Control Applications using Arduino

Debraj Chakraborty, Kishore Chatterjee, B.G. Fernandes Joseph John, P.C. Pandey, Dinesh Sharma, Narendra S. Shiradkar and Kushal Tuckley

Application Development with Arduino

Application development using Arduino boards involves:

1. Choice of appropriate additional hardware,
2. Development of required algorithms
3. Implementation of algorithms in software

Many of the hardware techniques such as negative feedback, hysteresis using Schmitt triggers etc. have their counterparts in software.

We'll review some of these briefly.

Application Development with Arduino

To illustrate some of the algorithms used for application development, we'll use temperature control of a water bath as an example.

We would like to keep the temperature of water in the bath as close as possible to a given temperature (called **set point**) which is higher than the room temperature (no cooling required) and which should be programmable.

Let us first look at the hardware required for this:

- We obviously need a heater. To control the temperature, we'll require some means of controlling power to the heater. This can be done through
  1. ON/OFF control through a relay, or
  2. Pulse Width control through thyristors driven from Arduino, or
  3. Voltage control through a programmable power source to the heater.
- Accordingly, we shall need drivers for relays/thyristors/DC source which will be driven using digital signals from Arduino.

Application Development: ON/OFF control

We also require hardware for measuring the actual temperature (using LM35 temperature sensor or a thermocouple), and means for adjusting the set point (through a potentiometer or dialing it in through a keyboard).

Let us first consider the simplest option – that of ON/OFF control.

- We can measure the temperature at regular intervals and compare it with the set point. If the actual temperature is higher than the set point, we turn the relay OFF, if it is lower, we turn it ON.
- When the temperature is close to the set point, a small amount of heating takes it above the set point which turns the relay OFF. However, then the temperature quickly drops below the set point, which turns the relay ON. This causes the relay to "chatter".

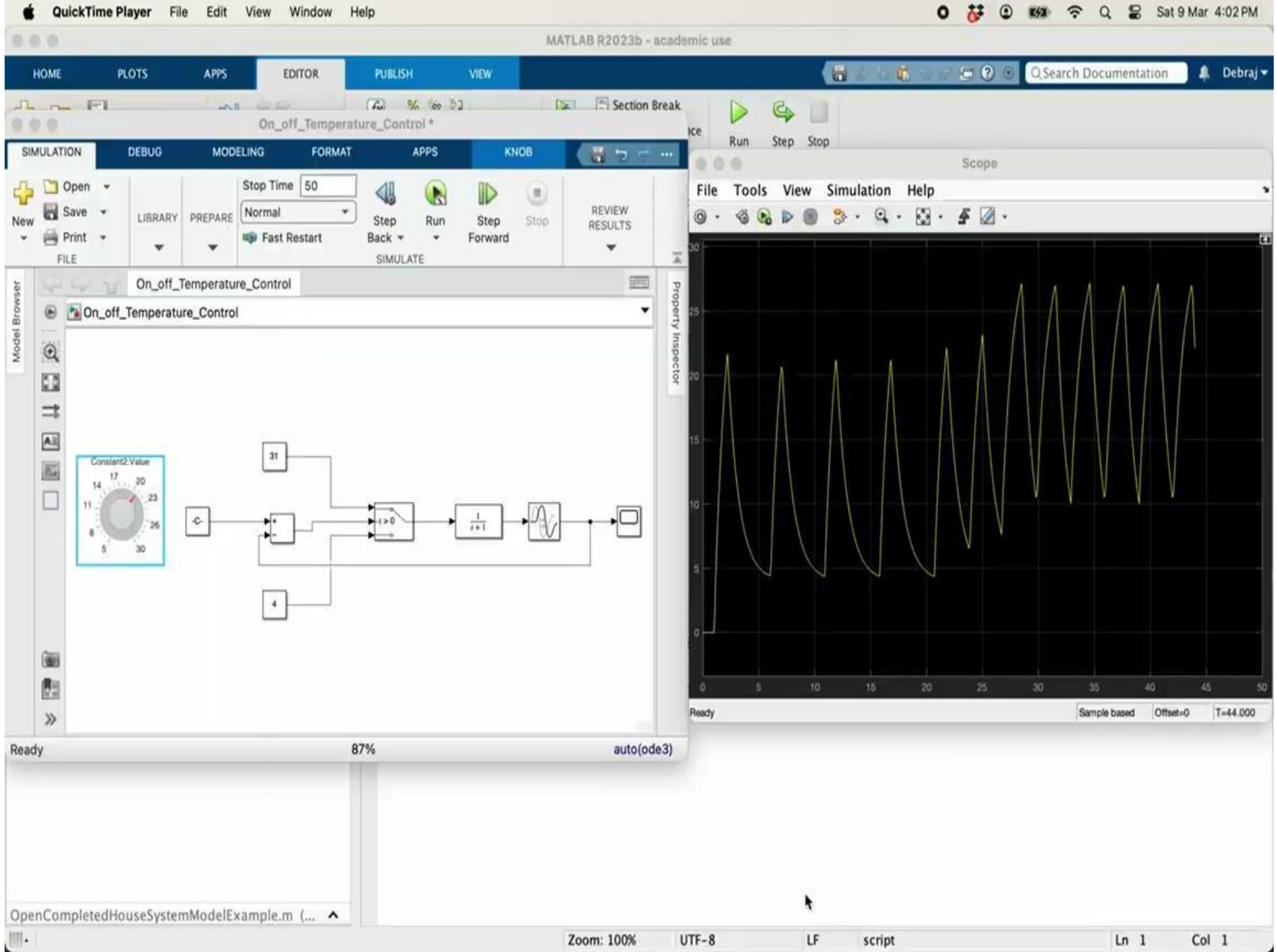
How to stop the relay from chattering due to this frequent switching?

Application Development: ON/OFF control

To prevent the relay from chattering, we can use a small amount of hysteresis in the decision to turn the relay ON or OFF.

- Instead of comparing the temperature with a single set point, we use a “high limit” and a “low limit” on either side of the set point.
- We turn the relay ON only if the temperature is below the “low limit”. We turn if OFF only if the temperature is above the “high limit”.
- We now have a trade off – if the high and low limit are too close to the set point, the relay will turn ON and OFF frequently.
- If these limits are set far from the set point, the temperature will ramp between these two limits and the worst case error between the actual temperature and the set point will be high.

You would have noticed this kind of control in many electric irons.


QuickTime Player File Edit View Window Help
Sat 9 Mar 4:02 PM
MATLAB R2023b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
Search Documentation
Debraj
Section Break
Run Step Stop
On_off_Temperature_Control *
SIMULATION DEBUG MODELING FORMAT APPS KNOB
New Open Save Print FILE
LIBRARY PREPARE
Stop Time 50
Normal
Fast Restart
Step Back Run Step Forward Stop SIMULATE
REVIEW RESULTS
Model Browser
On_off_Temperature_Control
On_off_Temperature_Control
Property Inspector
Constant2:Value
31
4
Ready 87% auto(ode3)
Scope
File Tools View Simulation Help
Ready Sample based Offset=0 T=44.000
OpenCompletedHouseSystemModelExample.m (...
Zoom: 100% UTF-8 LF script Ln 1 Col 1

Application Development: Proportional Control

A smoother way to control the temperature of the bath would be to apply power to the heater proportional to the error in temperature –

- The farther away we are below the set point, higher is the power applied.
- However as we approach the set point, the error becomes less and lower power will be applied.
- If we are at or above the set point, no power will be applied.

This results in more accurate control of temperature.

However, the power applied when we reach the set point is zero!
So the bath will always start cooling down due to heat losses as soon as it reaches the set point.

Proportionate control will always settle at non-zero error!
(A work around is to calculate the error from a point just above the set point, so that the power is non zero at the set point)

Application Development: Integral Control

We don't want to reduce the power to zero when we reach the set point. We want the heater to apply *constant* power, which keeps the bath at the set point.

- What operation gives a constant result when its argument reaches zero? The integral, of course!
- What we should do is to integrate the error and apply power proportional to this integral. Now when the error becomes zero, a constant power will be maintained which will keep the bath at the set temperature.
- If we overshoot the set point, error will become negative and the integral will reduce in value. With lower applied power, we'll come back to the set point and keep the power at this lower value.
- Since the proportional term is zero at the set point any way, we can use a value for applied power which is the weighted sum of proportional term and integral term. This is known as Proportional-Integral or P-I control.

Application Development: PI Control

The integration of error should be taken over how much time?

- Temperature error which was there long ago may be less relevant for determining the power to be applied at the current instant.
- Therefore we integrate the error over the recent history. Contribution due to error from long ago needs to be dropped out from the integral. This is known as the **reset rate**.
- Proportionality constants for the P and I contributions, as well as the reset rate have to be tuned for the specific system being controlled.

PI control works reasonably well for keeping the bath temperature constant. However, it is slow to react to sudden changes in the error. To take care of this, we add a third term – the differential term.

Application Development: PID Control

- PI control is very slow to react to sudden changes in error.
- For example, if we change the set point, suddenly there is a large error from the current temperature. PI control will be sluggish to react to it.
- To take care of sudden changes in the error, we need a term which will be proportional to the *rate of change* of the error. This can be provided by a term proportional to the differential of the error.
- When we include the differential term, the control strategy is called Proportional-Integral-Differential – or PID control.

PID control is widely used for controlling various process parameters.

How do we implement it in a micro-controller based system?

Implementing PID Control

- We measure the temperature at regular intervals, compute the error and store these values in an array.
- Every time we compute the current value of error, we add it to a moving sum and subtract the error which had been added “n” steps before. (“n” is the reset rate).
- We also compute the difference between the current and previous errors and use it as the differential term.
- We compute a weighted sum of the current error, current value of the moving sum and the differential term by multiplying each of these with their proportionality constants and adding them.
- We output this weighted sum to the external hardware which will apply power proportional to this value.

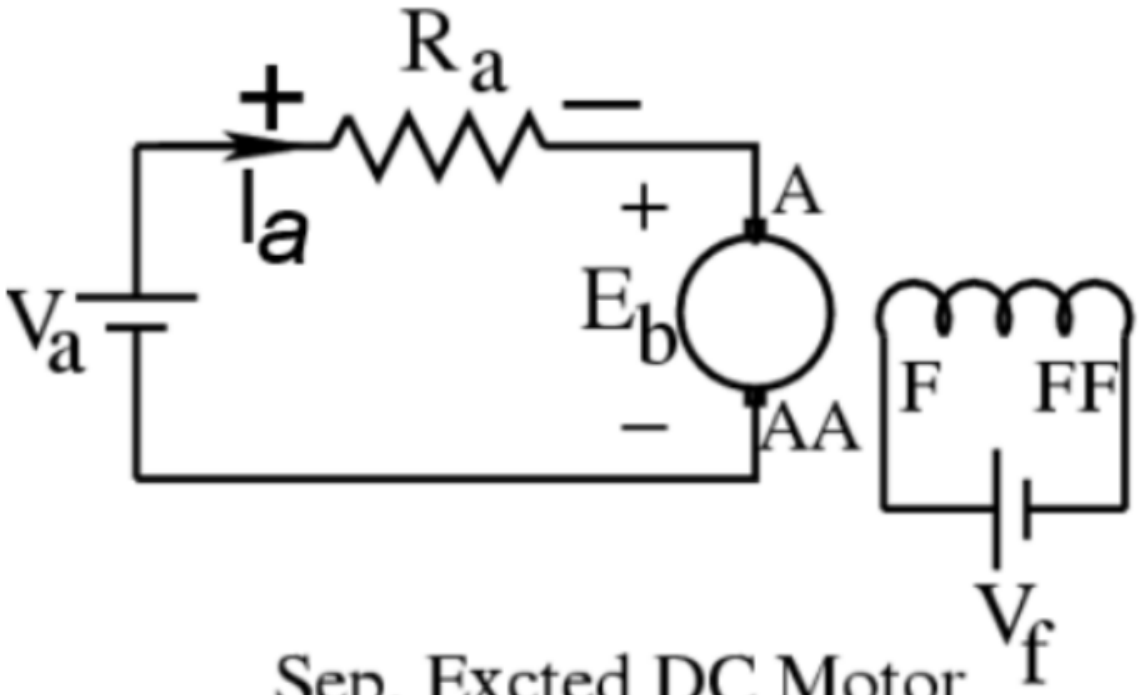
This algorithm is not restricted to temperature control. It can be applied to any parameter which has to be kept at a set value.
For example, it could be used for a line follower robot where we control the orientation to keep the robot centered on the line.

DC Motor Speed Control

Debraj Chakraborty, Kishore Chatterjee, B.G. Fernandes Joseph John, P.C. Pandey, Dinesh Sharma, Narendra S. Shiradkar and Kushal Tuckley

March 9, 2024

DC Motor Speed Control



Sep. Excted DC Motor

$$V_a = E_b + I_a R_a$$

$$E_b = K_e \Phi \omega$$

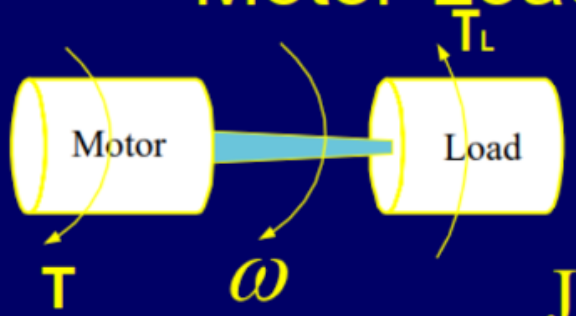
$$T = K_e \Phi I_a$$

$$\omega = \frac{V_a}{K_e \phi} - \frac{R_a}{K_e \phi} I_a$$

$$= \frac{V_a}{K_e \phi} - \frac{R_a}{(K_e \phi)^2} T$$

DC Motor with load (Slide from previous Lecture)

Motor-Load Interaction



$$T = T_L(\omega) + J\frac{d\omega}{dt} \quad \text{--- (A)}$$

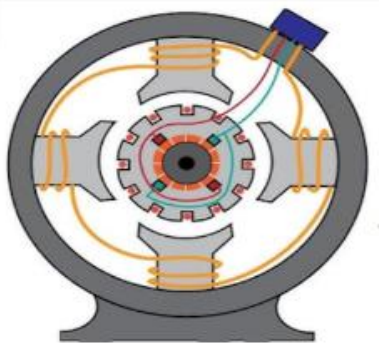
J = Polar moment of inertia

We have already derived:

$$\omega = \frac{V}{K_e \emptyset} - \frac{R_a}{(K_e \emptyset)^2} T \quad \text{---- (B)}$$

The equations (A) and (B) represent the model of a dc motor while driving a certain load having torque, $T_L(\omega)$

DC Motor with Load



We need to model the load torque T_L.

Assuming our load is,

$$T_L(t) = J_L \frac{\mathrm{d}\omega}{\mathrm{d}t} + D\omega$$

where D is the Viscous Damping due to air resistance, the resulting torque is assumed to grow linearly with ω .

Ignoring the in-between gears

$$T_m(t) = (J_m + J_L)\frac{\mathrm{d}\omega}{\mathrm{d}t} + D\omega$$

$$T_m(t) = J\frac{\mathrm{d}\omega}{\mathrm{d}t} + D\omega$$

Input Voltage vs Speed equation

$$R_a I_a(t) + E_b(t) = V_a(t)$$

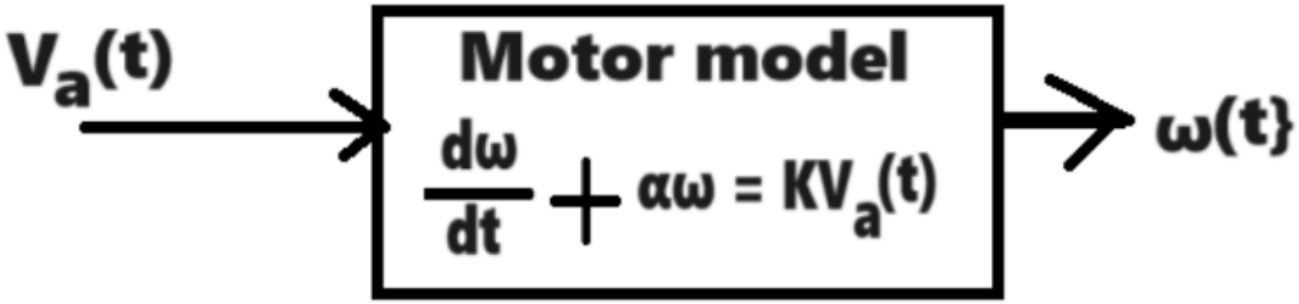
$$T_m(t) = K_t I_a(t) \; ; \quad E_b(t) = K_b \omega(t)$$

$$R_a \frac{T_m(t)}{K_t} + K_b \omega(t) = V_a(t)$$

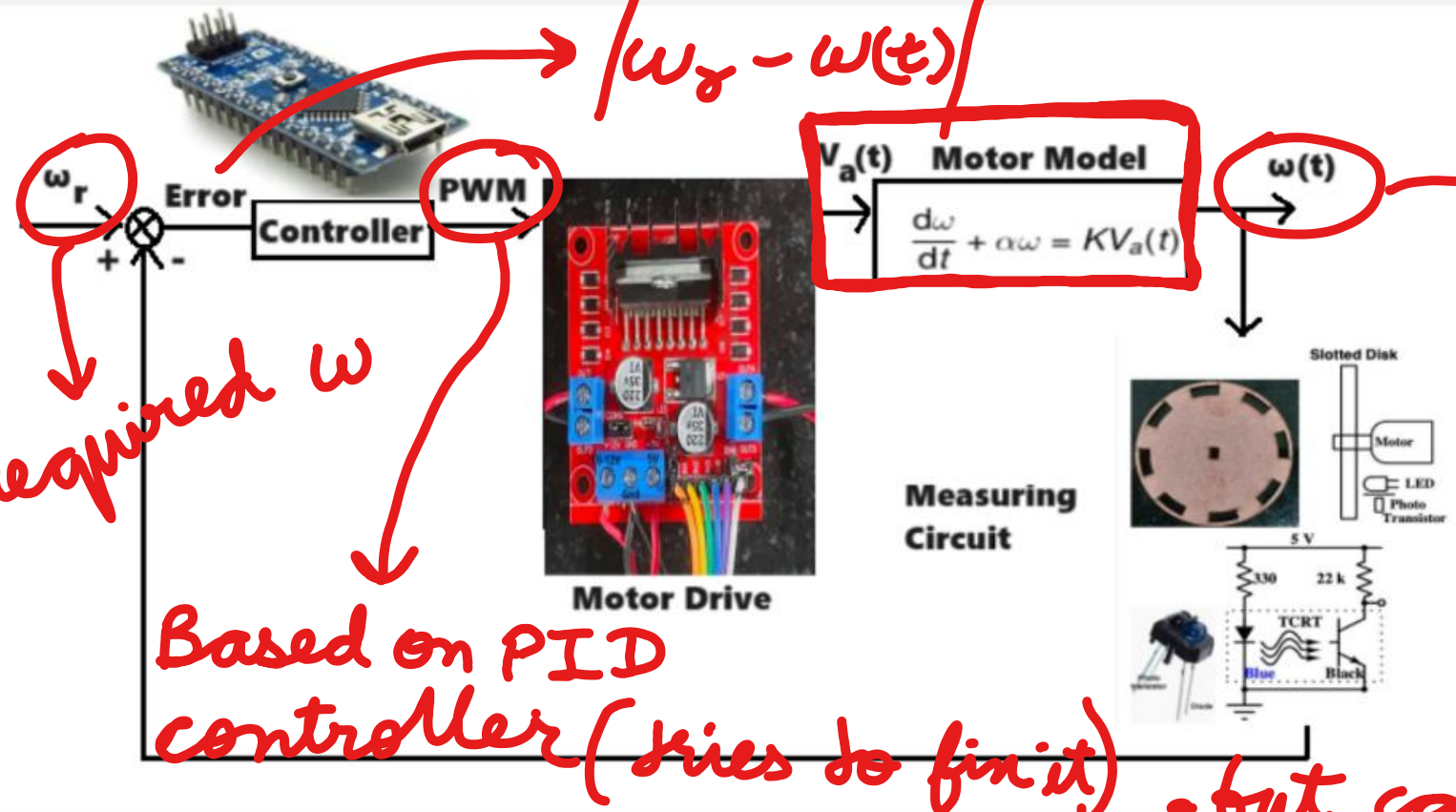
$$\frac{R_a}{K_t}\left[J\frac{\mathrm{d}\omega}{\mathrm{d}t} + D\omega\right] + K_b \omega(t) = V_a(t)$$

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} + \alpha\omega = K V_a(t)$$

$$\alpha = \left[\frac{D}{J} + \frac{K_t K_b}{R_a J}\right] \; ; \quad K = \frac{K_t}{R_a J}$$



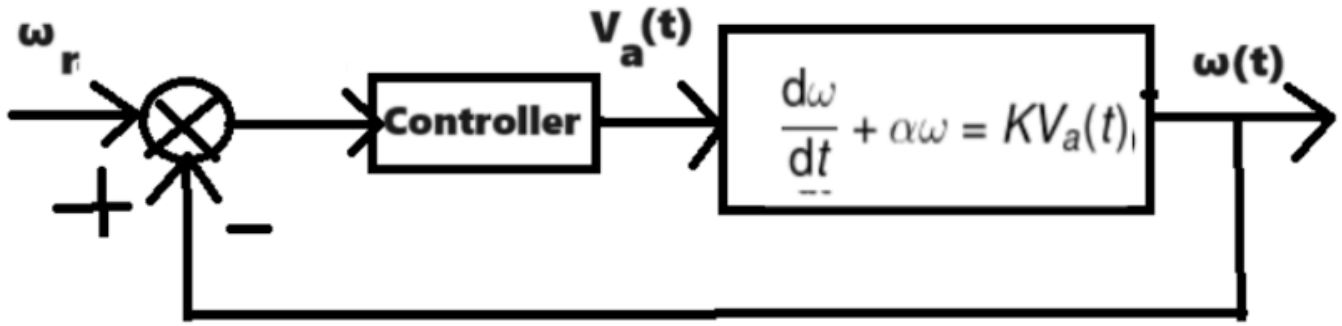
How do we automatically Control $\omega(t)$? - FEEDBACK



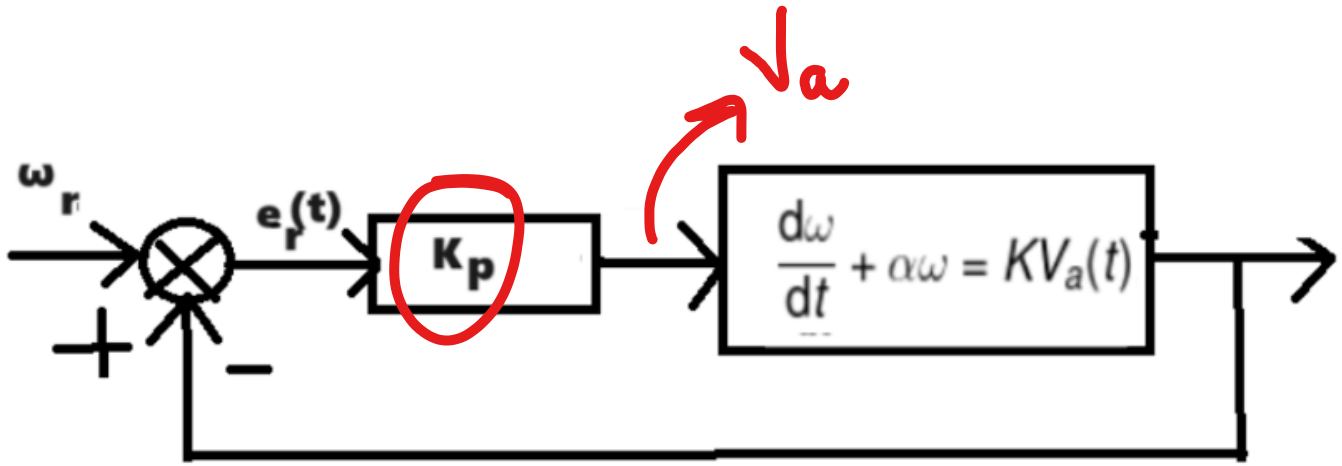
Feedback ⇒ Backbone of Automatic Control

1. Measure the current value of speed
2. Compare with the desired speed
3. Correct the speed; increase if less and decrease if more

Controller Structure? - Just a Gain



1st guess for controller $\Rightarrow$ Just a gain K_P



Steady State Error

Effect of this intervention

Error $e_r(t) = \omega_r - \omega(t)$ $V_a(t) = K_p e_r(t)$

Ideally, we would like $e_r(t) \to 0$ as $t \to \infty$

$\omega(t) = \omega_r - e_r(t)$; $\frac{d\omega}{dt} + \alpha\omega = KV_a(t)$

$$\Rightarrow \frac{d}{dt}[\omega_r - e_r(t)] + \alpha[\omega_r - e_r(t)] = KK_p e_r(t)$$

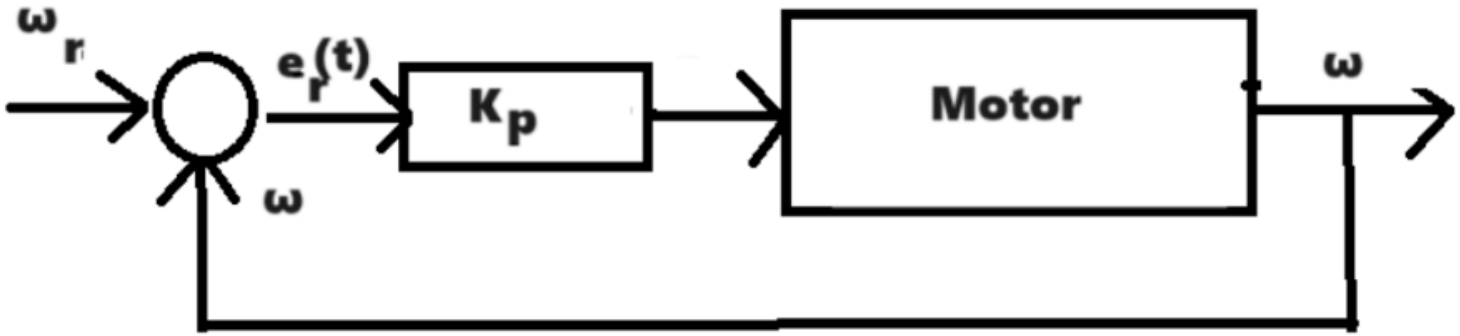
$$\Rightarrow \frac{d}{dt}e_r(t) + (\alpha + KK_p)e_r(t) = \alpha\omega_r$$

Solution: $e_r(t) = A + Be^{-(\alpha+KK_p)t}$

If you evaluate A and B, $A = \frac{\omega_r\alpha}{\alpha + KK_p}$

Clearly, $e_r(t) \nrightarrow 0$ as $t \to \infty$. $e_r(\infty) = \frac{\omega_r\alpha}{\alpha + KK_p} \neq 0$ is the steady state error

DC Motor Speed Control

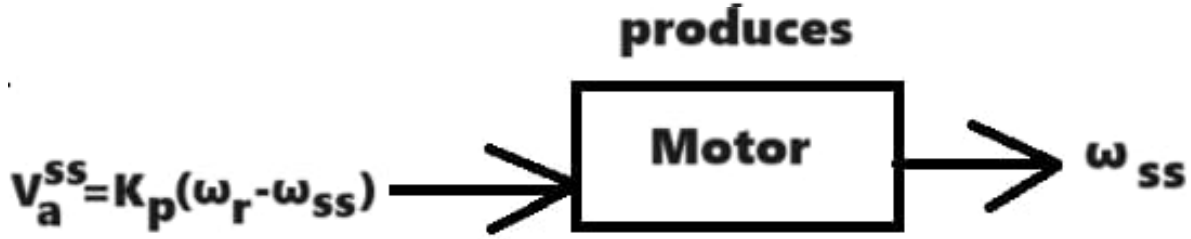


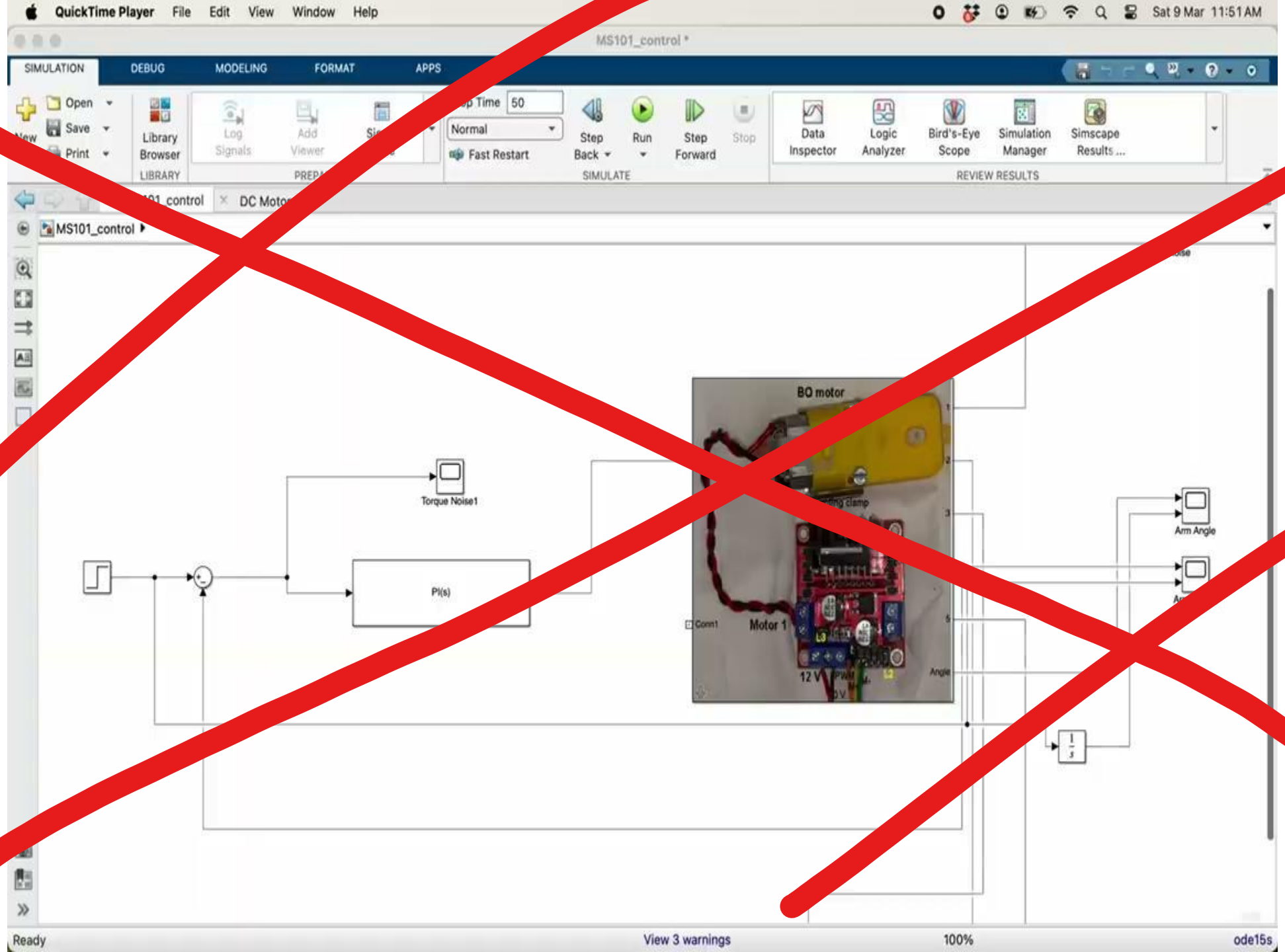
If $\omega_r = \omega, \quad e_r = 0,$

Motor has zero input voltage, $V_a(t) = K_p e_r \Rightarrow 0$

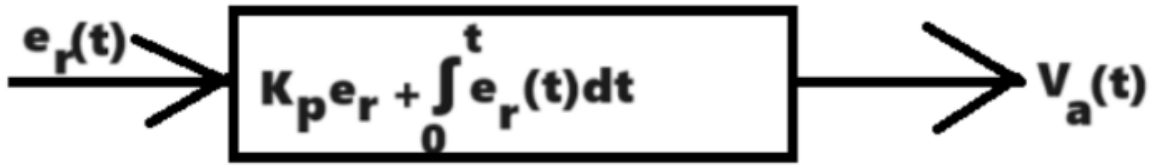
Hence instead of reaching the set speed

the system speed settles at ω_{ss}, where



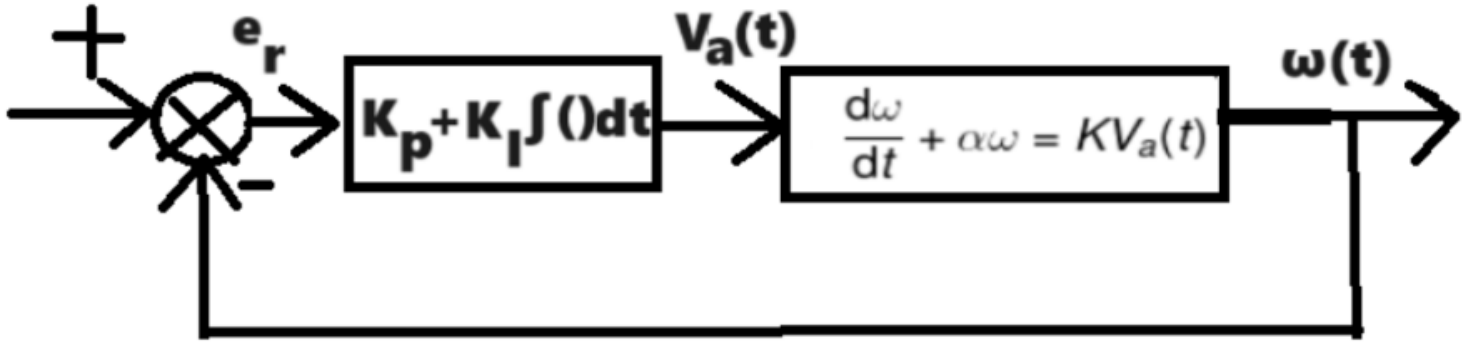

QuickTime Player File Edit View Window Help
Sat 9 Mar 11:51 AM
MS101_control *
SIMULATION
DEBUG
MODELING
FORMAT
APPS
Open
Save
Print
Library Browser
LIBRARY
Log Signals
Add Viewer
Stop Time
50
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
SIMULATE
Data Inspector
Logic Analyzer
Bird's-Eye Scope
Simulation Manager
Simscape Results ...
REVIEW RESULTS
MS101_control
Torque Noise1
PI(s)
BO motor
Motor 1
Conn1
12 V
Arm Angle
Angle
Ready
View 3 warnings
100%
ode15s

Solution to Non-zero SS Error: PI Controller



In this scheme $V_a(t)$ can be non-zero even when $e_r(t) = 0$.

Let's Calculate :



$$V_a(t) = K_p e_r + K_I \int_0^t e_r dt$$

$$e_r = \omega_{ref} - \omega$$

PI Controller

$$\frac{d}{dt}(\omega_{ref} - e_r) + \alpha(\omega_{ref} - e_r) = KK_p e_r + KK_I \int_0^t e_r \, dt$$

$$\frac{de_r}{dt} + (\alpha + KK_p)e_r + KK_I \int_0^t e_r \, dt = \alpha\omega_{ref}$$

$$\frac{d^2 e_r}{dt^2} + (\alpha + KK_p)\frac{de_r}{dt} + KK_I e_r = 0$$

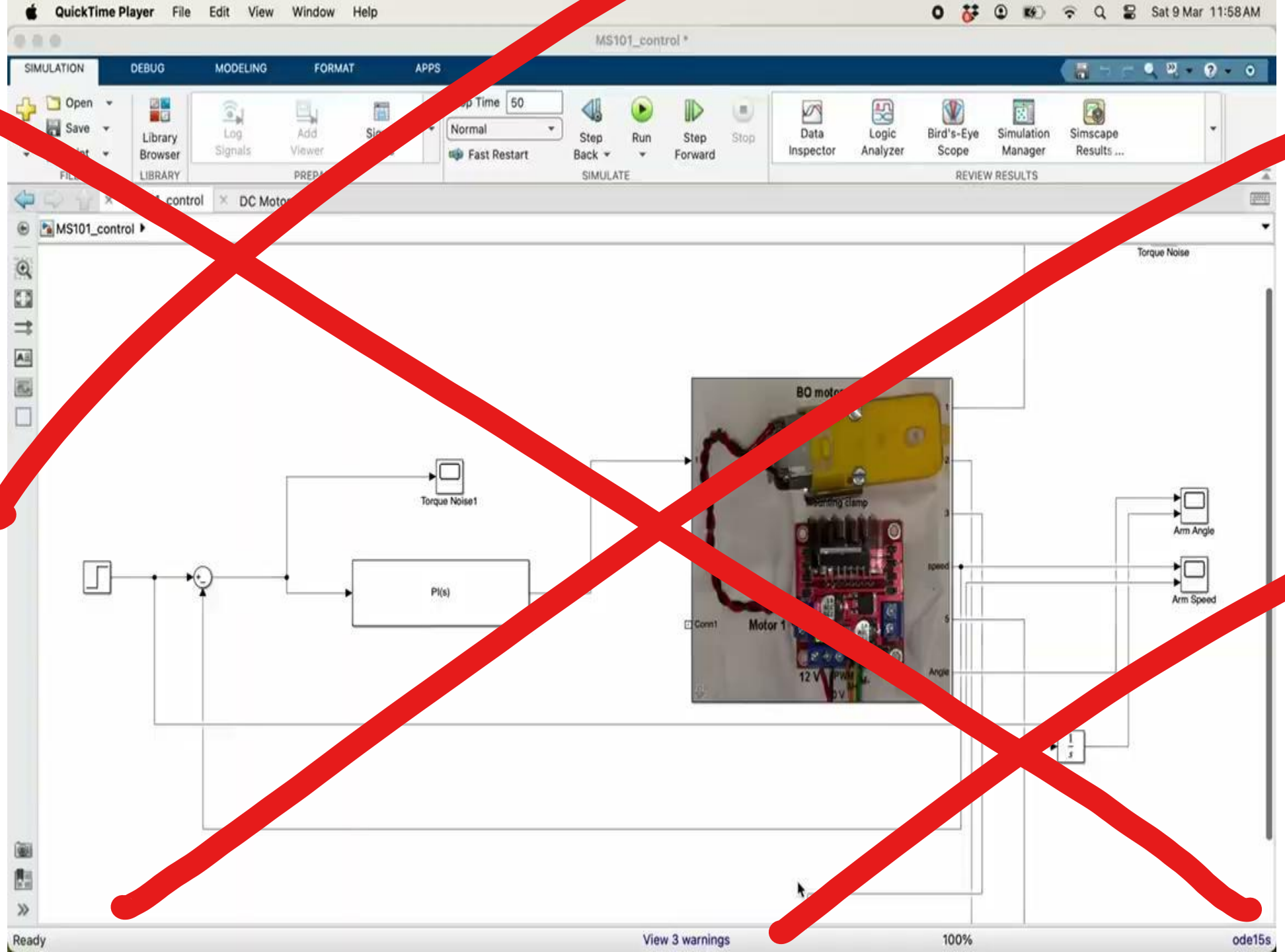
$$e_r(t) = Ae^{D_1 t} + Be^{D_2 t}$$

where D_1, D_2 are the roots of the quadratic equation:

$$x^2 + (\alpha + KK_p)x + KK_I = 0$$

Theorem

If D_1, D_2 have negative real parts $e_r(t) \to 0$ as $t \to \infty$.


QuickTime Player File Edit View Window Help
Sat 9 Mar 11:58 AM
MS101_control *
SIMULATION
DEBUG
MODELING
FORMAT
APPS
Open
Save
Library Browser
LIBRARY
Log Signals
Add Viewer
Stop Time
50
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
SIMULATE
Data Inspector
Logic Analyzer
Bird's-Eye Scope
Simulation Manager
Simscape Results ...
REVIEW RESULTS
MS101_control
Torque Noise
Torque Noise1
PI(s)
Motor 1
Arm Angle
Arm Speed
Ready
View 3 warnings
100%
ode15s

The End