18-331

Junior Project:

Coordination Control of

The Lunar Mobile Manipulator

By

Alongkrit Chutinan

Spring 1994

Final Project Report

Objectives

Two main objectives were pursued during the course of this project. The first objective was to come up with an optimal strategy to coordinate the motion of a five degree of freedom manipulator and its mobile base. The second one is to develop a software controller to achieve smooth motions of the robot. This report is thus organized around these two objectives.

Coordination Scheme

By Optimizing A Weighted Function

There are virtually infinite ways to coordinate the position of the mobile base and the configuration of the manipulator so that the position of the end effector is at the specified point in cartesian space. However, we need to choose only one combination, which has to be optimal in some sense. The question then arises as to what we would like to optimize in trying to coordinate the motion of the base and the manipulator to reach the final position. For the DLMM, we consider three issues. The first issue we are concerned with is the manipulability of the manipulator. The second one is the distance that the base has to move in order to get to the goal position. Finally, the third issue involves the static stability of the whole robot when it takes on a load. Before we continue and look at each of these issues, let us consider the general problem setup first.

General Problem Setup

Consider the case when the base of the manipulator, which is attached to the mobile base, is initially at the coordinates (0,0,0) and the goal position where the final position of the end effector is supposed to be is specified by the coordinates (x,y,z) as shown below.

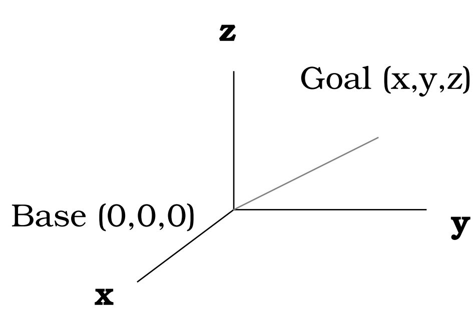
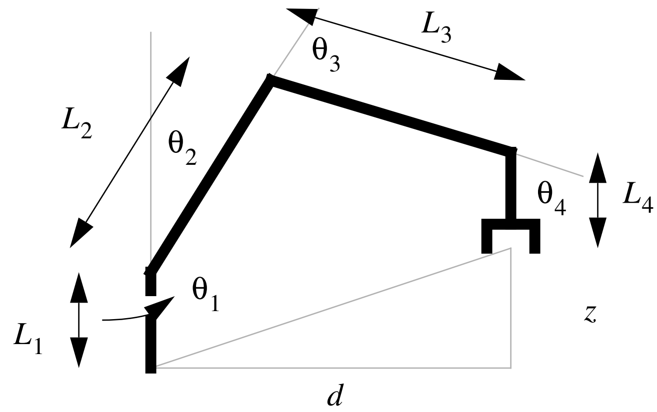


Figure 1

The side view of the manipulator at its final configuration is shown below,



d is the distance that the manipulator stretches in the x-y plane

Figure 2

From figure 2, we can write the forward kinematics of the manipulator as follows.

$$x = -(L_2\sin\theta_2 + L_3\sin(\theta_2+\theta_3) + L_4\sin(\theta_2+\theta_3+\theta_4))\sin\theta_1 \tag{1}$$

$$y = (L_2\sin\theta_2 + L_3\sin(\theta_2+\theta_3) + L_4\sin(\theta_2+\theta_3+\theta_4))\cos\theta_1 \tag{2}$$

$$z = L_1 + L_2\cos\theta_2 + L_3\cos(\theta_2+\theta_3) + L_4\cos(\theta_2+\theta_3+\theta_4) \tag{3}$$

In figure 2, the end effector is shown pointing down. In general, the orientation of the end effector will be specified, depending on what angle we want the manipulator to approach the goal. This specification imposes a constraint on the joint angles. In the above case, the end effector is pointing down, thus the constraint imposed on the joint angles is

$$\theta_2 + \theta_3 + \theta_4 = 180° \tag{4}$$

In general, equation (4) would be

$$\theta_2 + \theta_3 + \theta_4 = \gamma \tag{5}$$

or $$\theta_4 = \gamma - \theta_2 - \theta_3$$

where γ is a specified angle describing the orientation of the end effector.

From the forward kinematics of the manipulator, the distance that the manipulator stretches in the x-y plane can be written as

$$d = L_2\sin\theta_2 + L_3\sin(\theta_2+\theta_3) + L_4\sin(\theta_2+\theta_3+\theta_4) \tag{6}$$

We will refer to these equations as we proceed and discuss the three factors we want to optimize in trying to get the robot the goal position.

Manipulability of The Manipulator

Manipulability is a quantity that represents the ability of the manipulator to move around at a given configuration. We will assume that the greater manipulability, the better the robot can move or change to a new configuration. Therefore, we would like the maximize the manipulability of the manipulator at its final configuration to make it easier for the robot to perform the next task.

Neglecting link 4 in figure 2, the manipulator's forward kinematics reduces to

$$x = -(L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)) \sin\theta_1$$

$$y = (L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)) \cos\theta_1$$

$$z = L_1 + L_2 \cos\theta_2 + L_3 \cos(\theta_2 + \theta_3)$$

By directly differentiating the forward kinematic equations with respect to time and putting all the equations in matrix form, we have that

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \underbrace{\begin{bmatrix} -(L_2 s_2 + L_3 s_{23}) c_1 & -(L_2 c_2 + L_3 c_{23}) s_1 & -L_3 c_{23} s_1 \\ -(L_2 s_2 + L_3 s_{23}) s_1 & (L_2 c_2 + L_3 c_{23}) c_1 & L_3 c_{23} c_1 \\ 0 & -(L_2 s_2 + L_3 s_{23}) & -L_3 s_{23} \end{bmatrix}}_{\mathbf{J}} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

where

$$s_1 = \sin\theta_1 \qquad c_1 = \cos\theta_1$$

$$s_2 = \sin\theta_2 \qquad c_2 = \cos\theta_2$$

$$s_{23} = \sin(\theta_2 + \theta_3) \qquad c_{23} = \cos(\theta_2 + \theta_3)$$

The matrix J above is the so called *Jacobian matrix* and the manipulability ω is defined as the absolute value of the determinant of the *Jacobian matrix*.

$$\omega = |det(J)|$$

The final expression for the manipulability of the DLMM after simplification is found to be

$$\omega = |L_2 L_3 (L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)) \sin\theta_3| \qquad (7)$$

Distance That The Mobile Base Travels

As the mobile base is relatively harder to move than the manipulator, moving the base usually consumes a lot more energy than moving the manipulator. The distance that the base has to move is related to the energy required to move the base. Therefore, if we minimize the distance that the base has to move, we also minimize the energy consumption of the mobile base.

Let D be the total distance along the x-axis between the initial position of the base and the goal point as shown in the figure below.

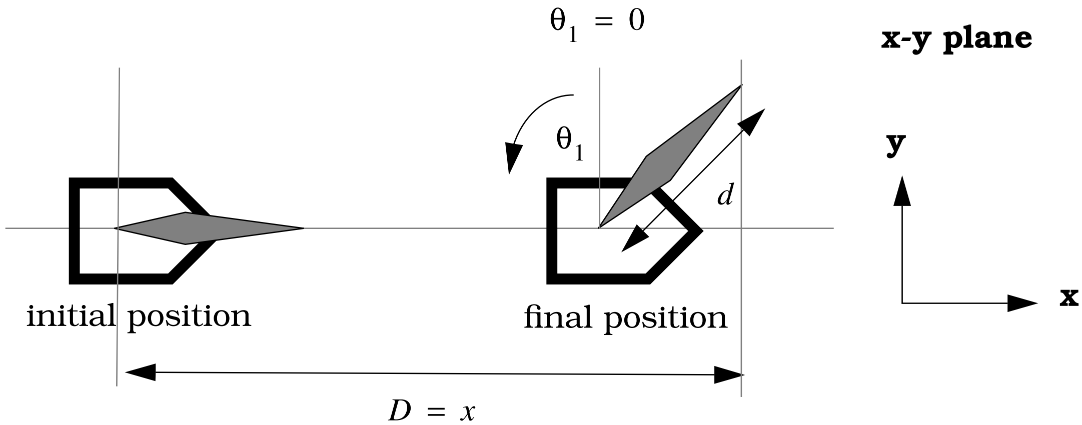


Figure 3

We will consider the case in which the base only moves in a straight line along the x-axis. From figure 3, the distance that the robot has to move in order to get to the final position is given by $|D + d\sin\theta_1|$. The reason why we have to take the absolute value is because the optimization result might give a value of $D + d\sin\theta_1$ that is negative in which case the base will actually move backwards.

Static Stability of The Robot When Taking On A Load

The robot might fall over if the load it is taking on is too heavy. It might also fall over if it is stretching too much to reach the load, since the moment produced by the load increases as the stretching distance of the arm increases. The chance of the robot falling over can be reduced by keeping the center of mass of the whole system with the load closest to where it was before without the load. Since all gravitational forces (i.e. the weights) are vertical, they cannot produce any moment around the z-axis. Thus, we can restrict our attention to only the x-y plane when considering the center mass of the system.

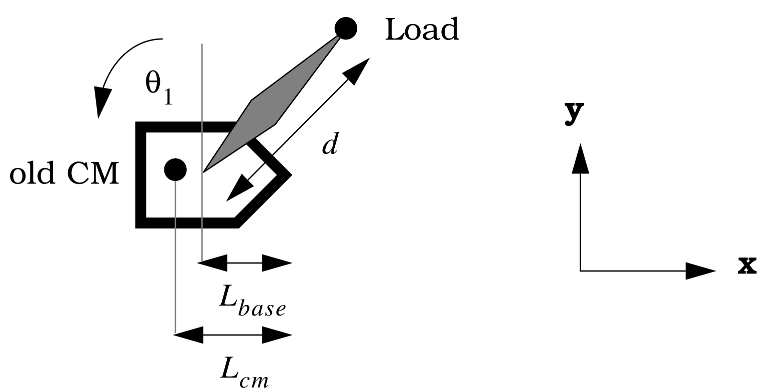


Figure 4

From figure 4, we can find the location of the new center of the mass with respect to the old center of mass as follows.

$$\bar{x} = \frac{((L_{cm} - L_{base}) - d\sin\theta_1)\, W_{load}}{(W_{load} + W_{base})}$$

$$\bar{y} = \frac{(d\cos\theta_1)\, W_{load}}{(W_{load} + W_{base})}$$

To keep the new center of mass closest to the old one, we minimize the distance square $\bar{x}^2+\bar{y}^2$.

Formulating The Optimization Function

In summary, we would like to perform the following optimizations simultaneously when we move the robot to the final position. The first one is to maximize the manipulability of the robot at its final configuration. The second one is to minimize the distance that the base travels. The third one is to minimize the distance square of the new center of mass from the old one.

To perform all three optimization at the same time, we would have to put all the factors we are considering into one function. Each element in the optimization function is weighted according to its significance. For example, we might put the priority on the static stability of the robot and make the coefficient for the distance square of the center of mass larger than others. All the weighted elements in the combined function have to change in the same direction. We have chosen to perform the optimization by maximizing the combined function. Thus, we have to put a negative sign in front of every element we want to minimize before to turn it into a maximization problem.

Now we can formulate our optimization function which we would call f with three weighted elements in it, namely the manipulability ω the distance that the base travels $|D+d\sin\theta_1|$ and the distances square of the center of mass from the old center of mass $\bar{x}^2+\bar{y}^2$.

$$f = -k_1|D + d\sin\theta_1| + k_2\omega - k_3(\bar{x}^2 + \bar{y}^2) \tag{8}$$

The coefficients k_1, k_2, and k_3 are positive, and f is to be maximized.

Substituting the expression for θ_4 into equations (3) and (4), we obtain that the optimization function f can be written in terms of only θ_2 and θ_3, and its constraint is essentially equation (3) with θ_4 substituted. All the parameters other than θ_2 and θ_3 are completely specified.

Thus, our optimization task can be stipulated as follows.

maximize $\quad f(\theta_2, \theta_3) = -k_1|D + d\sin\theta_1| + k_2\omega - k_3(\bar{x}^2 + \bar{y}^2)$

subject to $\quad z = L_1 + L_2\cos\theta_2 + L_3\cos(\theta_2 + \theta_3) + L_4\cos(\gamma)$

where $D = x$

$$d = L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3) + L_4 \sin(\gamma)$$

$$\omega = \left| L_2 L_3 \left(L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3) \right) \sin\theta_3 \right|$$

$$\bar{x} = \frac{((L_{cm} - L_{base}) - d\sin\theta_1) W_{load}}{(W_{load} + W_{base})} \qquad \bar{y} = \frac{(d\cos\theta_1) W_{load}}{(W_{load} + W_{base})}$$

$$\theta_1 = \begin{cases} \mathrm{acos}\left(\frac{y}{d}\right) & , \mathrm{x} < 0 \\ -\mathrm{acos}\left(\frac{y}{d}\right) & , \mathrm{x} > 0 \end{cases}$$

An attempt to solve this optimization problem using the Lagrange multiplier method has been made. However, the intermediate results were far too complicated, and the final analytical result was never achieved. Therefore, a brute force method using a computer program to find the combination of θ_2 and θ_3 that maximizes f has been used instead.

To speed up the optimization process, instead of going through all combinations of θ_2 and θ_3, we will just look for θ_3 that satisfy the constraint for a given θ_2, which will run from 0 to 90 degrees by 1 degrees.

For a given θ_2, we can solve for θ_3 by the following manipulations of the constraint equation.

$$z = L_1 + L_2\cos\theta_2 + L_3\cos(\theta_2 + \theta_3) + L_4\cos\gamma$$

$$z = L_1 + L_2\cos\theta_2 + L_3\cos\theta_2\cos\theta_3 - L_3\sin\theta_2\sin\theta_3 + L_4\cos\gamma$$

$$z = L_1 + L_2\cos\theta_2 + L_3\cos\theta_2\cos\theta_3 - L_3\sin\theta_2\sqrt{1 - (\cos\theta_3)^2} + L_4\cos\gamma$$

$$L_3\sin\theta_2\sqrt{1 - (\cos\theta_3)^2} = (L_1 + L_4\cos\gamma - z + L_2\cos\theta_2) + L_3\cos\theta_2\cos\theta_3$$

$$(L_3\sin\theta_2)^2(1 - (\cos\theta_3)^2) = (L_1 + L_4\cos\gamma - z + L_2\cos\theta_2)^2 + (L_3\cos\theta_2\cos\theta_3)^2 + 2(L_1 + L_4\cos\gamma - z + L_2\cos\theta_2) L_3\cos\theta_2\cos\theta_3$$

$$\Rightarrow \quad L_3^2(\cos\theta_3)^2 + 2(L_1 + L_4\cos\gamma - z + L_2\cos\theta_2) L_3\cos\theta_2(\cos\theta_3) + [(L_1 + L_4\cos\gamma - z + L_2\cos\theta_2)^2 - L_3^2(\sin\theta_2)^2] = 0$$

The above result is a quadratic equation in cos θ_3. Thus, we have two solutions for cos θ_3. We can take an inverse cosine for each solution to get θ_3 that satisfy the constraint.

Simulation Results

The coordination scheme was simulated in MATLAB. All the MATLAB scripts used in this simulation are also attached to this report. Most of the variable names are consistent with the conventions established in the previous discussions. Thus, the reader can refer to the previous discussions and figures to understand the MATLAB simulation code. The script file *coordinate.m* is the one to be run in order to start the simulation. The script file will ask the user to specify the goal coordinates (x,y,z), the wrist orientation γ, and the weight of the load (*Wload*). The resulting run time was reasonably fast and can be used to generate a real-time optimization and execution of the coordination control.

The simulation results are shown on the next 4 pages. We can see the effect of the load on the optimization result. Without out the load, the optimization only compromises between the manipulability and the distance that the base travels. With the load, the result is different. The robot now tries, at the same time, to move its base as little as possible, maintain as high manipulability as possible, and keep the overall center of mass as close to its center of mass as possible. The last two plots show the same kind of result when another wrist orientation is specified. The optimization result will be different if the weighing coefficients, which are now set to be all 1's, are altered. These coefficients can be experimentally tuned in the simulation to achieve a reasonable behavior of the robot before the actual implementation.

Software Controller Design

The controller implemented to control the DLMM is not a conventional one. It basically is a PI controller with some additional features, one of which is the damping term which is proportional to the velocity of the output. The basic controller output is of the form

$$u(t) = K_p e + K_i \int e - K_{damping} v$$

where e is the error signal r(t) - y(t)

v is the joint velocity y'(t)

The underlying idea behind this controller is the difficulty of establishing an accurate model for a complex system dynamics. That is, designing the controller according to the plant dynamics can become increasingly difficult as we have a more complex model of the plant. The added damping term makes it possible to achieve the response that is fast, yet being overdamped in nature by the trial and error tuning of the gains Kp, Ki, and Kdamping. The tuning of the controller can be done without actually having to know the detailed dynamics of the system. We can just assume that the higher the output voltage, the greater the torque for the motor. Even though this assumption is not necessarily true all the time, it is good enough for the purpose of designing our controller.

The main purpose of putting the damping term into the controller is to prevent overshoots. Most of the physical systems tend to overshoot with a high proportional gain. The output will move at a high speed, gaining some momentum and causing overshoots. The damping term is proportional to the velocity of the output; however, it is going against the motion of the output. At small speed, the damping term won't have much effect on the control output. However, if the speed is too high, the damping term will help slow the response down as it can become so large that the overall control output is negative. Once the speed is slowed down, the proportional gain takes charge again. Thus, the proportional gain can be cranked up to a higher value than usual. The higher proportional gain can help the system get to the set point faster when the error is small. Increasing the integral term to push the system to the steady state faster is usually not a good idea for it can cause some overshoots, and it would actually take more time for the system to settle with a large integral gain.

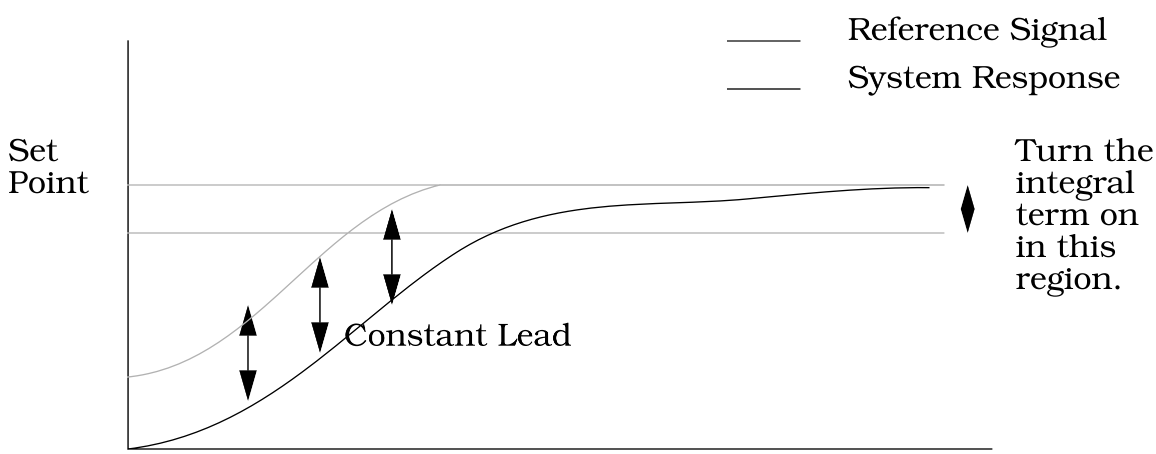
There are some additional detail we need to take care of in our controller. The first one has to do with the integral term. Since we usually do not want a large integral term, we would not turn it on until we got into some close range around the set point.

The second thing to be taken care of is friction in the system. We defined some friction terms for the controller. The right values of friction terms will help the controller overcome the friction faster so the system will not get stuck when the proportional term is too small (e.g. small error). However, the friction terms defined are only used to overcome the static friction. In other words, they are turned on only when the velocity is zero. At some point where the error is really small, we cut the friction term out and let the integral term takes care of pushing the joint position to the set point.

The third one is the gravity terms. Gravity terms are needed to compensate for gravitational effects on some joints that the gravity compensation system cannot compensate for.

Reference Signal Generation

To limit the maximum error signal, we made the reference position lead the measured position towards the set point by a constant amount (currently 7.5 degrees). However, when the leading reference signal reaches the set point, we hold it there. The a hypothetical lead reference signal and the response is shown below.



In *controller.c*, the portion of the code that generates the reference signal lies in the function *feedfor()*. Please refer to the commented source code attached to this report for more information.

Experimental T uning of The Controller

As mentioned earlier, the controller tuning can be done by trial and error, by looking at the real response of the system and adjusting the gains accordingly. However, we can record the data for each trial run and plot it to give us some idea about how the system responds and to help us tune the controller. We can look at various responses of the system to help us adjust the gains, the friction terms, and the gravity terms.

There are some rules of thumb that we can use to decide whether the current gains for the controller are appropriate. If the response is too slow, or the torque applied by the motor is too weak when we try to twist it by hands, the proportional gain is probably too small. If the motion was too fast, then we reduce the proportional gain. If near the final set point, the motor is shaking back and forth and it takes so long to settle, then the integral gain is probably too big. The integral gain is probably too small if the system takes a considerable amount of time to go from the point where the error is small to the set point. The damping gain should not be too big. The output position will never get to the set point if the damping term is too sensitive and keeps pushing hard against the desired motion. The friction term should be just enough or almost enough for the motor to overcome the friction. The friction term should be helping the proportional term when the system get stuck. It should not become the dominant term all the time.

The controller is implemented in the routine *feedback()* in the file *controller.c*. See the source code for the value of gains used for each joint.

Samples of The Real Responses And Controller Problems

Two sets of data were gathered from two experimental runs of the controller. The results were fairly satisfactory. Most of the response were overdamped and we did not see much of the oscillations going on. In some of the plots, we can see stair-case-liked kind of responses. These stair-case pattern is presumably due to friction in the system. The system got stuck at some point then moved on because we had a static friction term for each joint. As soon as the system moved, the static friction term went away so the system got stuck again at some other point. These results suggested that we add another friction term to our controller, namely the kinetic friction term, to help smoothing the responses.

Even though we had a damping term for each joint to damp out the response of each joint and prevent overshoots, the damping term is sometimes not enough to compensate for the momentum or the gravitational forces that acted on the system. One key problem was that the damping term cannot be raised so high as it would make the system become too sensitive to any change in velocity and would induced a lot of oscillation. Small change in velocity would cause the controller to output a high voltage in the opposite direction which would cause the big velocity change in that direction. This change in velocity would again cause the controller to try to push the system back in the original direction even harder. The cycle would repeat and the system could even become unstable if the damping term is not properly set.

The data for joint 7 (steering) was not included since we were working on the case where the base only moved in a straight line and the steering was kept straight all the time.

Another problem with this controller is that right now the controller cannot operate over all range of dynamics. If the robot has to carry a heavy load, the controller might not be able to make the robot hold the load and stay in the specified configuration at the same time. This is because the gains are too low right now (The main objective at first was to achieve a smooth motion for the robot without the load). If the gains are raised, the response would not be smooth anymore, but more force could be applied to each joint. Perhaps, the next thing we have to think about is the development of some kind of adaptive controller that can operate over a wide range of dynamics and produce a desired smooth motion for the robot.

User Interface

The user can execute the coordinating control of the DLMM by writing a sequence of command in a command file. Eight basic commands are already developed for the DLMM. Each command is a letter followed by variable number of parameters, depending on which command is being executed. Each line contains only one command. The user interface part lies essentially in the *feedfor()* routine in the file *controller.c.* (See the source code).

'g': Go to A Specified Point

Command Syntax: g x y z wrist_orient hand_orient check_time
x,y,z: coordinates of the goal position
wrist_orient: angle (in degrees) describing the orientation of the wrist (γ)
hand_orient: angle in degrees describing the orientation of the gripper (joint 5). It is only used when gamma is 180 degrees, i.e. when the end effector is pointing straight down. If a zero degrees is specified, the hand orientation will be parallel to the gripper that is attached to the base. If a 90 degree is specified, it will be perpendicular to the base gripper.
check_time: the amount of time (in 1/10th's of a second) during which the response of the system need to be accurate (near the set point) and stable before the controller can read in the next command.

'w': Specify The Weight of The Load

Command Syntax: w Wload
Wload: The weight of the load in lbs.

's': Set The Steering Angle

Command Syntax: s steering_angle check_time
steering_angle: the desired steering_angle in degrees

'b': Move The Base To A Specified Point

Command Syntax: b base_position check_time
base_position: the desired position of the base along the x-axis

'j': Specify The Joint Positions

Command Syntax: j T1 T2 T3 T4 T5 check_time
Ti: The desired joint angles in degrees

'o' Open The Gripper

Command Syntax: o check_time
Need to use the gripper enable button during the execution of this command.

‘c’: Close The Gripper

Command Syntax: c check_time

Need to use the gripper enable button during the execution of this command.

‘q’: Quit

Command Syntax: q

Quit, end the execution.

Example of A Command File

An example of a command file *sequence* is shown below.

```
w 40
j -90.0 45.0 90.0 45.0 0.0 10
g 20.0 -30.0 20.0 180 0.0 0
g 20.0 -30.0 5.0 180 0.0 10
g 20.0 -30.0 -1.0 180 0.0 10
c 20
g 60.0 40.0 20.0 180 90 0
g 60.0 40.0 10.0 180 90 10
g 60.0 40.0 0.0 180 90 10
o 20
g 60.0 40.0 15.0 180 90 0
g 76.0 -30.0 10.0 180 90.0 10
g 76.0 -30.0 -1.0 180 90.0 10
c 20
g 76.0 -30.0 10.0 180 90 0
g 60.0 40.0 20.0 180 90 0
g 60.0 40.0 10.0 180 90 10
g 60.0 40.0 0.0 180 90 10
o 20
g 60.0 40.0 10.0 180 90 10
b 0.0 0
j -90.0 45.0 90.0 45.0 0.0 50
q
```

Source Code Listing

The source code *controller.c* is attached to the back of this report.