Control of a mobile Robot: Part I

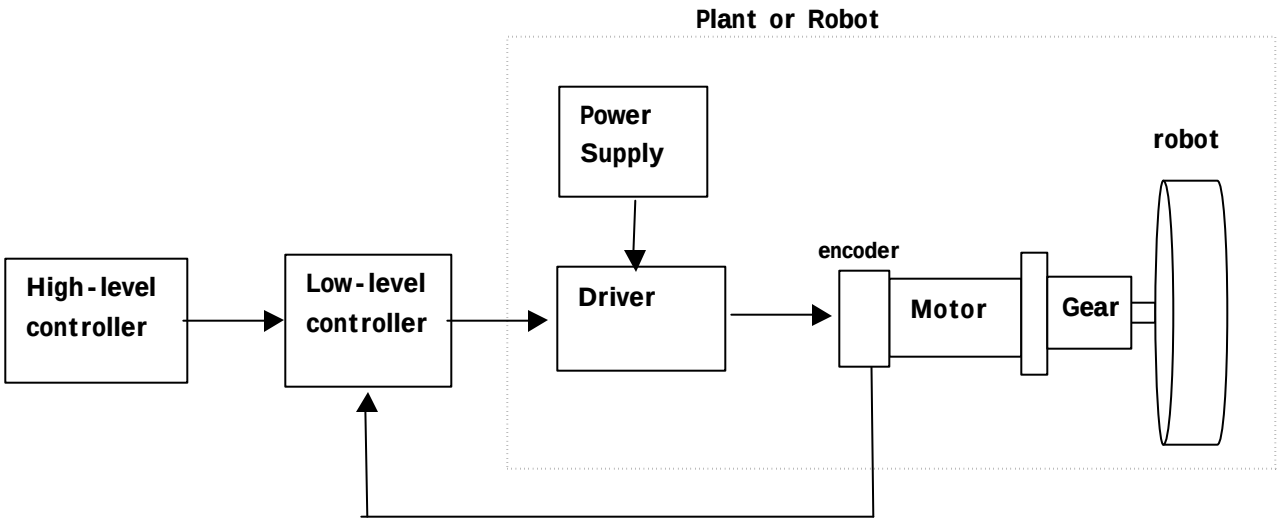
Introduction:

This tutorial will discuss the basic elements in the control of a mobile robot. The process will then be demonstrated through implementation of a follow-the-leader control on a lab robot. This tutorial will provide an overview of the critical system elements, a discussion of system modeling, modeling and design of a controller, and finally implementation and testing on the lab robot. This tutorial is designed to accompany labs 11-14 of the TTU ME3060 course.

Outline:

1. Elements of a motion-control system

The primary motion control elements in our robot system are shown in the figure below. This diagram shows the system divided into three parts; the robot system or plant, a low-level controller, and a high-level controller. The robot system includes the robot motors, mass and inertia components, the motor drivers and power supply. The low-level controller in our example will consist of a microcontroller (Motorola HC12) embedded on the robot capable of controlling two drive channels, left & right wheels for a differential drive or drive and steering for swivel wheel steer. The high-level control provides the interface between the robot and human operator and in our case resides on a lab PC with wireless communication to the low-level controller. In some cases the high-level controller will perform advanced computations such as inverse kinematic solutions, mapping, localization with GPS, etc.



2. Overview of motors

While there are many components in this robot system, a brief discussion of the drive motors and their selection will be provided since this selection influences the model of the electro-mechanical interaction in the system. The primary motors used in robot (or other) systems are listed in Table I, followed by some general comments related to applicability.

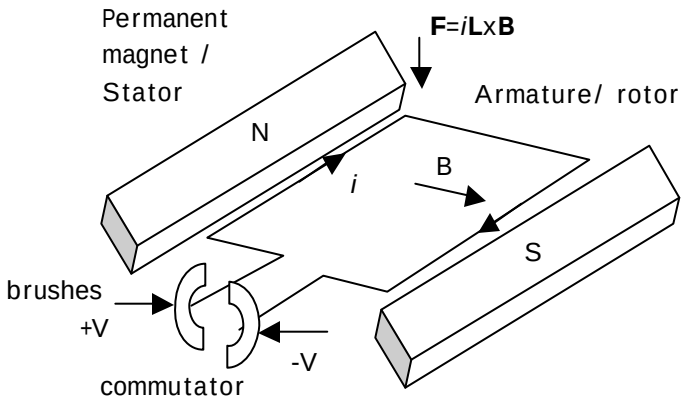
Brushed, PM DC Motors	**Stepper Motors**	**Brushless DC Motors**	**Brushless AC Motors**
Cheap, rugged, high-reliability	Cheap, rugged, high-reliability	No brushes, suitable for any environment	No brushes, suitable for any environment
Low Torque ripple	No brushes, suitable for any environment	Historically available in lower power ratings	Most commonly used in current industrial robots
Encoder + controller + feedback loop required to servo	Does not require feedback for position/velocity control		Requires power conditioning electronics affecting cost
Widely available, both commercial and surplus	At low speeds, provide up to 5x torque of brushed motor, 2x torque of brushless motor		
	Suffers from resonance and long settling times		
	Consume current regardless of load or motion, run hot		
	Losses at speed are high		

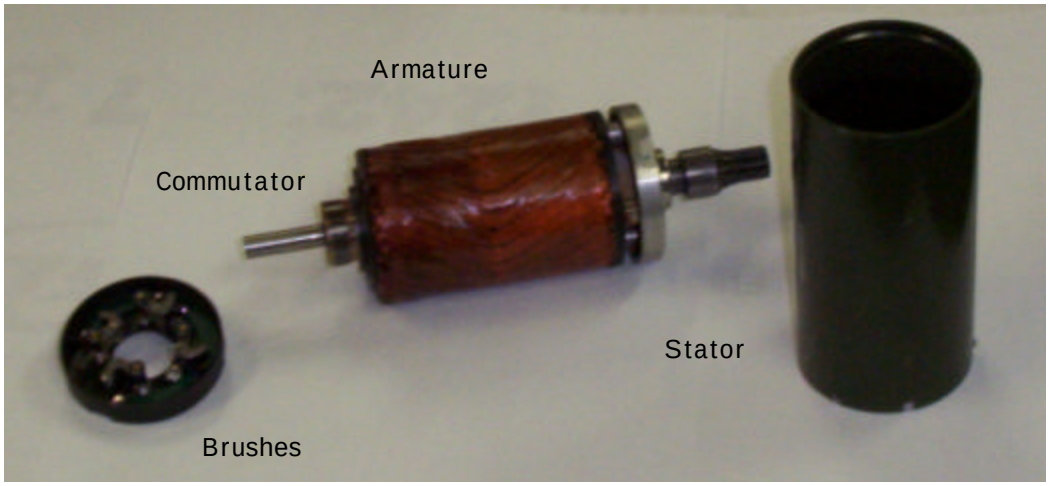
	Undetected position loss as a result of open loop operation		

The term servo motor or "servo" implies position and/or velocity control. In order to create a servo motor based on a PM DC motor, the system requires a motor + encoder + feedback loop + controller. However, once this expense is incurred the servo system becomes a highly robust system. Therefore, the robot system considered here will be based on PM DC motors. A brief review of PM DC motors will follow. For more information on the technical and practical aspects of implementing all the motors in the table above, please refer to the tutorials on the ME 4370 (Mechatronics) website.

3. Overview of the Brushed PM DC Motor:

The iron-core brushed DC motor historically has been one of the most commonly used motors in servo-motor systems. As its name implies, this motor contains brushes, permanent magnets, and operates on DC current. The motor is often simply called a DC motor. The primary components of this motor are shown in the following simple schematic and photograph.





3.1 PM DC Motor

The permanent magnets of the stator are attached to the motor walls, while the armature is part of the motor shaft. Current flow through the armature winding interacts with the magnetic field created by the stator magnets to create a couple or net torque on the armature. The armature contains many windings (many more than the one shown in this figure), and as the winding shown moves out of the field, another winding moves in. Therefore, this motor generates an output with fairly low torque ripple. The commutators provide a mechanical means to sense the position of the motor and send current to a new set of windings.

Based on this analysis, there are just a few factors the largely dictate the output torque and velocity (power) of this motor. Primary factors affecting torque and speed are;

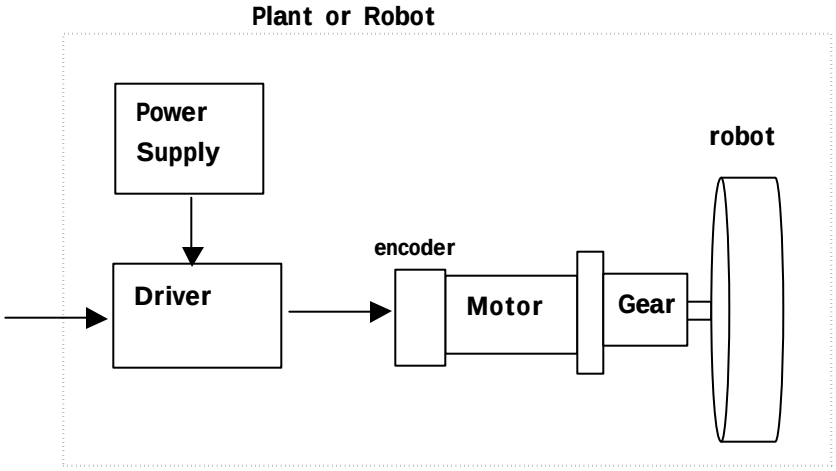
1) Amount of current flowing through the windings, a function of voltage potential and winding inductance and resistance.
2) Field strength created by the magnets
3) Radius of the motor armature
4) Back emf in the motor proportional to motor speed and magnetic field strength

As an additional note, brushed DC motors have two primary difficulties;

1) Contact between the brushes and commutator creates carbon dust, sparks and wear points
2) The armature generates heat as a function of the power loss equation i^2R , and this heat must escape through the magnets or motor shaft.

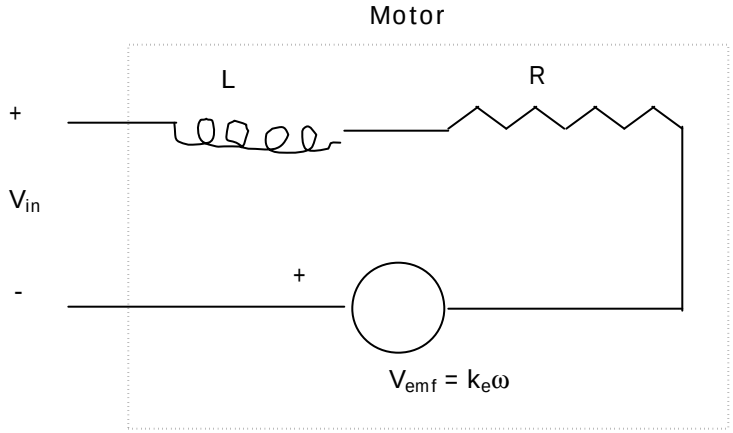
4. Modeling Electro-mechanical Systems based on PM DC Motor:

The goal is to design a representative model of the robot system or plant shown in figure 1. The robot model will be based on a system of equations that relate the input to the system (a command voltage) to the robot output parameters (robot motion or position, velocity and acceleration under a given load). This model will require; 1) a description of the current flow in the motor, 2) equations of motion for the robot with motor rotation the generalized coordinate, and 3) electrical/mechanical relations in the system. These three parts are described in sections 4.1 – 4.3.



4.1 Motor Electronics:

A circuit of the motor drive system is constructed and evaluated using Kirchoff's voltage law.



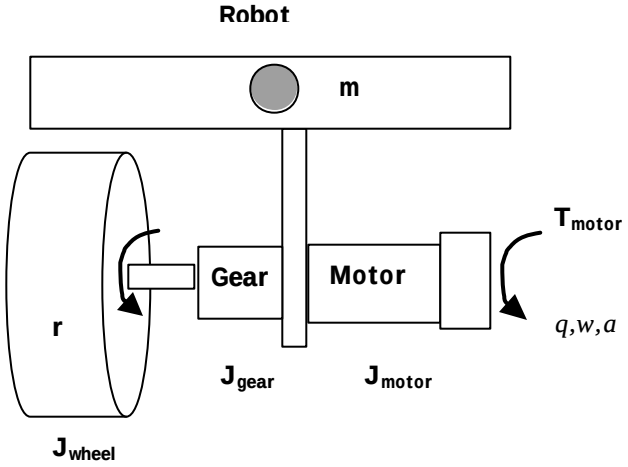
Summing voltages throughout the circuit gives

$$V_{in} = L\,di/dt + Ri + k_e w\,, \tag{1}$$

a first order ODE in current, i, with L, R the motor inductance and resistance, k_e the electric constant, V_{emf} the back emf and ω the rotational speed of the motor (rad/s).

4.2 Equations of Motion:

The equations of motion for the robot will consider the simple case of single-degree-of-freedom motion of the robot, moving forward and reverse. A free-body diagram of a symmetric half of the robot is constructed and used to write the equations of motion:



$$J\ddot{q} + C\dot{q} = T - T_{load} \tag{2}$$

where J is the equivalent inertia as seen by the motor, C is the equivalent viscous damping seen by the motor, T_{motor} is the input torque from the motor and T_{load} all other loads on the system. The total inertia of the system as seen by the motor is given by the following equation,

$$J_{equiv} = J_{motor} + J_{gear} + \left(J_{wheel} + mr^2\right)\left(\frac{1}{GR}\right)^2 \tag{3}$$

where J_{motor}, J_{gear} are the inertias of the motor and gear train relative to the motor, J_{wheel} the robot wheel inertia, m the robot mass, r the wheel radius and GR the transmission ratio expressed as the ratio of input rotation to unit output rotation. Equation 2 gives a second order ODE in motor rotation, q.

4.3 Electro-Mechanical Relation

The electrical and mechanical components are coupled in two ways. First, an approximate relation is generally used that describes motor torque as a linear function of current in the motor,

$$T = k_t i \tag{4}$$

where k_t the motor torque constant. In addition, the back emf in the motor is linearly related to the motor rotational velocity,

$$V_{emf} = k_e \dot{q} \tag{5}$$

4.4 Combined system model

The electrical and dynamic relationships are now combined to result in a system of equations that govern the robot response. Equations 4 and 5 are substituted into Eqs. 2 and 1 respectively to result in the final system equations; a 1st and 2nd order ODE with two unknowns, i and q:

$$J\ddot{q} + C\dot{q} - k_t i = -T_{load} \tag{6}$$

$$L\dot{i} + Ri + k_e \dot{q} = V_{in}. \tag{7}$$

This system can be cast into state-space form to result in a system of three 1st order ODE's as,

$$\begin{aligned} x_1 &= q \\ x_2 &= \dot{q} \\ x_3 &= i \end{aligned} \qquad \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -C/J\, x_2 + k_t/J\, x_3 - T_{load} \\ \dot{x}_3 &= -k_b/L\, x_2 - R/L\, x_3 + V_{in}/L \end{aligned} \tag{8}$$

or;

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -C/J & k_t/J \\ 0 & -k_b/L & -R/L \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{Bmatrix} 0 \\ T_{load}/J \\ VR/L \end{Bmatrix}$$

$$\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{bmatrix} r/GR & 0 & 0 \\ 0 & r/GR & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} \tag{9}$$

following the traditional state-space form for linear systems,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{10}$$

where **x** is the state vector, **A** the state coefficient matrix, **B** the input coefficient matrix, **u** the input vector, **y** the output vector and **C** the linear transformation from state to output variables. In this case, the output variables are robot linear position and velocity, and motor current.

Matlab provides a variety of tools to easily model this system. A simple first step could be to observe a step response of this system (step input of the motor voltage) using the command;
>step(A,B,C,D)
with A,B,C the matrices defined above. More discussion on system modeling will be provided in the simulink section.

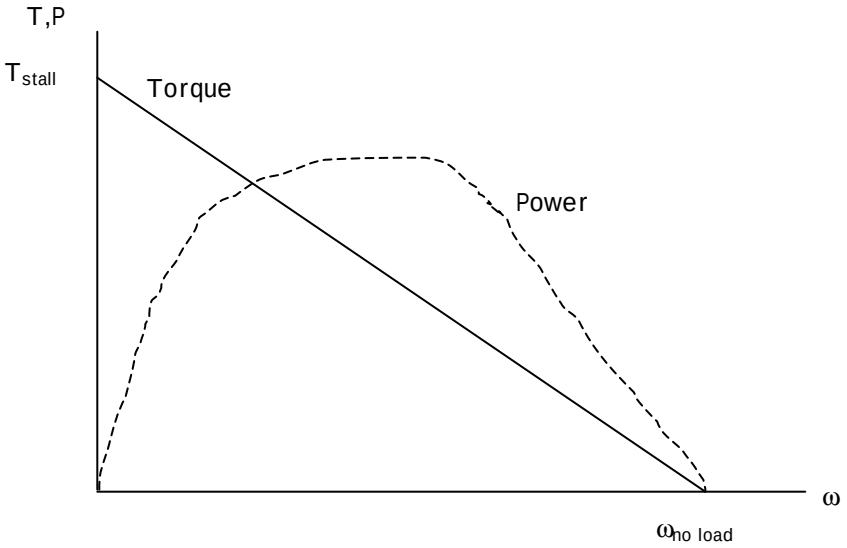
5. Steady-State DC Motor Behavior

At steady state, the dynamic motor model above can be greatly simplified ($\dot{i} = \ddot{q} = 0$) to yield the equations;

$$Ri + k_e w = V_{in} \quad + \quad T = k_t i \rightarrow \tag{11}$$

$$T = \left(R / k_t \right) V_{in} - \left(k_e k_t / R \right) w \,, \tag{12}$$

This equation represents a linear torque/speed relation for a PM dc motor. The response (assuming a constant input applied voltage) looks like;



The maximum generated torque occurs at rest (stall), and decreases to zero at the maximum motor speed under no load. At this speed, the back-emf voltage in the motor is equal to the input voltage (minus a small amount to overcome inefficiencies in the motor).
To consider power in the motor, write a linear equation for T as a function of motor speed;

$$T(w) = T_s \left(1 - w / w_{\max} \right) \tag{13}$$

and then express power as;

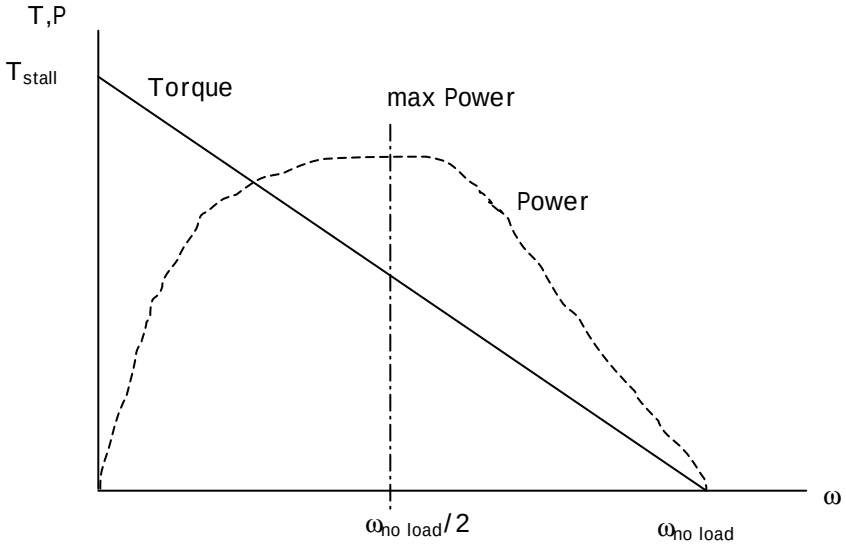
$$P(w) = T(w) = wT_s\left(1 - w/w_{\max}\right). \tag{14}$$

The maximum power occurs at;

$$dP(w)/dw = T_s\left(1 - 2w/w_{\max}\right) = 0 \tag{15}$$

or

$$w = \tfrac{1}{2} w_{\max} \tag{16}$$



The above information can be used in selecting a motor for your application, the first taking into account dynamic response, the second considering only steady-state behavior. More information can be found on this in the motor selection tutorial on the ME 4370 website.

6. Using Simulink to Model the Open-Loop System

Simulink provide a means to represent the system graphically in terms of a block diagram and then simulate the behavior of the system. Further, control loops can be easily added to this block diagram. The block diagrams are described in Laplace space, requiring Laplace transformations of the state equations. What follows is a derivation of the desired block diagram. The process starts with the state space equations derive above. Laplace Transforms of these equations are taken, and these Laplace equations are converted to block diagram form.

The state space equations derived earlier (ignoring the T_{losf} term) were,

$$\begin{aligned} x_1 &= q & \dot{x}_1 &= x_2 \\ x_2 &= \dot{q} & \dot{x}_2 &= -C/J\, x_2 + k_t/J\, x_3 \\ x_3 &= i & \dot{x}_3 &= -k_b/L\, x_2 - R/L\, x_3 + V_{in}/L \end{aligned} \tag{8}$$

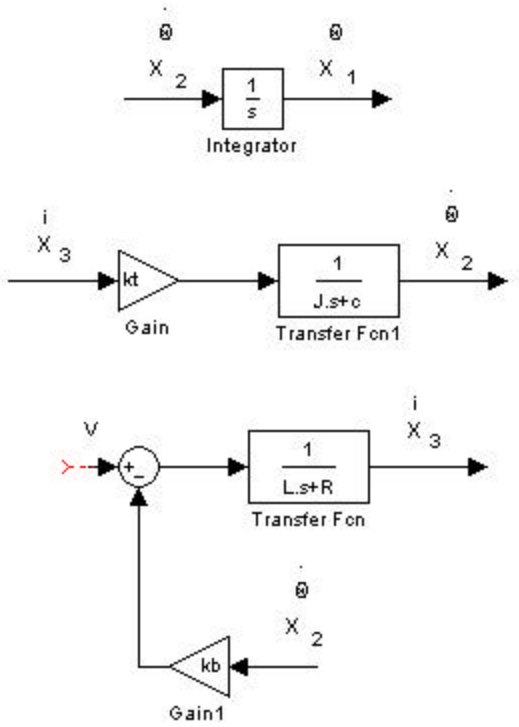
Taking the Laplace Transform of $\dot{x}_1$, $\dot{x}_2$, and $\dot{x}_3$ gives,

$$\begin{aligned} sx_1 &= x_2 \\ sx_2 &= -C/J\, x_2 + k_t/J\, x_3 \\ sx_3 &= -k_b/L\, x_2 - R/L\, x_3 + V_{in}/L \end{aligned} \tag{17}$$

Rearranging the above equations into an $\frac{output}{input}$ form yields (after some algebra),

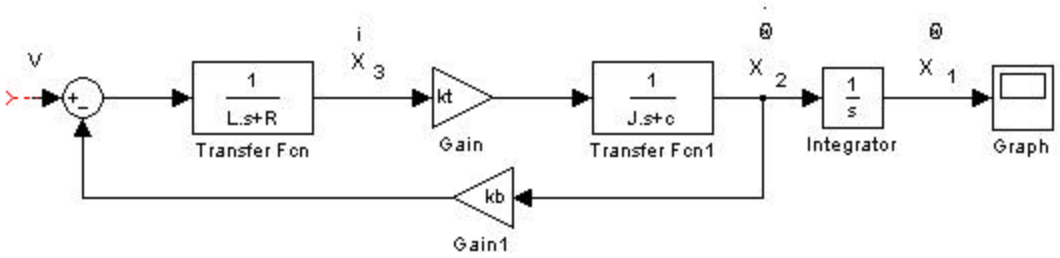
$$\begin{aligned} \frac{x_1}{x_2} &= \frac{1}{s} \\ \frac{x_2}{x_3} &= k_t \frac{1}{Js + C} \\ \frac{x_3}{V_{in} - k_b x_2} &= \frac{1}{Ls + R} \end{aligned} \tag{18}$$

These equations can now be represented as block diagrams as follows,



Note that data flows in the direction of the arrows, and the arrow pointing into the block represents "input", and the arrow pointing away from the block represents output. The triangular blocks represent "gain", which is a multiplication. The circular block in the third configuration is a "summer" block (summer blocks can also be represented as rectangles). In this instance, $+V - k_b x_2$ is the input into the Transfer Function. The Transfer Functions can even be thought of as "multipliers" of the input.

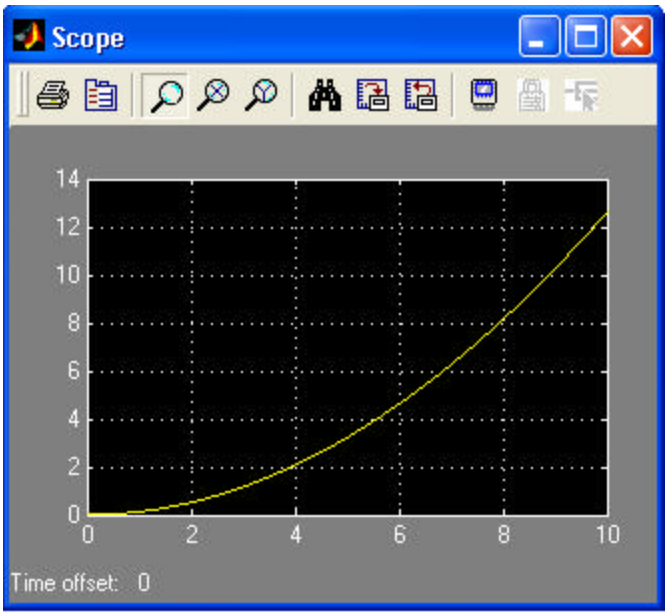
Connecting the above three block configurations so that the output of one will be the input of the next will give,



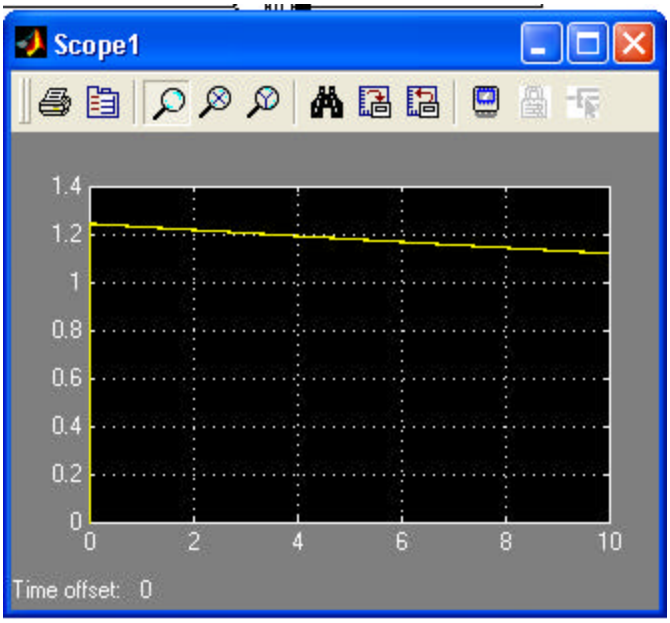
The above block diagram is the SIMULINK representation of our system (often called the "plant"). The input to this plant (robot) is a voltage signal and the output is the motor (wheel) rotation. The Transfer Function blocks are located in the Continuous library along with the Integrator block. Note that variables are used in the blocks. SIMULINK will use the values in the MATLAB Workspace assigned to those variables. Therefore, an M-file needs to be written to define these values. The M-file must be run before starting the simulation. Use the following parameters in your M-file:

```
m_robot=5;                          %Payload mass
g=9.81;                             %gravity
rw=.02;                             %Wheel radius
GR=1/30;                            %Gear ratio
J_robot=m_robot*rw^2*1.1;           %Robot inertia
J_motor=1.3e-4;                     %Motor inertia
J=J_motor+J_robot*(1/GR)^2;         %Total inertia at motor
c=1.0791e-5;                        %Viscous damping
kt=.415;                            % torque constant
kb=50.68*60/(2*pi*1000);            %Back emf constant
L=4.8*1e-3;                         %Inductance
R=9.65;                             %Resistance
V=12;                               %Input voltage
```

Initial conditions for the angular displacement can be defined by double-clicking the $\frac{1}{s}$ block. The state variable x_1 can be observed as a function of time using the scope block (labeled "graph" above). Using the parameters given above, a plot is obtained from the simulation of the angular displacement as a function of time. The length of the time vector used can be changed in SIMULINK's Simulation Parameters. Also, you may have to uncheck a Limit box in the Scope parameters to obtain the plot of all of your data points.



Angular Displacement vs. Time



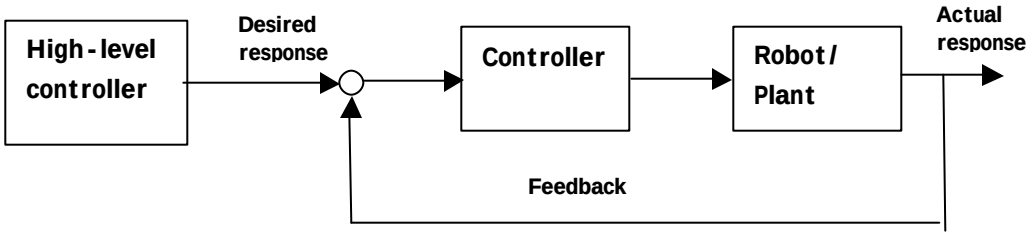
Current Draw vs. Time

Several observations can be made from the open-loop response of this robot system. First, the robot motion takes a few seconds to reach a steady speed (linear slope of the wheel angular displacement curve) and maximum speed of the robot can be seem from the state variable x_2. In addition, the robot system draws about 1.25 A peak and 1.1 A continuous in operation according to this model.

7. PID Control

Closed-loop control of a system provides the most common technique to maximize performance of engineered systems. A good online tutorial of this information can be

found at www.engin.umich.edu/group/ctm/ A controller added to our system is shown in the following figure.



The PID control technique provides a three term controller with gains kp, ki and kd where:

Kp = proportional gain
Ki = integral gain
Kd = derivative gain

These gains act on the error that occurs between the desired response and actual response to provide command information to the robot. The proportional gain is directly multiplied by the error while the derivative gain is multiplied by the derivative of the error and the integral gain is multiplied by the integral of the error as,

$$V = k_p e + k_d \dot{e} + k_i \int e dt$$

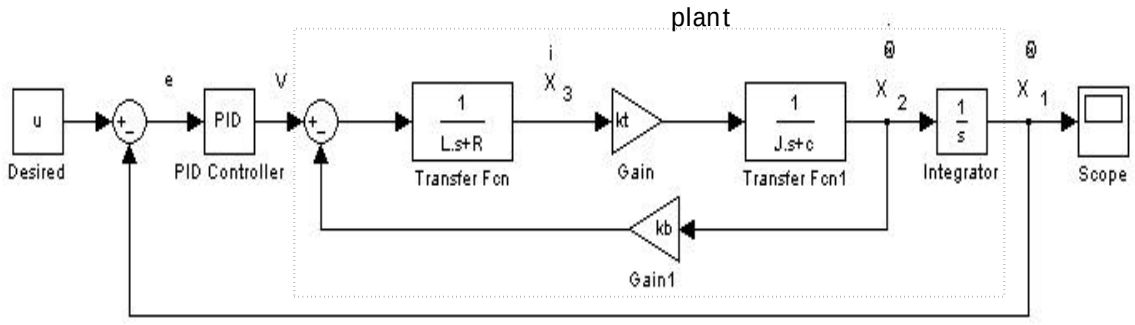
This process is carried out continuously, with the actual response moving toward the desired response (and the desired response altered by the high-level controller). It is desired that this process occur in real-time, which is generally considered to be in the range of 200 Hz for typical industrial manipulators. However, the requirements here depend greatly on the system.

Each of the three gains in the PID controller affects different aspects of the system response. For example, the proportional gain will shorten the system rise time, but will also increase the system overshoot or oscillation about the desired response. The derivative gain provides damping to the system, in generally decreasing any overshoot or oscillation in the system. Finally, the integral gain is intended to primarily eliminate any steady-state error in the system.

There are many formal means to select values for the kp, ki and kd gains in a PID controller. There is also the approach of trial and error. With a good system model and tools such as Matlab, trial and error is not an unreasonable approach.

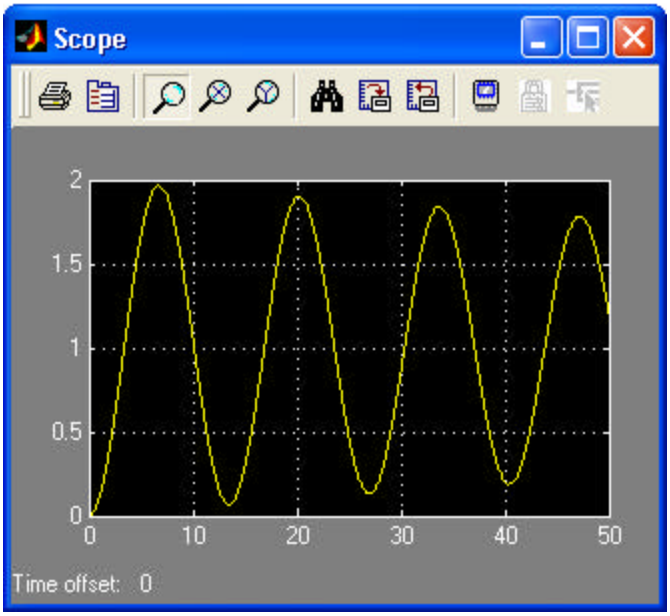
8. Using Simulink to Add PID Control to the System

Next, a closed-loop controller will be added to the system. This controller will be a three-term, PID controller with a unity feed-back loop. The controller is located prior to the plant and following the sum of desired reference and feedback. This can be represented in block diagram form as:



Here, u represents the desired response. The PID Controller can be found in the SIMULINK Extras – Additional Linear library. The first summer block takes the difference of our desired response u (here a constant) and our actual response from the plant x_1 or *?* to give an error signal e. It is obviously desired that the error term e be as close to zero as possible. The PID controller will take the error term e and adjust the voltage accordingly in order to achieve a smaller error. Double-clicking the PID block, allows change of the PID parameters. Change the values to the variables k_p, k_i, and k_d respectively. Use your previous M-file to change the value of these variables as well. The desired u value was set to a constant 1 also in the M-file.

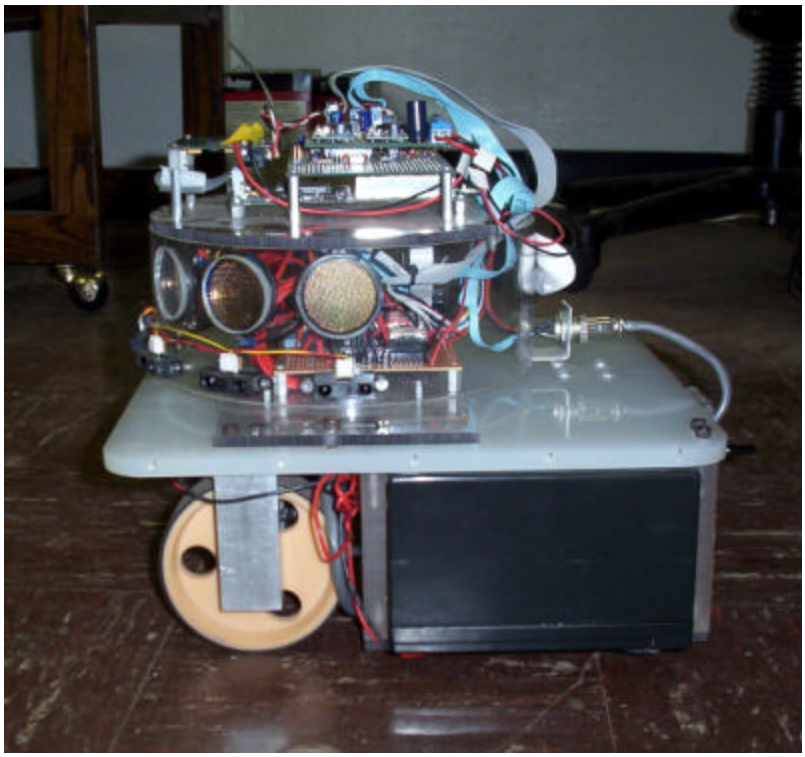
The scope below shows response with the PID controller (k_p=10, k_i=0 , and k_d=0). The desired u value was set to a constant 1 also in the M-file.

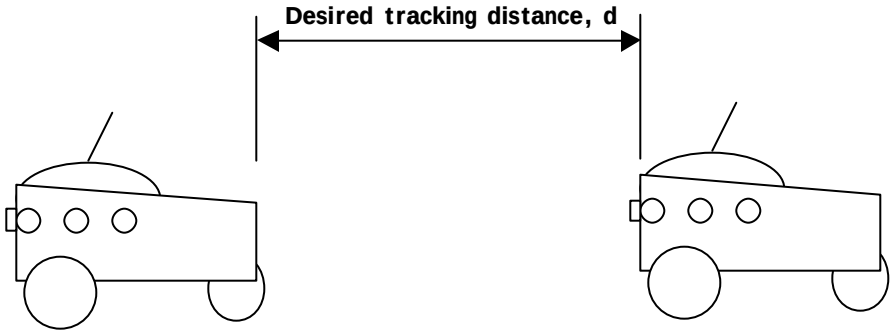


Angular Displacement vs. Time with PID Control

9. Implementation

The modeling and controller design process will now be demonstrated on one of the TTU lab robots. This robot is shown in the following figure. In this example, the goal is to design a controller to make the robot track or follow another robot in one dimension (figure 2)





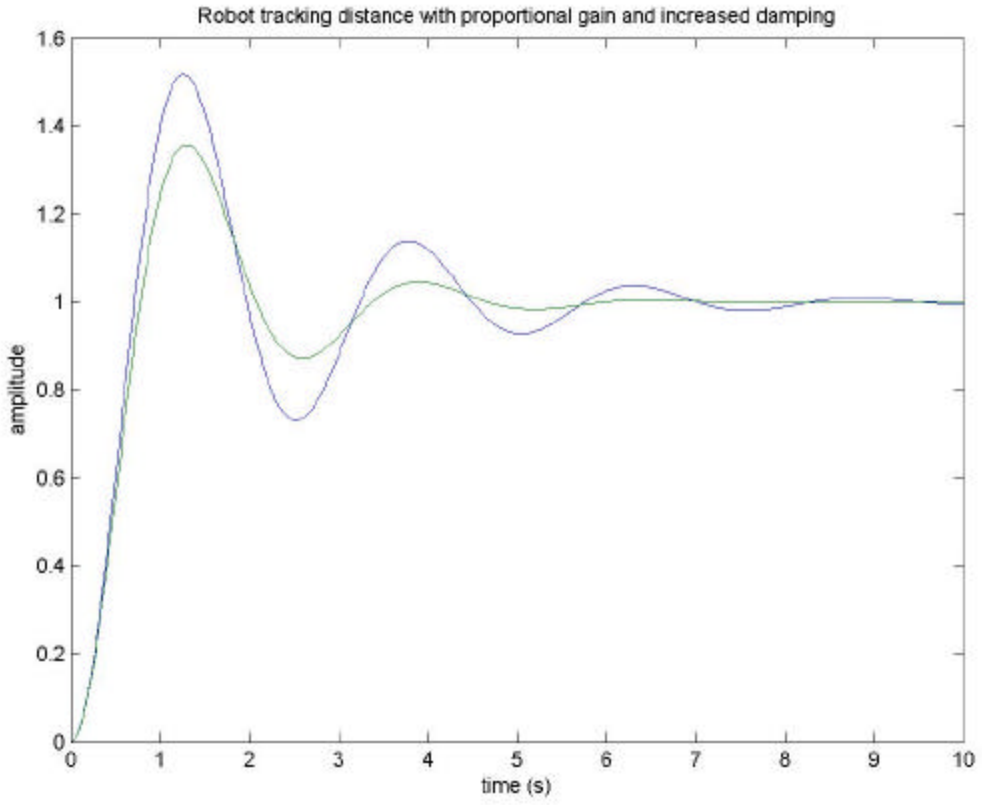
The desired tracking distance, d_d will be specified for the system. The actual tracking distance will be measured using sensor s1 on the tracking robot. This sensor is an analog

8-bit IR ranger (Sharp GP115). Additional pertinent information on the lab robot is included in the table below:

Component	**Details**
Embedded controller	HC12
Sensors	Analog, 8bit IR sensors
Robot Mass	5 kg
Wheels	4 cm dia
Motor:	Kohl-

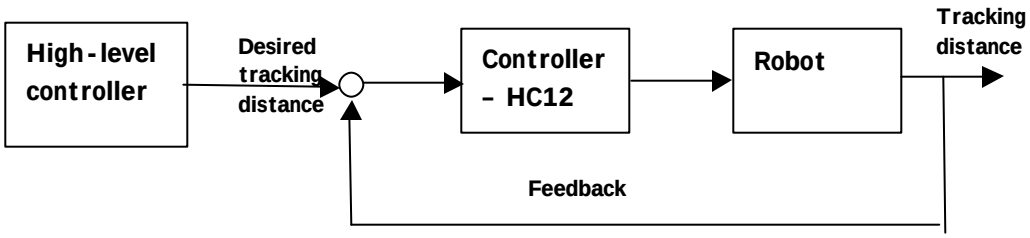
9.1: Model of system + controller

A model of the system + controller is first implemented. This procedure is based on the simulink model developed in section 6 above, with the parameters from table III implemented. A trial-and-error process is used to find a suitable set of controller gains. The controller gains and resulting system performance are shown below.

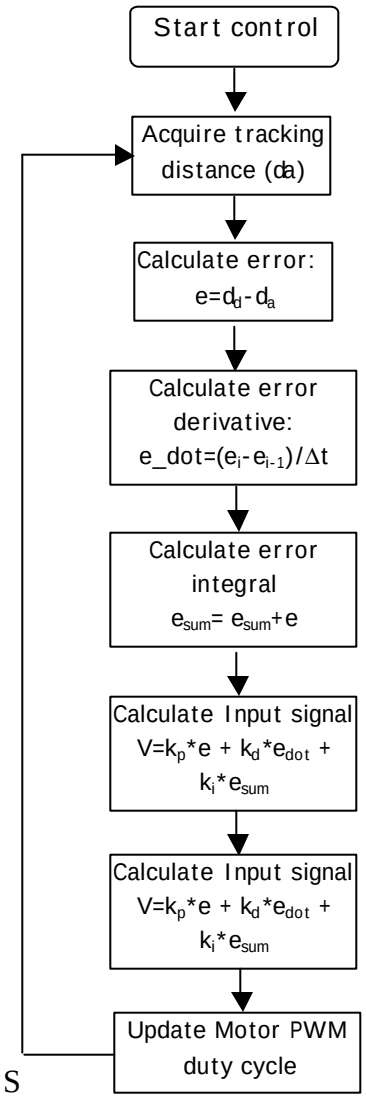


9.2: Implement controller on robot platform.

The controller designed in section 8.1 is used as a starting point for the robot system. The block diagram for our control system is shown as,



The control is implemented inside the HC12 controller based on the following flowchart:



Experiment with the lab robot. To do this, run the matlab file, gains.m. Enter the desired values of gains kp, ki and kd in gains.m. Turn on the lab robot, and run gains.m. The robot will wait to receive the three gains and then will begin the tracking process using those gains in its PID controller. The gains program in matlab will continue to acquire the real-time tracking distance for approximately one minute. At the end of this time, the tracking distance can be plotted to observe the controllers performance. Modify the gains as desired and continue to run until the robot performance is optimized.

Two comments about this system are noted;

1) The controller gains can quickly over-saturate the range of motor response. Thus, the linear controller actually plateaus at the input limits (100 % duty cycle).
2) There are a few lags in the motor response which affects the control performance.