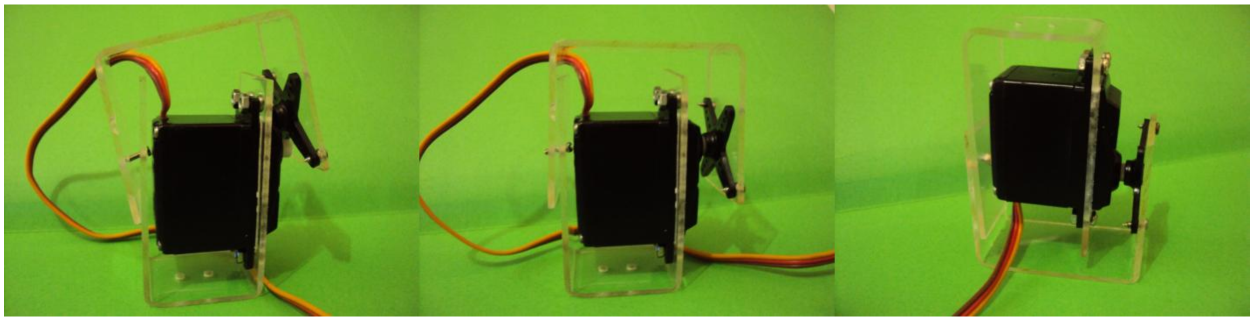


Figure 3.17 **Joining Bracket A and B, Horn assembly**

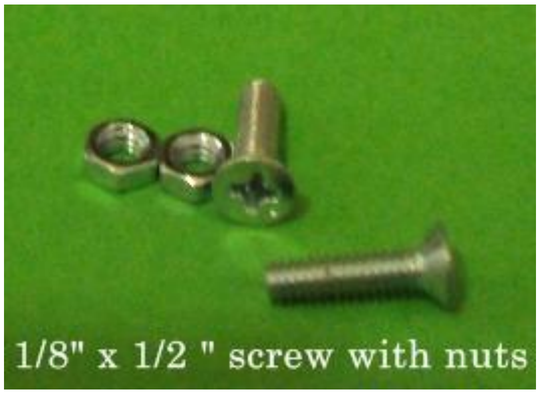


Figure 3.18 **Screws used in joining DC motor bracket to Bracket B of servo**

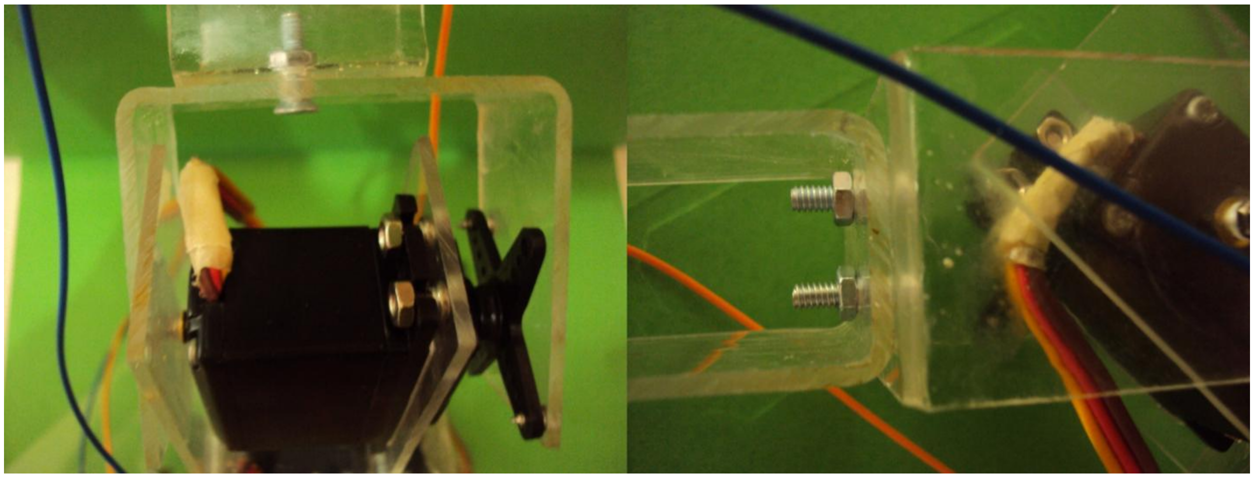


Figure 3.19 **DC motor bracket connected to Bracket B of servo**

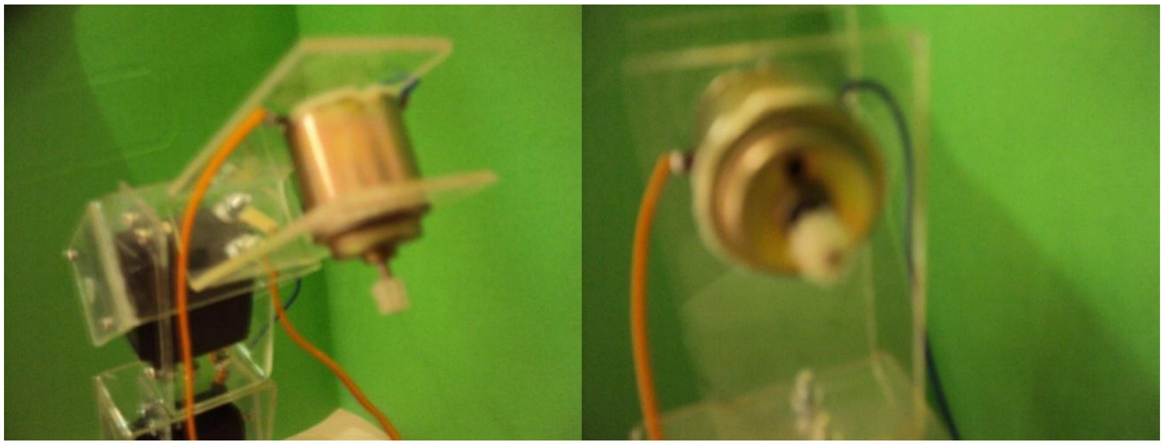


Figure 3.20 **DC motor bracket**

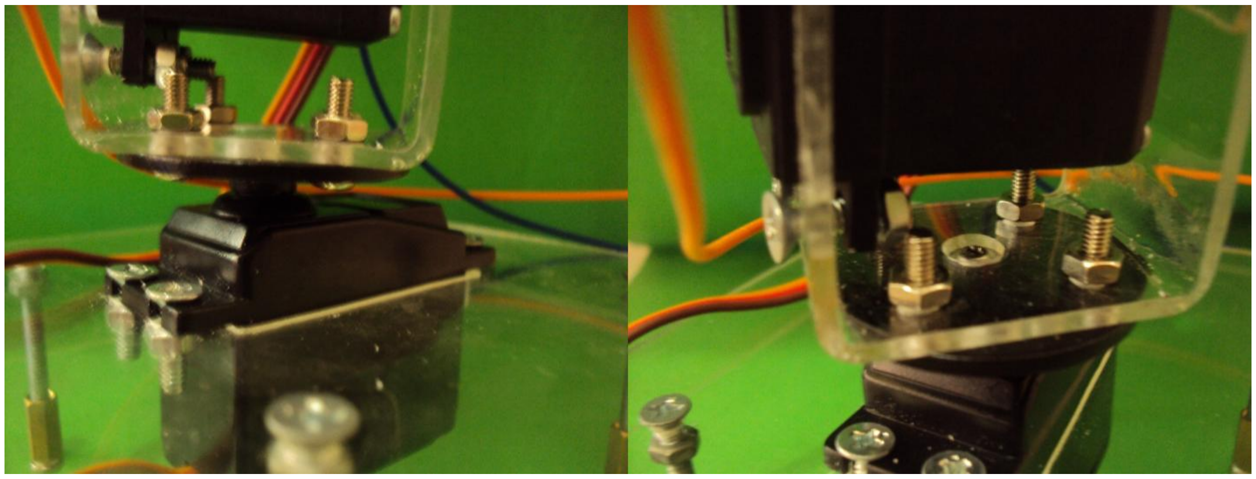


Figure 3.21 **Assembly of bottom servo to bracket A of middle servo**

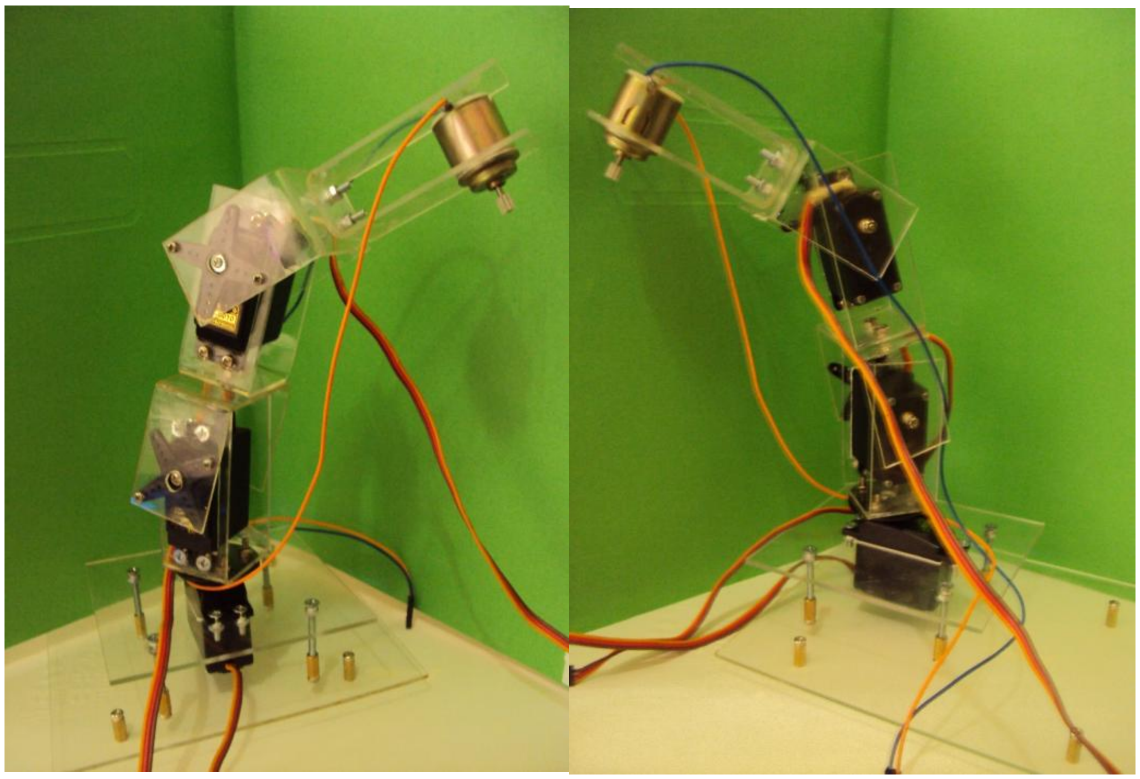


Figure 3.22 **4-motor Robotic Arm (3 servos, 1 DC motor)**

The ramp for the car consists of a rotating circular platform and an inclined plane. The servomotor used is similar to the top and middle servos. The complete set is shown in Fig 3.24.

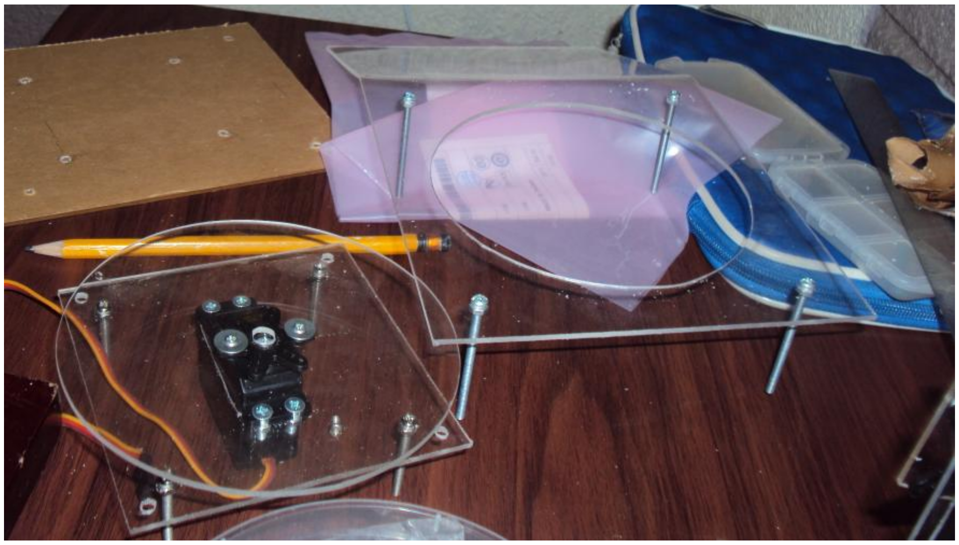


Figure 3.23 **Ramp with Rotating Platform and Inclined Plane**

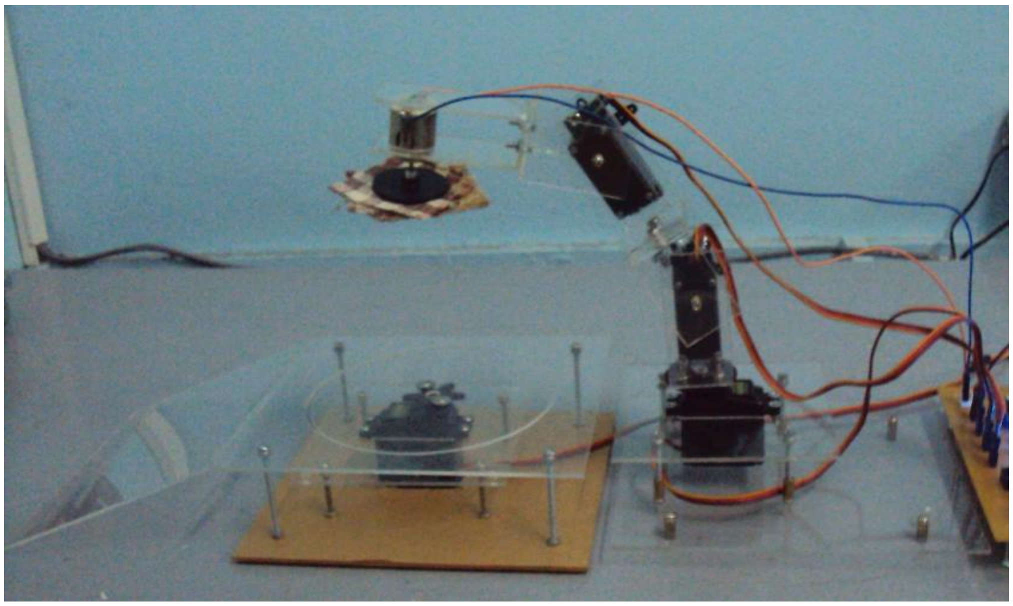


Figure 3.24 **Robotic Arm with Ramp**

3.2.4 Control Circuit

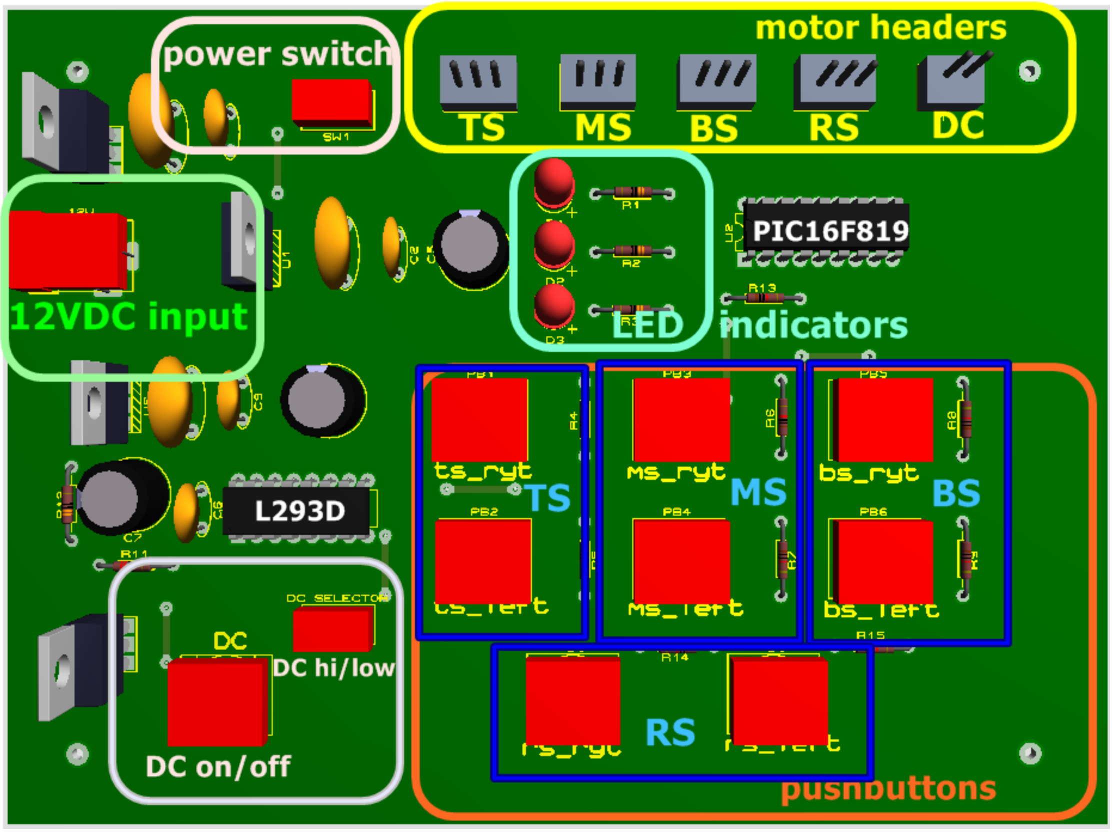


Figure 3.25 **Robotic Arm Control Circuit**

The circuit consists of the PIC MCU (PIC16F8190 which serves as the brain of the robot, pushbuttons which provides control to the user. There are also slide switches for on/off switching of the robotic arm and for rotation of the DC. There are also several headers where the servos and DC motor will be connected. The circuit accepts power through a 12VDC input jack. TS refers to Top Servo, MS refers to Middle Servo, BS refers to Bottom Servo, and RS refers to Ramp Servo. DC refers to DC motor.

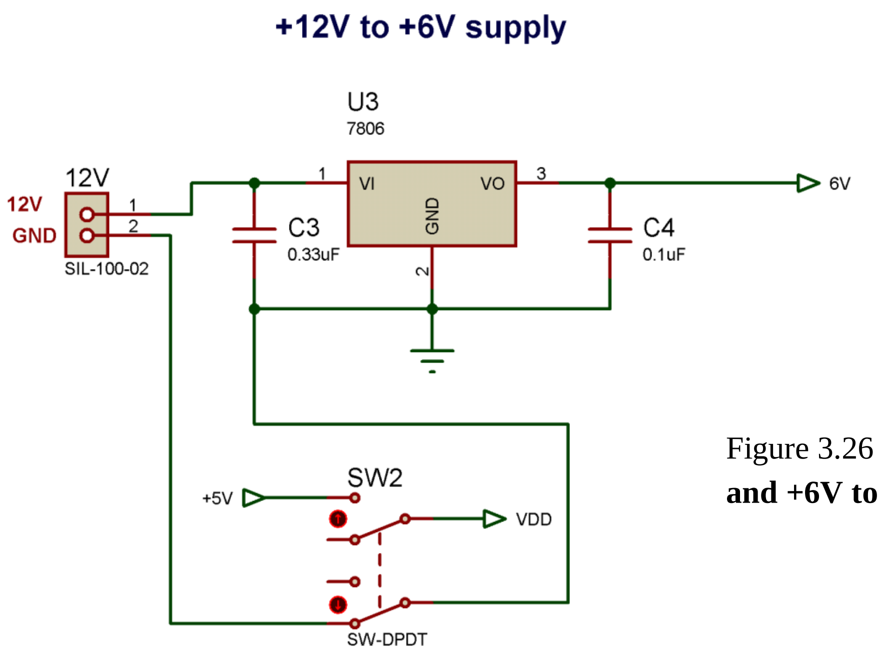
A guide on the preparation and operation of this circuit can be found in the User Manual.

Circuit Components

The project circuit includes several power supplies:

- +12V to +6V => 6V to power ServoMotors
  - Using 7806 voltage regulator
- +6V to +5V A => 5V for PIC MCU and pushbutton pull-ups
  - Using 7805 voltage regulator
- +6V to +5V B => 5V for DC Motor Driver L293D
  - Using 7805 voltage regulator
- +5V B to +3V => 3V for DC Motor
  - Using LM317 adjustable voltage regulator

An overall circuit switch, a SPDT (Single Pole Double Throw) slide switch is also added that shorts or breaks the VDD +5V A supply for the PIC MCU and the GND from the main power supply. The main power supply is a 220V AC to 3V-12V DC selectable supply.



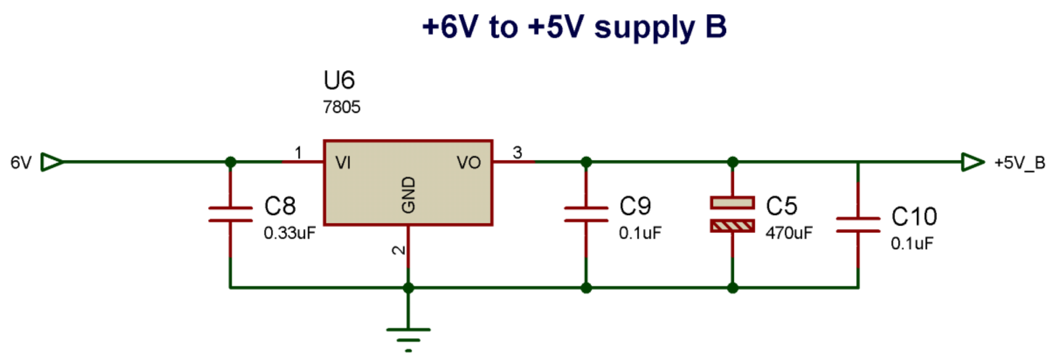
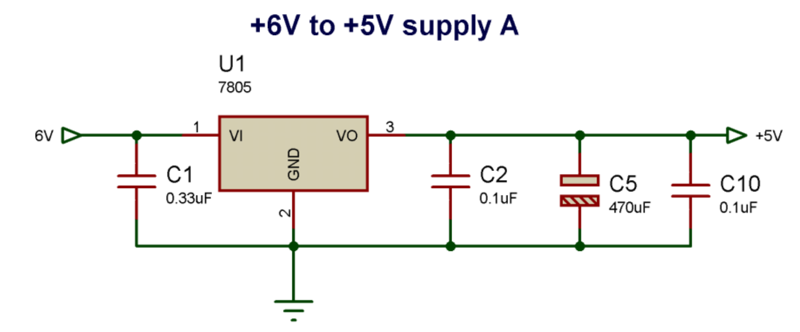


Figure 3.26 **+12V to +6V and +6V to +5V converter**



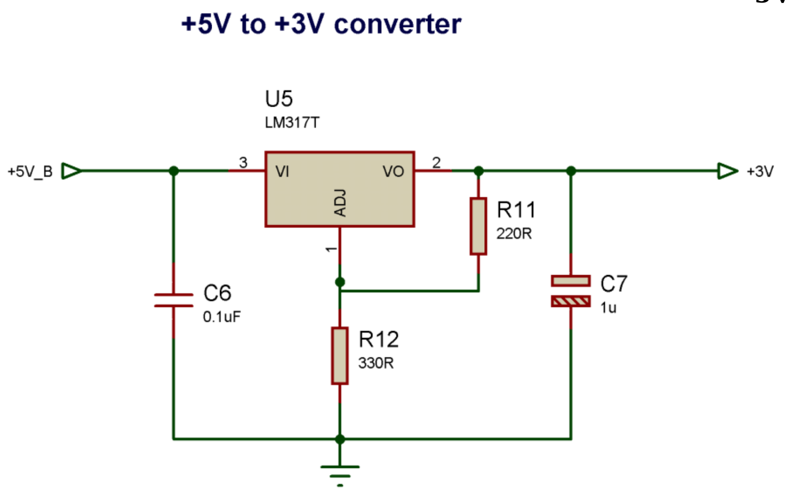


Figure 3.27 **+6V to +5V and +5V to +3V converter**

A separate 5V supply is provided for the DC Motor circuit to reduce the electrical noise produced by the DC Motor. The driver used is L293D quad H-bridge that handles rotation of the motor and reduces electrical noise. 5V is supplied to the IC and the 3V is tapped as input for the motor. 1 pin of the motor is pulled to GND already and the positive supply remains open until shorted out by the pushbutton.

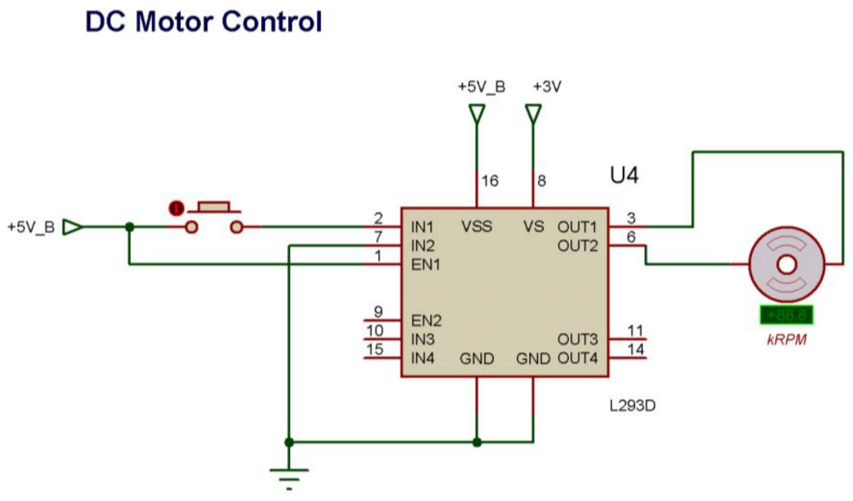


Figure 3.28 **DC Motor control**

The user control circuitry consists of 9 pushbuttons (SPST momentary Normally Open push button) which are pulled high through a 10k resistor. All of the servo switches are connected to the input pins of the PIC MCU.

- 1 switch for the momentary on/off of DC motor actuator (SPST momentary NO -normally open- push button)
- 2 switches for Clockwise and Counterclockwise rotation of Top Servo
- 2 switches for Clockwise and Counterclockwise rotation of Mid Servo
- 2 switches for Clockwise and Counterclockwise rotation of Bottom Servo
- 2 switches for Clockwise and Counterclockwise rotation of Ramp Servo

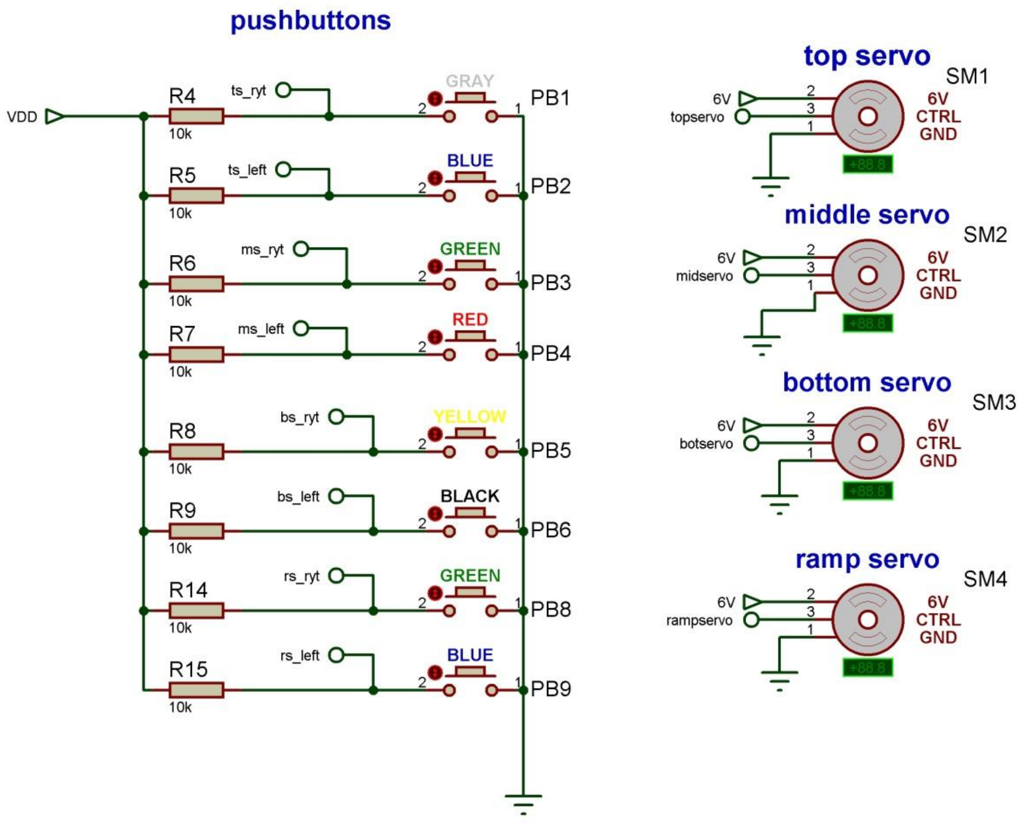


Figure 3.29 **Pushbuttons and Servo Motors**

The PIC MCU used is an 18-pin PIC MCU 16F819. It has a total of 15 I/O pins. 4 of the I/O pins outputs the pulse to the 3 servos. Three LEDs indicate which servo is activated.

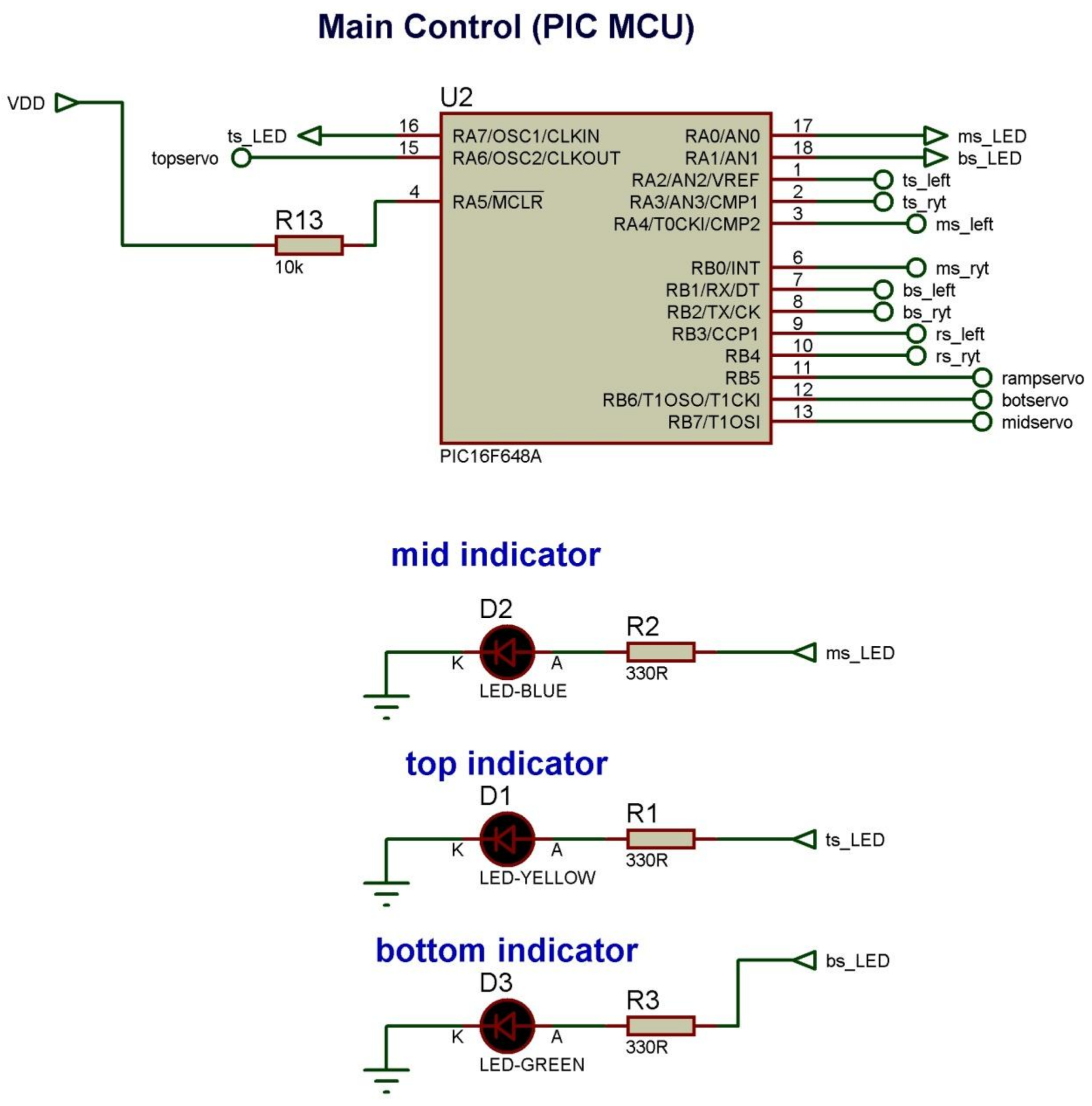


Figure 3.30 **PIC MCU and LED indicators**

3.2.5 Programming

The servomotor controllers use the PicBasic Pro **pulsout** command. The command format is as follows:

pulsout *pin, period*

The pulsout command generates a pulse on the pin specified for the period of time specified. The time is in 10-us (microsecond) increments. So to send a 1.5- ms pulse out on port B pin 0, you could use one of the following command for the PicBasic Pro compiler:

pulsout portb.0, 150

The angles of each servo is declared using its pulse counterpart

```
'============ PULSE widths for ANGLE ranges =============

'50 far left
'100 left
'150 center
'200 right
'250 far right


'min - lower limit of the angle pulse
'max - higher limit of the angle pulse


'the higher the range, the larger angle


'TOP
topmin CON 120
topmax con 200

'MID
midmin CON 120
midmax con 180

'BOTTOM
botmin CON 80
botmax con 220
```

Each of the buttons are polled individually using an IF-END IF method.

```
'=================== TOP servo control =======================

IF PORTB.0 = 0 then               'TOP-right switch [gray]

    PORTA.6 = 1               'LED1 on
    PORTA.7 = 0               'LED2 off
    PORTA.3 = 0               'LED3 off


    'decrease pulse until it equals the lower limit pulse

    IF pulseWidth1 <= topmin Then
             pulseWidth1 = topmin
    Else
         pulseWidth1 = pulseWidth1 - 1
    Endif


ENDIF
```

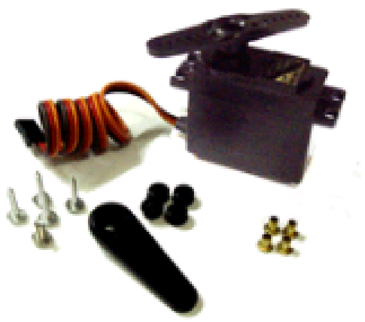
3.2.6 Hardware Resources

- **Servomotor**

TowerPro SG-5010

Stall Torque:
5.2kg/cm @ 4.8V 6.4kg/cm @ 6.0V
Speed:
0.2 seconds/60deg. @ 4.8V
or 0.16 seconds/60deg. @ 6.0V
Dimensions:
41mm x 20mm x 38mm
Temperature Range:
0c -55c
Operating Voltage:
4.8V - 6.0V
Weight:
41grams

Ball Bearing
PHP 590.00



TowerPro MG-995
Digital Servo

Stall Torque:
9.0kg/cm @ 4.8V
11kg/cm @ 6.0V
Speed:
0.17 seconds/60deg. @ 4.8V
0.13 seconds/60deg. @ 6.0V
Dimensions:
40.6 x 19.8 x 37.8mm
Temperature Range:
0c - 55c
Operating Voltage:
3.5V - 7.2V
Weight:
56grams

Metal Gear, Ball Bearing
PHP 890.00

Figure 3.31 **TowerPro SG-5010 and MG-995 servo motors**

➢ **DC motor**

Figure 3.32 **3V DC Motor**

➢ **PIC 16F819**

This powerful (200 nanosecond instruction execution) yet easy–to-program (only 35 single word instructions) CMOS Flash-based 8-bit microcontroller packs Microchip's powerful PIC® architecture into an 18-pin package and is upwards compatible with the PIC16C7x, PIC16C62xA, PIC16C5X and PIC12CXXX devices. The PIC16F819 features 8MHz internal oscillator, 256 bytes of EEPROM data memory, a capture/compare/PWM, a synchronous serial port that can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and 5 channels of 10-bit Analog-to-Digital (A/D) converter that make it ideal for advantage analog / integrated level applications in automotive, industrial, appliances and consumer applications.

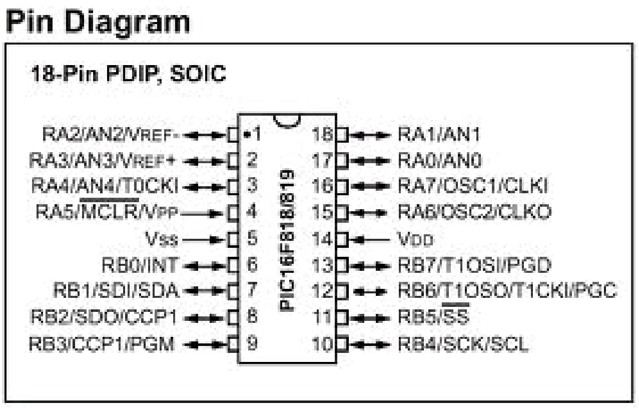


Figure 3.33 **PIC16F819**

➢ 78xx 3-terminal Positive Voltage Regulators

The 78xx (sometimes LM78xx) is a family of self-contained fixed linear voltage regulator integrated circuits. The 78xx family is commonly used in electronic circuits requiring a regulated power supply due to their ease-of-use and low cost.

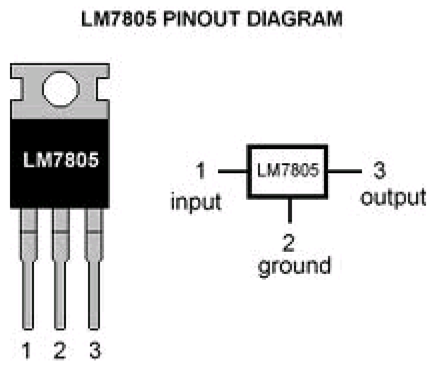


Figure 3.34 **LM7805 voltage regulator**

➢ U-bracket

The U brackets are fabricated to fit the servo motor and prepare the motor for assembly with other brackets.

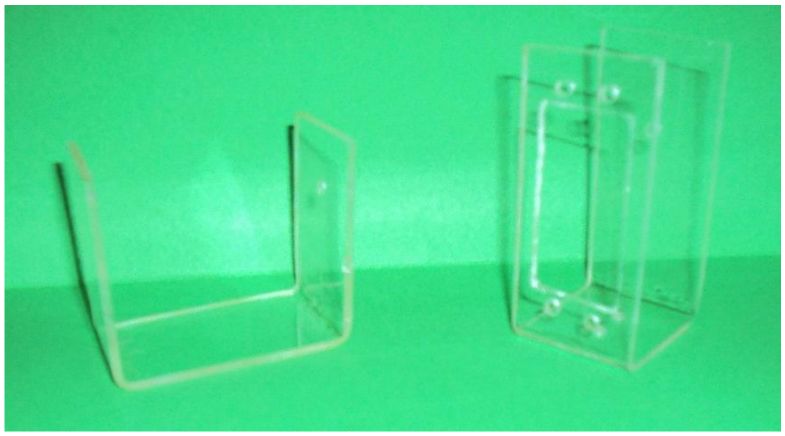


Figure 3.35 **Fiber-glass U-Brackets**

3.2.7 Software Resources

- **PIC BASIC PRO**

The PICBASIC PRO™ Compiler (or PBP) makes it even quicker and easier for you to program Microchip Technology’s powerful PIC® microcontrollers (MCUs). The English-like BASIC language is much easier to read and write than assembly language. [MICR2008]

- **WinPic800**

This free software serves as the burner that transfers the compiled HEX codes from the host PC through the MCU Programming device into the microcontroller. It allows you to program all types of serial PICs using Windows 95/98/NT/2000/ME/XP. It supports many PIC Microcontroller families and 24Cxx EEPROMs. It can also be called through a command line providing GUI optimization [WINP2010]

- **Proteus PCB Design Package**

Proteus PCB design combines the ISIS schematic capture and ARES PCB layout programs to provide a powerful, integrated and easy to use suite of tools for professional PCB Design. It includes an integrated shape based autorouter and a basic SPICE simulation capability as standard.

The developers use Proteus ISIS and ARES to schematic designs and board layouts for the project hardware. [LABC2011]

APPENDIX A Bill of Materials

Materials	Quantity	Unit Price	Total Price
SG-5010 Servo Motor	3	P 590.00	P 1770.00
MG-995 Servo Motor	1	890.00	890.00
DC Motor	1	25.00	25.00
Resistors	16	0.25	4.00
Electrolytic Capacitors	4	2	8.00
Nonpolar Capacitors	3	0.5	1.50
7805	2	5	10.00
7806	1	5	5.00
L293D	1	120	120.00
LM317	1	5	5.00
LED	3	3	9.00
16F819	1	150	150.00
2- Pin Header	2	2	4.00
DPDT slide switch	1	10	10.00
Push Button	7	10	70.00
3-pin cable	1	15	15.00
Wires (per m)	3	5	15.00
Screws	-	-	40.00
Fiber Glass	-	-	30.00
Travel Expenses	-	-	500.00
TOTAL			P 3681.50

APPENDIX B Codes

```
'##############################################################################
'##############################################################################

'ooooooooo.             .o8                   .    o8o
'`888   `Y88.          "888                 .o8    `"'
' 888   .d88'  .ooooo.   888oooo.   .ooooo.  .o888oo oooo   .ooooo.
' 888ooo88P'  d88' `88b  d88' `88b d88' `88b   888   `888  d88' `"Y8
' 888`88b.    888   888  888   888 888   888   888    888  888
' 888  `88b.  888   888  888   888 888   888   888 .  888  888   .o8
'o888o  o888o `Y8bod8P'  `Y8bod8P' `Y8bod8P'   "888" o888o `Y8bod8P'
'
'     .888.                                      '######################
'    .8"888.     oooo d8b ooo. .oo.  .oo.        '######################
'   .8' `888.    `888""8P `888P"Y88bP"Y88b       '######################
'  .88ooo8888.    888      888   888   888       '######################
' .8'     `888.   888      888   888   888       '######################
'o88o     o8888o d888b    o888o o888o o888o      '######################
                                                 '######################
                                                 '######################
'                                                '######################
'                       -&-#                     '######################
'                           \                    '######################
'                           #                    '######################
'        ________       ____|_____               '######################
'       /    #   |     |    #     |              '######################
                                                 '######################
                                                 '######################
'BSCoE 1001-A batch 2012                         '######################
'COEELE2 Robotics and Mechatronix                '######################
                                                 '######################
'PROJECT ENGINEER                                '######################
'Engr. Paolo Lozada                              '######################
                                                 '######################
'HARDWARE ENGINEERS                              '######################
'Jan Paul Portuguez                              '######################
'Walter Salatandre                               '######################
'Agustin Ace DOnes                               '######################
'Romeo Paderna III                               '######################
                                                 '######################
'SOFTWARE ENGINEERS                              '######################
'Lenmor Dimanalata                               '######################
'Jon Jon Legaspi                                 '######################
                                                 '######################
'RESEARCH and DEVELOPMENT ENGINEERS              '######################
'Chona Mojillo                                   '######################
'Armeo TOlentino                                 '######################
                                                 '######################
'##############################################################################
'##############################################################################
'##############################################################################


'proto 1 : 1-27-2012
'proto 2 : 3-13-2012
```

```
'proto 3 : 3-20-2012
'proto 4 : 3-24-2012
'proto 5:  4-01-2012

'DC Motor -> seperate switch (not controlled by PIC)
            ' gray button

'TOP servo   -> PORTA.6

    'LED indicator YELLOW -> PORTA.7

    '-> initial pos at center

    'goes for 75 degrees 115:185 pulse

    'right -> PORTA.3 [gray button]
    'left -> PORTA.2 [blue button]


'MID servo  -> PORTB.7

    'LED indicator BLUE -> PORTA.0

    '-> initial pos at center

    'goes for 110 degrees 95:205 pulse

    'right -> PORTB.0 [green button]
    'left -> PORTA.4 [red button]


'BOTTOM servo    -> PORTB.6

    'LED indicator GREEN -> PORTA.1

    '-> initial pos at center

    'goes for 90 degrees 75:175 pulse

    'right -> PORTB.2 [yellow button]
    'left -> PORTB.1 [black button]

'RAMP servo     ->  PORTB.5

    '-> initial pos at center

    'goes for 160 degrees 75:175 pulse

    'right -> PORTB.4 [yellow button]
    'left -> PORTB.3 [black button]


INCLUDE "modedefs.bas"

define _XTAL_FREQ 4000000
OSCCON = %01100000        '  Set intRC to 4 MHz
ADCON1 = 6                '  Turn off ADC, all pins must be digital I/O
```

```
TRISA = %00011100
                                        'RA0 -> 0 midservo LED indicator BLUE
                                        'RA1 -> 0 botservo LED indicator
GREEN
                                        'RA2 <- 1 ts_left
                                        'RA3 <- 1 ts_ryt
                                        'RA4 <- 1 ms_left
                                        'RA5 -- x [reserved MCLR/VPP pin]
                                        'RA6 -> 1 topservo
                                        'RA7 -> 1 topservo LED indicator
YELLOW


TRISB = %00011111

                                        'RB0 <- 1 ms_ryt
                                        'RB1 <- 1 bs_left
                                        'RB2 <- 1 bs_ryt
                                        'RB3 <- 1 rs_left
                                        'RB4 <- 1 rs_ryt
                                        'RB5 -> 0 rampservo
                                        'RB6 -> 0 botservo
                                        'RB7 -> 0 midservo




PORTA =0                            'off PORTA at start
PORTB =0                            'off PORTB at start



pulseWidth1 var byte                'variables to hold each servo pulse
pulseWidth2 var byte
pulseWidth3 var byte
pulseWidth4 VAR BYTE

refreshPeriod CON 20                'refresh/pause period 20ms
                                    '[servo expects a pulse every 20ms]

'============ PULSE widths for ANGLE ranges =============

'60 extreme left (not recommended)
'70 far left
'100 left
'150 center
'200 right
'250 far right
'270 extreme right   (not recommended)


'min - lower limit of the angle pulse
'max - higher limit of the angle pulse
```

```
'the higher the range, the larger angle


'TOP
topmin CON 105
topmax con 200

'MID
midmin CON 90
midmax con 210

'BOTTOM
botmin CON 75
botmax con 225

'RAMP
rampmin CON 75
rampmax CON 225

'========================================



pulseWidth1 = 110   'set starting position at working position, a bit right
pulseWidth2 = 100   'set starting position at working position, a bit right
pulseWidth3 = 150   'set starting position at center / pulse 150
pulseWidth4 = 150    'set starting position at center / pulse 150


start:

'take the output pins low so we can pulse 'em high
LOW PORTA.6         'TOP
LOW PORTB.7         'MID
LOW PORTB.6         'BOTTOM
LOW PORTB.5         'RAMP


' pulse the pins / output to servo by sending PWM (Pulse Width Modulated)
signal

PulsOut PORTA.6, pulseWidth1

PulsOut PORTB.7, pulseWidth2

PulsOut PORTB.6, pulseWidth3

PulsOut PORTB.5, pulseWidth4

' pause for as long as needed

Pause refreshPeriod


'=================== TOP servo control =======================

IF PORTA.3 = 0 then             'ts_ryt switch [gray]
```

```
    PORTA.7 = 1             'LED1 on
    PORTA.0 = 0             'LED2 off
    PORTA.1 = 0             'LED3 off


    'decrease pulse until it reaches the lower limit pulse

    IF pulseWidth1 <= topmin Then
             pulseWidth1 = topmin
    Else
         pulseWidth1 = pulseWidth1 - 1
    Endif


ENDIF


IF PORTA.2 = 0 then             'ts_left switch [blue]

    PORTA.7 = 1             'LED1 on
    PORTA.0 = 0             'LED2 off
    PORTA.1 = 0             'LED3 off


    'increase pulse until it exceeds the higher limit pulse


    IF pulseWidth1 > topmax Then
      pulseWidth1 = topmax
    Else
      pulseWidth1 = pulseWidth1 + 1
    Endif


ENDIF


'=================== MID servo control =======================

IF PORTB.0 = 0 then             'ms_ryt  switch [green]

    PORTA.7 = 0             'LED1 off
    PORTA.0 = 1             'LED2 on
    PORTA.1 = 0             'LED3 off

    'decrease pulse until it equals the lower limit pulse

    IF pulseWidth2 <= midmin Then
             pulseWidth2 = midmin
    Else
         pulseWidth2 = pulseWidth2 - 1
    Endif


ENDIF
```

```
IF PORTA.4 = 0 then               ' ms_left switch [red]

    PORTA.7 = 0               'LED1 off
    PORTA.0 = 1               'LED2 on
    PORTA.1 = 0               'LED3 off

    'increase pulse until it exceeds the higher limit pulse

    IF pulseWidth2 > midmax Then
      pulseWidth2 = midmax
    Else
      pulseWidth2 = pulseWidth2 + 1
    Endif


ENDIF


'=================== BOTTOM servo control =======================


  IF PORTB.2 = 0 then               'bs_ryt switch [yellow]

    PORTA.7 = 0               'LED1 off
    PORTA.0 = 0               'LED2 off
    PORTA.1 = 1               'LED3 on

    'decrease pulse until it equals the lower limit pulse

    IF pulseWidth3 <= botmin Then
             pulseWidth3 = botmin
    Else
         pulseWidth3 = pulseWidth3 - 1
    Endif


ENDIF

IF PORTB.1 = 0 then               'bs_left [black]

    PORTA.7 = 0               'LED1 off
    PORTA.0 = 0               'LED2 off
    PORTA.1 = 1               'LED3 on

    'increase pulse until it exceeds the higher limit pulse

    IF pulseWidth3 > botmax Then
      pulseWidth3 = botmax
    Else
      pulseWidth3 = pulseWidth3 + 1
    Endif


ENDIF
```

```
'=================== RAMP servo control =======================

  IF PORTB.4 = 0 then              'rs_ryt switch [green]

    PORTA.7 = 0               'LED1 off
    PORTA.0 = 0               'LED2 off
    PORTA.1 = 0               'LED3 on

    'decrease pulse until it equals the lower limit pulse

    IF pulseWidth4 <= rampmin Then
             pulseWidth4 = rampmin
    Else
         pulseWidth4 = pulseWidth4 - 1
    Endif


ENDIF

IF PORTB.3 = 0 then              'rs_left [blue]

    PORTA.7 = 0               'LED1 off
    PORTA.0 = 0               'LED2 off
    PORTA.1 = 0               'LED3 on

    'increase pulse until it exceeds the higher limit pulse

    IF pulseWidth4 > rampmax Then
      pulseWidth4 = rampmax
    Else
      pulseWidth4 = pulseWidth4 + 1
    Endif


ENDIF


GoTo start          'loop again to wait for input

END
```

BIBLIOGRAPHY

[HARR2012] T Harris (2012). ‘ HowStuffWorks: The Robotic Arm’.
http://science.howstuffworks.com/robot2.htm

[IOVI2004] J Iovine (2004). PIC Robotics, A Beginner’s Guide to
Robotics Projects Using the PICmicro. USA: McGraw-Hill Companies, Inc.

[KAUD2010] A Kauderer (2010). ‘Canadarm2 and the Mobile Servicing System’.
http://www.nasa.gov/mission_pages/station/structure/elements/mss.html

[LABC2011] Labcenter Electronics (2011). ‘Labcenter Electronics - Proteus PCB Design - Integrated EDA/CAD PCB Software’. http://www.labcenter.com/products/pcb_overview.cfm

[MICR2008] microEngineering Labs, Inc (2008). PICBASIC PRO Compiler Manual. USA: microEngineering Labs, Inc.

[ROBO2012a] RobotWorx (2012). ‘Motoman UP6 Robot’.
http://www.robots.com/motoman/up6/196

[ROBO2012b] RobotWorx (2012). ‘Fanuc ArcMate 120i Robot’.
http://www.robots.com/fanuc/arcmate-120i/53

[SOCI2012] Society of Robots (2012). ‘How to Build a Robot Tutorial - Society of Robots’.
http://www.societyofrobots.com/robot_arm_tutorial.shtml

[TWEN2008] D Tweney. ‘Dean Kamen’s Robot Arm Grabs More Publicity’.
http://www.wired.com/gadgetlab/2008/05/dean-kamens-rob/

[WINP2010] WinPic800 (2010). ‘WinPic800 – Home’.
http://www.winpic800.com/index.php?lang=en