ECE 3510 PID Tuning

A.Stolp a
3/24/20

For some time now we have been studying root-locus plots because these plots give us information about the closed-loop system response using only the open-loop transfer function and the system gain. We even extended the basic idea so that we can draw unconventional root-locus plots for variables other than gain. We also found that we could use the knowledge of how the open-loop poles and zeros affected the closed-loop response to design compensators which added new poles and zeros. The proportional-integral-differential (PID) compensator turned out to be especially useful and, not surprisingly, is also one of the most and common. All of this depended on knowing the open-loop transfer function. This makes it look like we can't get the benefits of a PID compensator (or controller) without the transfer function. Oh, how wrong you are. Smarter people than us found that you can put an adjustable PID in a feedback system and twiddle the knobs 'til you get the response you want. Even smarter people than them developed ways to get a good starting settings for the knobs and more systematic ways to twiddle from there. These methods are called "PID Tuning" and you should be aware of their existence.

Ziegler-Nichols PID Tuning Methods

Reaction-Curve Method Measurements are made on the **open-loop** system to determine controller parameters.
Can only be used:

1. Open-loop system is stable, and it's step response doesn't ring. (Typically worded as "doesn't have integrators or dominant complex-conjugate poles".)
2. The open-loop system (without any feedback), has a simple S-shaped unit step response like the one shown below. This curve is called "the reaction curve".

Measurements

Draw a tangent line at the inflection point of the curve.

Measure A, L, and T, as shown.

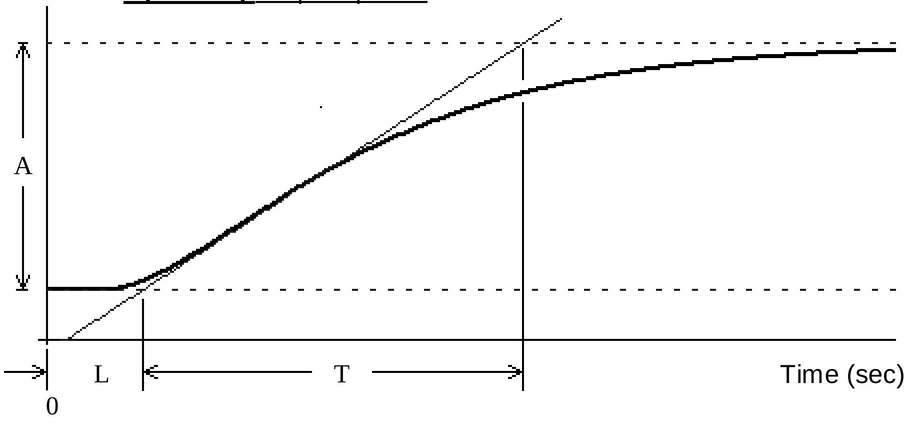
Calculate the slope at inflection point:

$$R = \frac{A}{T}$$

If the input is not a unit step (is a step of x_m instead of 1), modify R like this:

$$R = \frac{A}{x_m \cdot T}$$

Open-loop step response



The units of R should come out to be $\frac{1}{\text{sec}}$

Decide on what type of controller you would like to use. Proportional only, Proportional with Integral (PI) to eliminate steady-state error, OR Full PID to inprove dynamic response as well.

Type of Controller	Parameters of Controller (These are initial settings and may be subject to finer adjustments later)		
Proportional	$k_p = \frac{1}{R \cdot L}$	$k_i = 0$ k_i and k_d are both 0 because this is just proportional control	$k_d = 0$
PI	$k_p = \frac{0.9}{R \cdot L}$	$k_i = k_p \cdot \frac{0.3}{L} = \frac{0.27}{R \cdot L^2}$ OR $T_I = \frac{L}{0.3}$	$k_d = 0$
PID	$k_p = \frac{1.2}{R \cdot L}$	$k_i = \frac{k_p}{2 \cdot L} = \frac{0.6}{R \cdot L^2}$ OR $T_I = 2 \cdot L$	$k_d = \frac{k_p \cdot L}{2}$ OR $T_D = \frac{L}{2}$

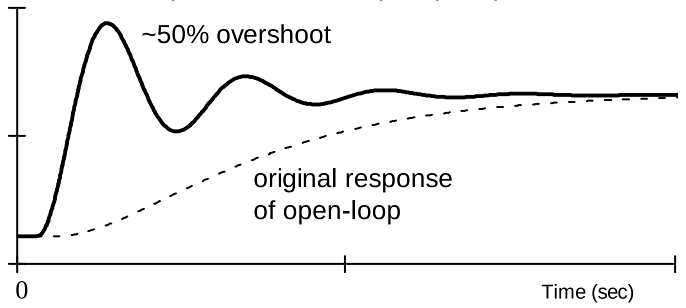
Controller transfer function

$$\mathbf{C}(s) = k_p \cdot \left(1 + \frac{1}{T_I \cdot s} + T_D \cdot s\right)$$ using T parameters (eq. 4.110 in other notes)

$$= k_p + \frac{k_p}{T_I \cdot s} + k_p \cdot T_D \cdot s$$

$$= k_p + \frac{k_i}{s} + s \cdot k_d$$ using k parameters, like we do in our class

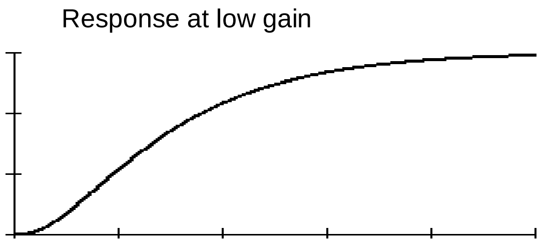
Expected closed-loop step response

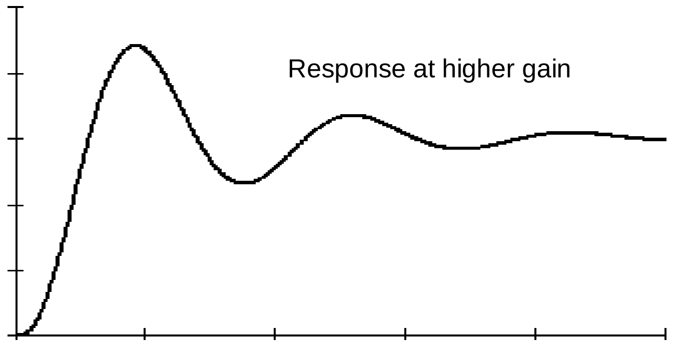


Ultimate-Sensitivity Method Measurements are made on the **closed-loop** system to determine controller parameters.

1. Can be used when the open-loop system is unstable, and requires feedback to be stable.
2. Use **only proportional gain** to make initial measurements.

Slowly turn up the gain while observing the step response.

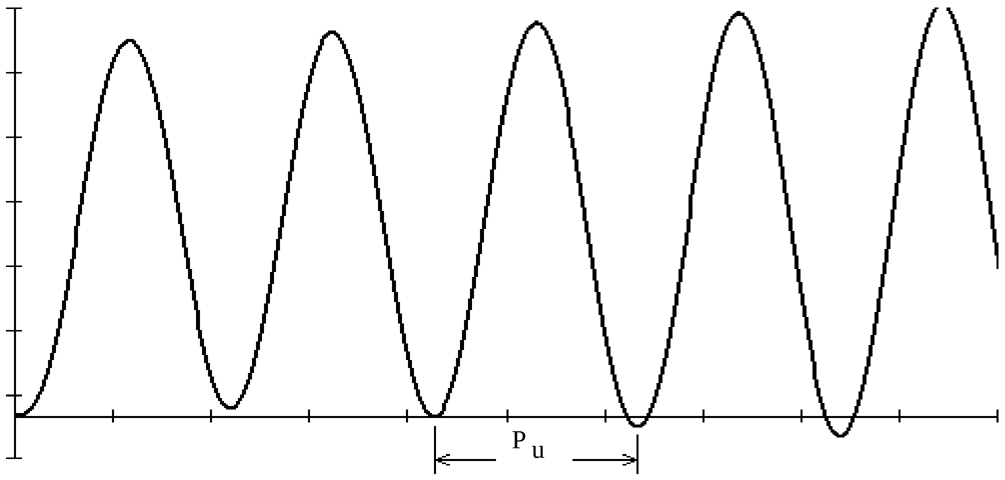




Response at the "ultimate gain" (K_U), when the system begins to oscillate.

Record the "ultimate gain" (K_U).

Measure the "ultimate period" (P_U), of the oscillation.



Decide on what type of controller you would like to use.

Type of Controller	Parameters of Controller (These are initial settings and may be subject to finer adjustments later)		
Proportional	$k_p = 0.5 \cdot K_u$	$k_i = 0$ k_i and k_d are both 0 because this is just proportional control	$k_d = 0$
PI	$k_p = 0.45 \cdot K_u$	$k_i = k_p \cdot \frac{1.2}{P_u}$ OR $T_I = \frac{P_u}{1.2}$	$k_d = 0$
PID	$k_p = 0.6 \cdot K_u$	$k_i = k_p \cdot \frac{2}{P_u}$ OR $T_I = \frac{P_u}{2}$	$k_d = k_p \cdot \frac{P_u}{8}$ OR $T_D = \frac{P_u}{8}$

The Ziegler-Nichols PID Tuning Methods usually result in systems that have quite a bit of overshoot, so you will probably want to make minor adjustments after the initial settings.

Effects of increasing a parameter independently

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
k_p	Decrease	Increase	Small change	Decrease	Degrade
k_i	Decrease	Increase	Increase	Eliminate faster	Degrade
k_d	Little effect	Decrease	Decrease	Little effect	Improve if k_d is small

More about PID Tuning: https://cdn.instructables.com/ORIG/FC1/NAZC/IVA51KF1/FC1NAZCIVA51KF1.pdf
(and source material) https://en.wikipedia.org/wiki/PID_controller#PID_tuning_software
Google "PID Tuning" for much more.

ECE 3510 Ladder Logic and Programmable Logic Controllers

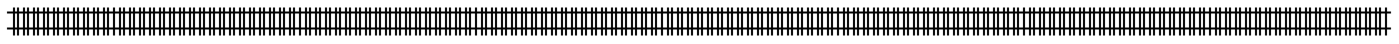
A.Stolp 3/26/20 a

Two reasons electricity is useful: Power conversion and transmission. Power available in one location as heat or mechanical motion can be used many different ways at some other location. First major use, lighting.

Signals. A voltage or current can represent information. Information in electrical form can be transmitted to a distant location and/or recorded for later use. First major use, telegraph.

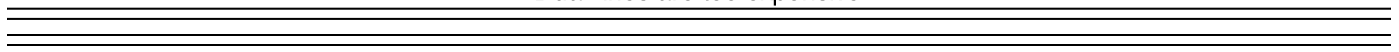
Rail Roads

One of the first big users of the telegraph

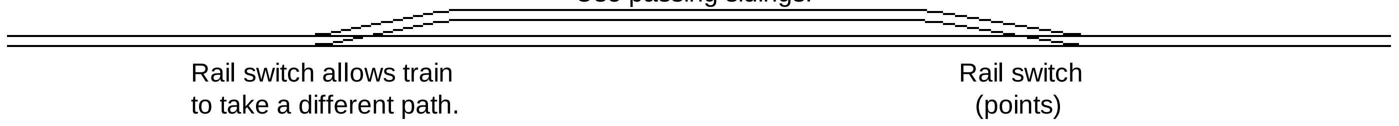


All railroad lines were initially single track,
but trains need to go both ways.
How do you deal with westbound trains and eastbound trains on the same tracks?
Crashes are **NOT** the answer!

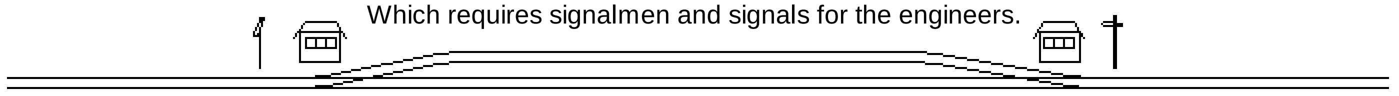
Dual lines are too expensive.



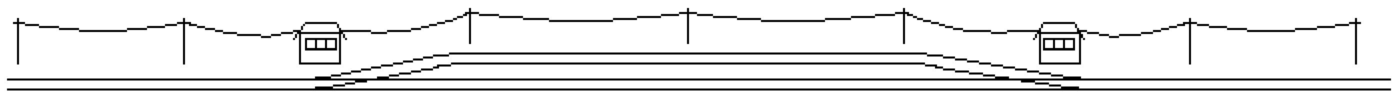
Use passing sidings.



Which requires signalmen and signals for the engineers.



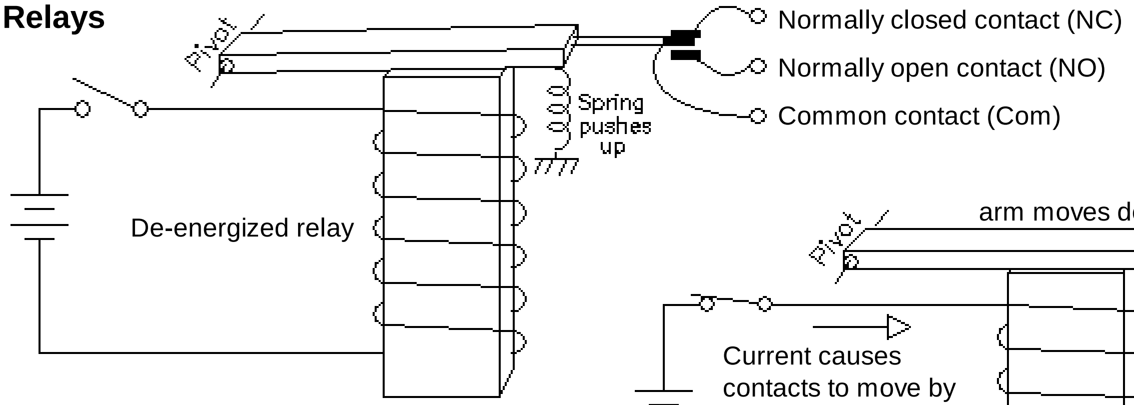
Of course these signalmen need to communicate whit each other, so add the telegraph line.

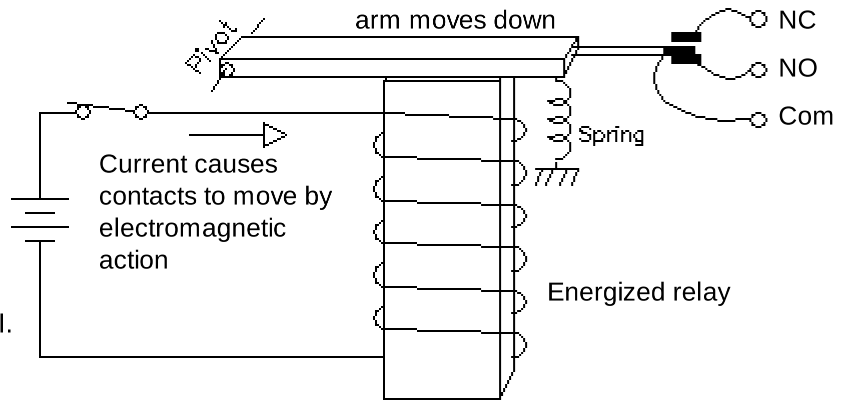


Clearly this is a system which cries out for a more automated approach, but remember, this is before computers, before logic gates, before transistors, even before vacuum tubes. I know, it's hard to imagine, but such times *did* exist.

They did have *one* device that could be used for logic, a magnetically operated switch, commonly called a relay.

Relays





A single relay can have many contacts.

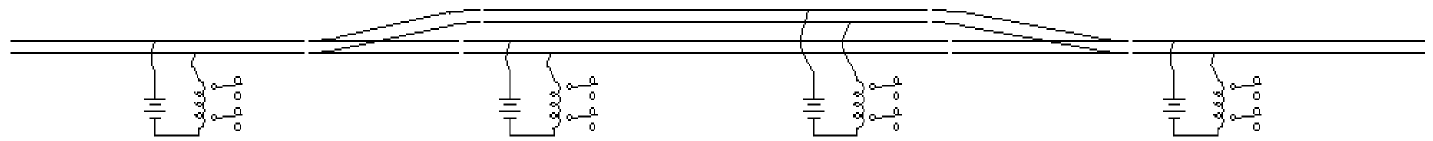
The current through the contacts can be much greater that current needed to energize the relay coil.

Relay coils may be operated from DC or AC.

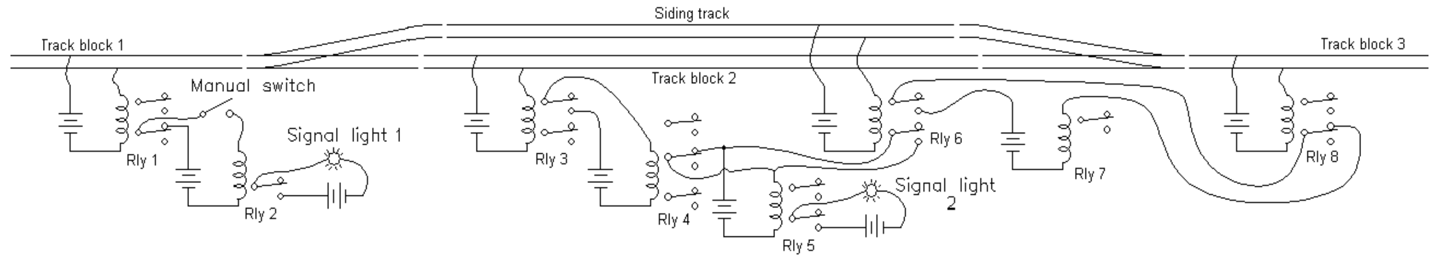
Fancier relays may have a time delay before switching the contacts.

Even fancier relays may rotate a drum with many contacts that connect in sequence.
And the fanciness doesn't end there.

Relay-relay logic



Notice the little gaps in the rails, they electrically separate the tracks into different regions, or blocks. When a train enters one of these regions, it makes contact between the two rails and the corresponding relay will be energized.

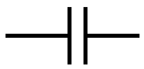


Relays can be hooked to one another and to other switches to create logic systems which which could operate signal lights or switch points or do many other things. But the logic is VERY hard to follow.

Ladder Logic

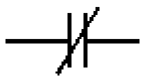
To more easily deal with the logic of all these relays and switches a new way to draw the schematic was developed. The power source for the relays was drawn as a vertical line on the left side. The ground or common was drawn as a vertical line on the right side. The individual logic circuits were drawn horizontally with switches and contacts on the left and relay coils and outputs on the right. Contacts and switches are the symbols that look to us like capacitors.

Normally open contact or switch

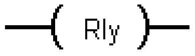


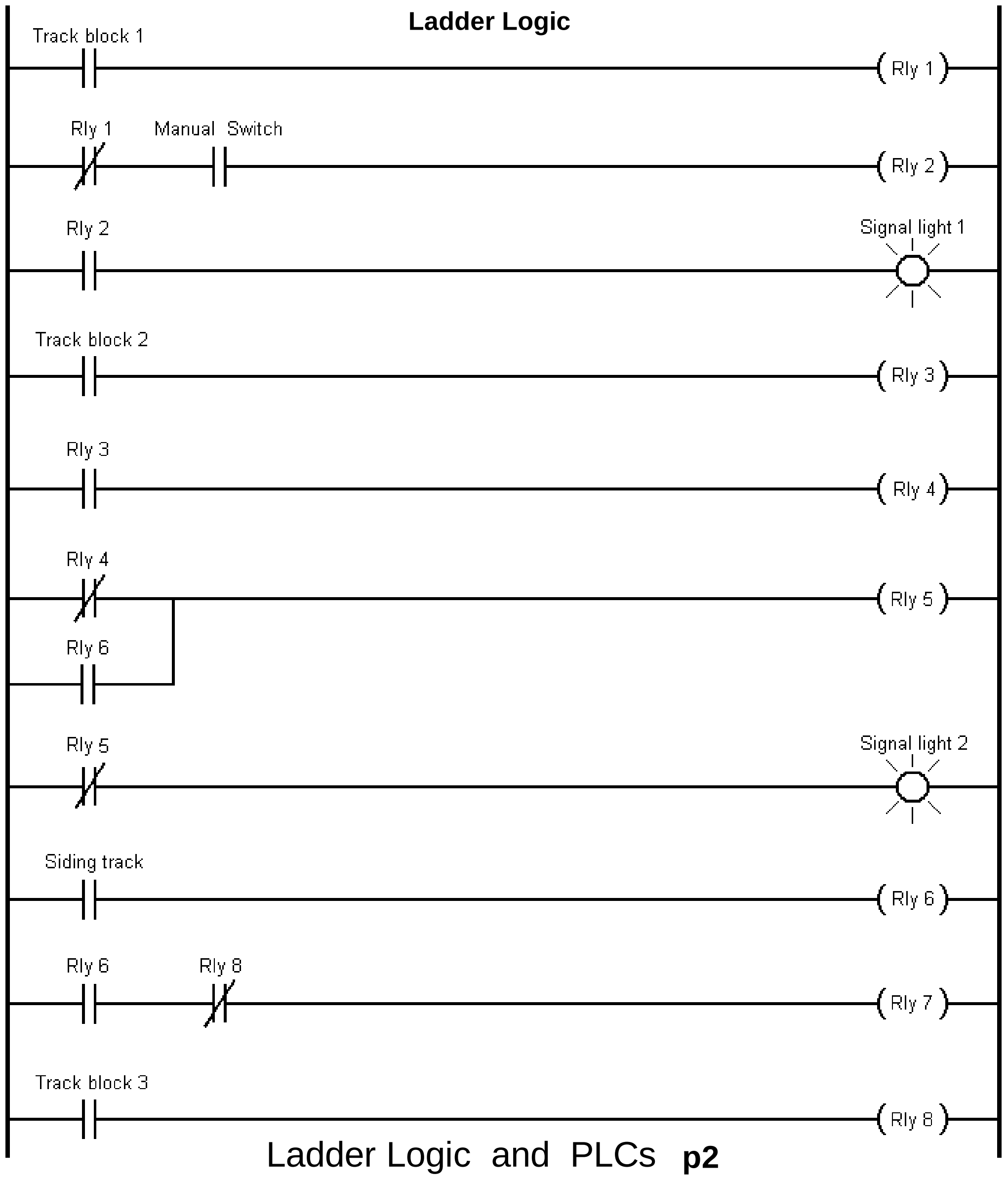
NOT capacitors

Normally closed contact or switch



Relay coil





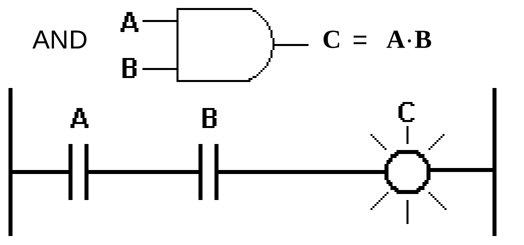
Sometimes this left vertical line is not shown.

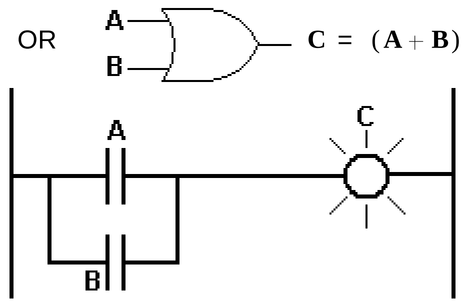
It should be obvious how ladder logic got its name.

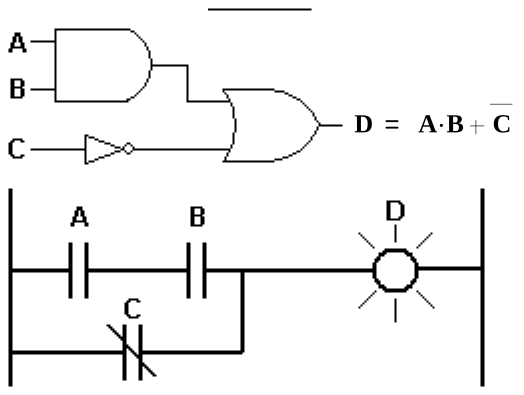
Don't waste time looking for logic in this circuit, it's just an illustration. Go to the next page to see logic

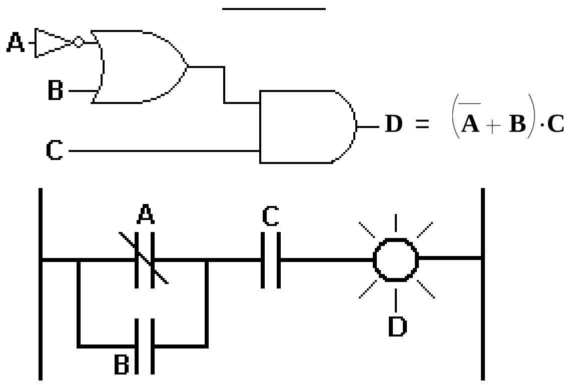
Logic

You are probably far more familiar with digital logic gates than this relay logic, so let's look at some analogies.





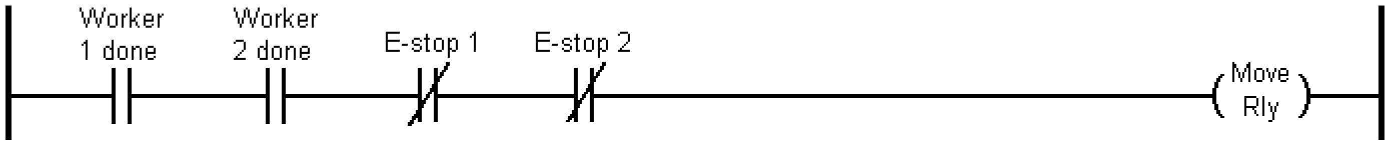




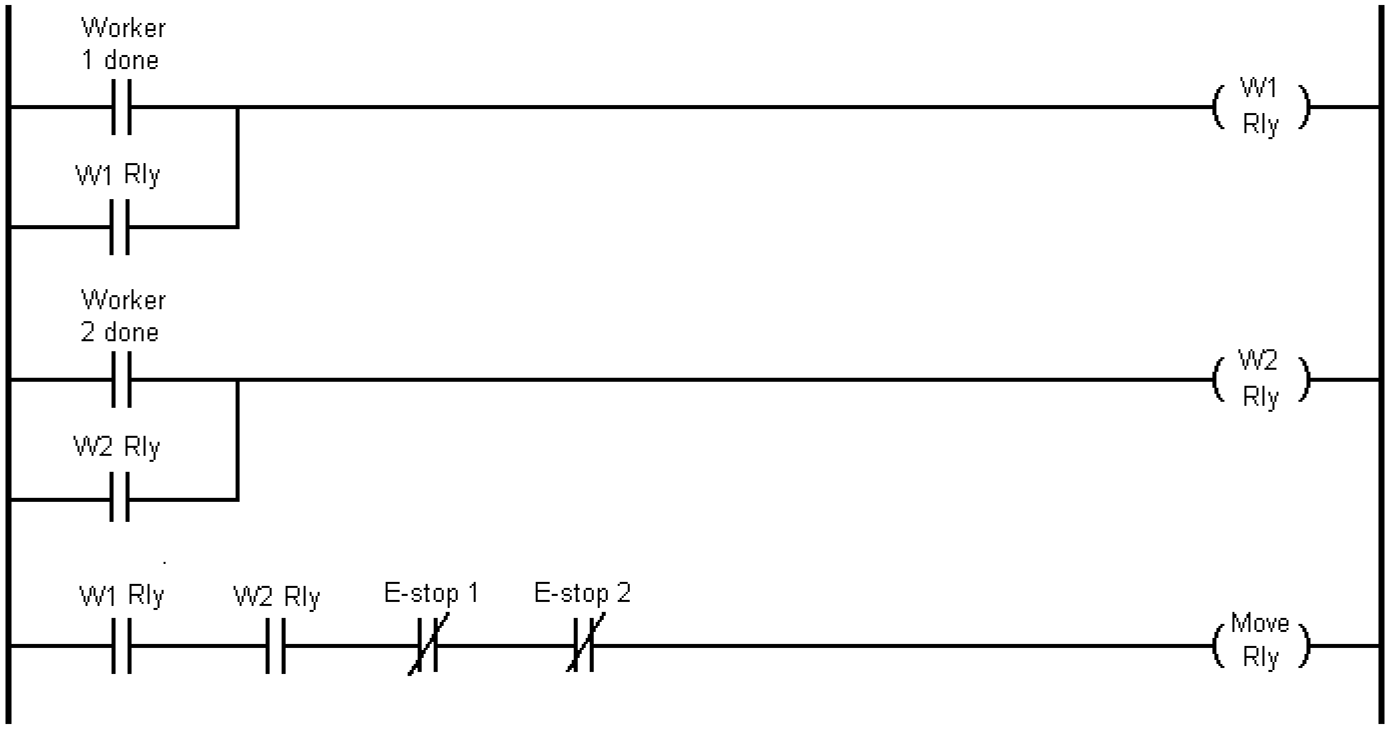
Industrial Control

This is where ladder logic is still used extensively, because it is so well suited to simple on/off logic.

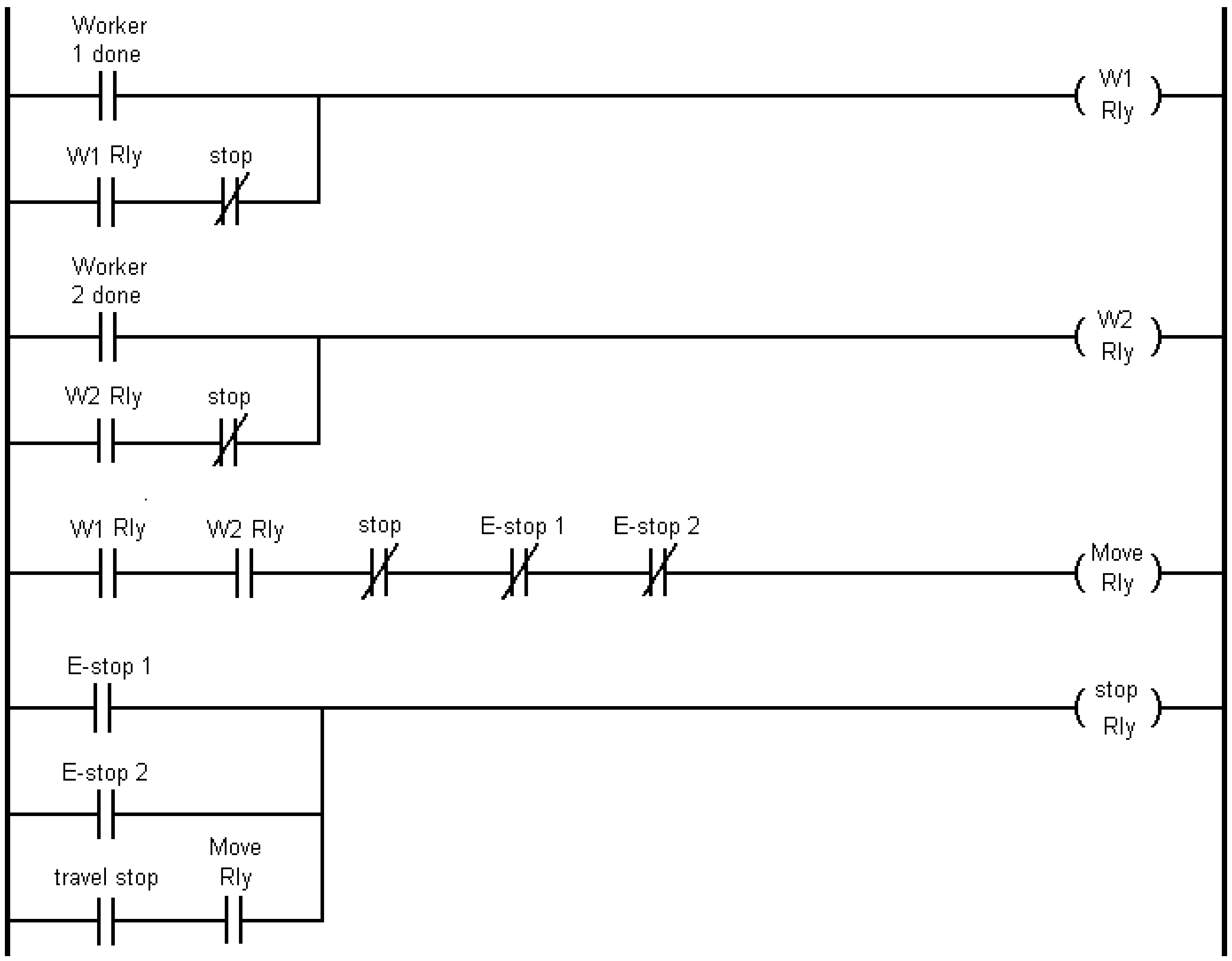
Ex Say 2 workers are performing tasks on an item on an assembly line. Each worker must push a button when he is done to allow the item to move to the next station. Each worker also has an emergency stop (E-stop) button.



Maybe you'd like to change this so the workers don't have to hold down the buttons while the other worker finishes and the item moves to the next station.



Unfortunately, now it won't stop moving unless one of the workers holds down their E-stop button. Let's put in a stop relay.



I've also added a "travel stop" input to stop the movement when the item reaches the next station and a reset method (which may not allow the next movement the unless you roll past the "travel stop" sensor). Also note that emergency stop buttons work in two ways, one of which does not rely on the stop relay.

What I haven't done is build this circuit to see if it will really work as expected. Wouldn't it be nice if I didn't have to build the circuit with real relays. What if most of these relays could be simulated with a computer and all I would need to do is wire up a few switches and sensors and one big relay for the motor. A Programmable Logic Controller (PLC) does exactly that.

PLCs

Actual relays may still be used to control large currents to motors and such, but they're not used for the logic anymore. Programmable Logic Controllers (PLCs) have replaced the logic-only relays. They are faster, more reliable, more flexible, and way easier to program than rewiring relay circuits. PLCs are programmed with special software that runs on a normal desktop or laptop. The software runs a form of ladder logic and can often run some C code or other high-level language as well. They also simulate the entire program operation, including inputs and outputs so tah you can make sure you program works before connecting to any harware.

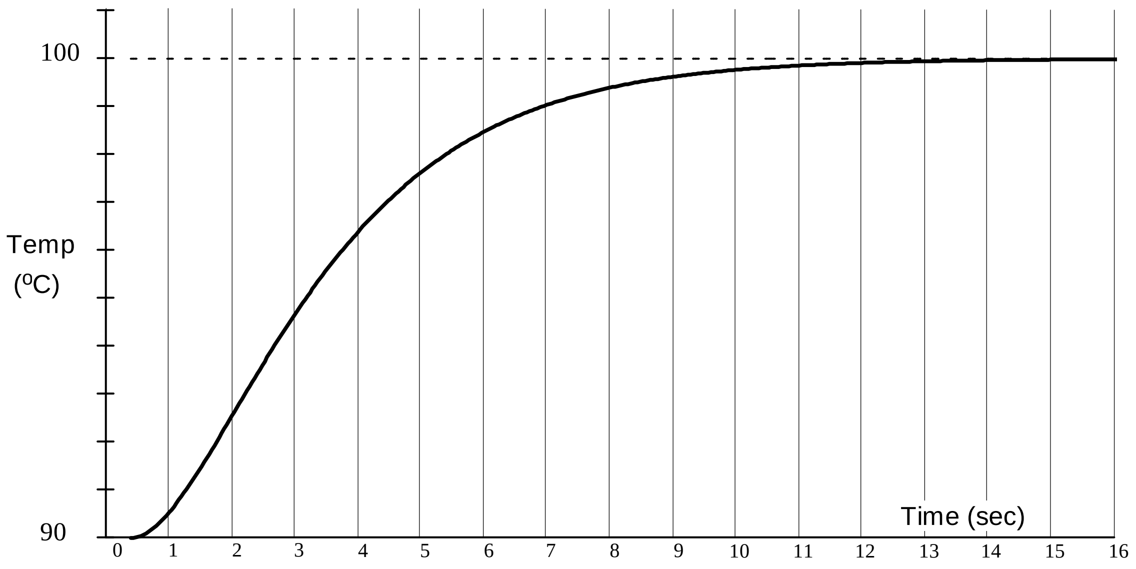
The main purpose of these notes and the supporting lecture and homework is to gave you short introduction to a very common way of implementing industrial control systems. I want you to know the terms and have confidence that you could quickly become proficient with this type of programming. That way, if you are ever asked about ladder logic and PLCs in a job interview, you won't come off as an ignorant idiot.

In the homework you will use PLC programming software learn more about ladder logic and run some simulations. You will not become an expert, but at least will know you <u>can</u> become one.

Folder ___________ Name ___________________________________

ECE 3510 hw FC1 PID Tuning Due: Thur, 4/1 A.Stolp

1. You are trying to control a process temperature in a nanofabrication lab. The current open-loop response to a temperature adjustment is shown below. Use the Ziegler-Nichols tuning method to answer the questions below.



a) Find L and R from the plot above, show your work above and hand in this plot with your homework.

b) Use these values to find the proportional gain of a proportional-only (P) controller in a closed loop.

c) Use these values to find the proportional gain and integral time of a Z-N PI controller.

d) Use these values to find the proportional and integral gains of the PI controller we commonly use in class.

e) Use these values to find the proportional gain integral time and differential time of a Z-N PID controller.

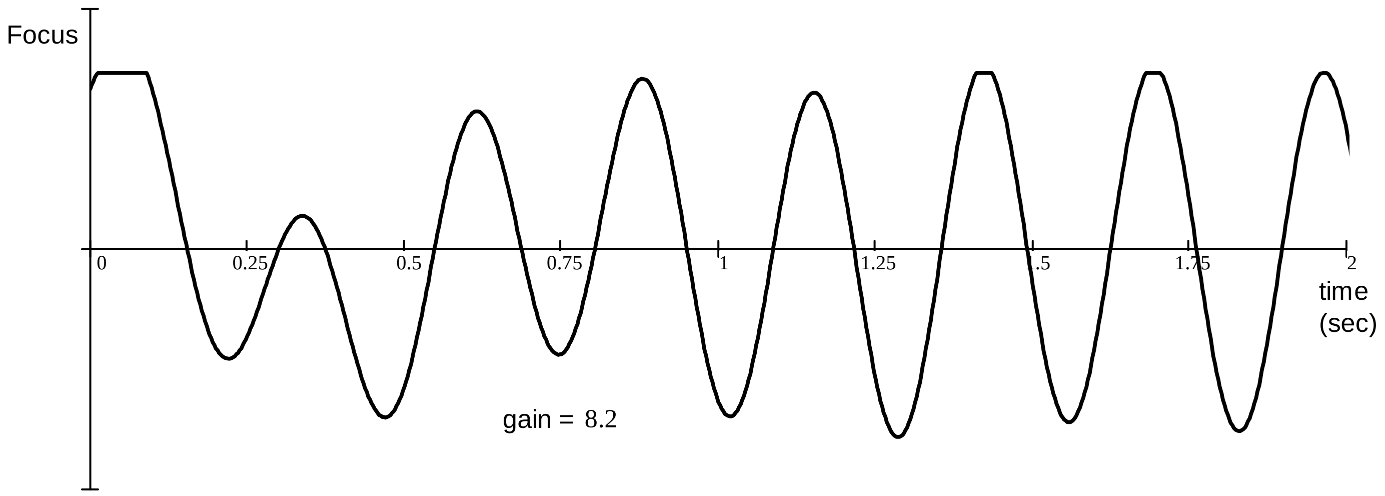
f) Draw a block diagram of this controller, showing the gains.

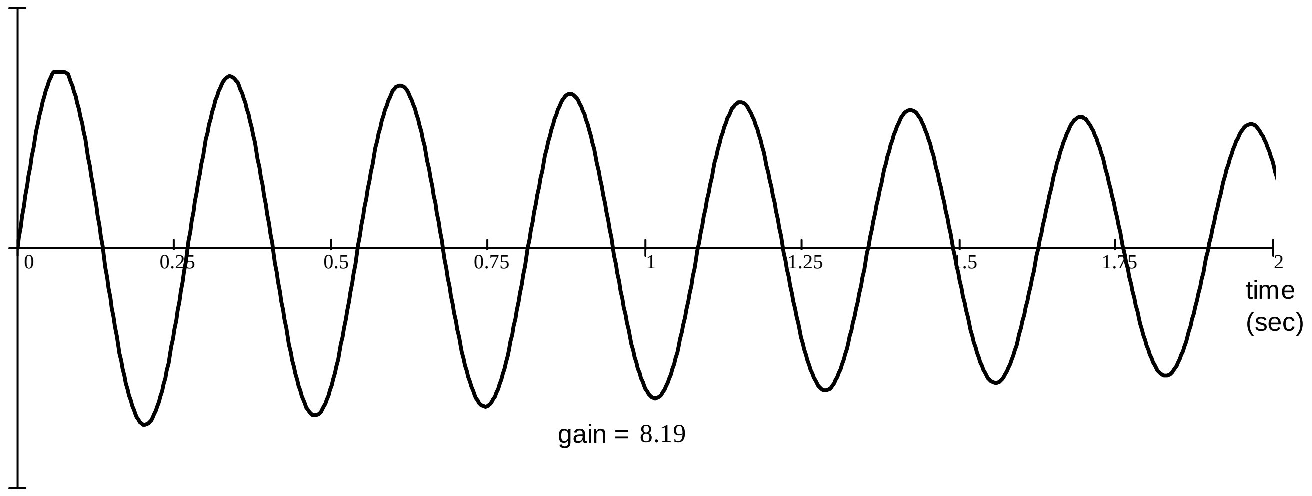
g) Use these values to find the 3 gains of the PID controller we commonly use in class.

2. Next, you are trying to control a photographic focus mechanism in a nanofabrication lab. The mechanism uses a DC motor to control the position of the focusing lens. Use the Ziegler-Nichols tuning method to answer the questions below on another sheet of paper.

   a) Can you use the method of problem 1 to find the controller parameters? If not, why not?

   You've created a simple proportional-gain feedback loop to control the focus and are playing with the gain to get the responses below at the gain settings shown.





Find the values you need from these plots (show work on plots) to answer the following on another sheet:

b) Find the proportional gain of a proportional-only (P) controller.

c) Find the proportional gain and integral time of a Z-N PI controller.

d) Find the proportional and integral gains of the PI controller we commonly use in class.

e) Find the proportional gain integral time and differential time of a Z-N PID controller.

f) Find the 3 gains of the PID controller we commonly use in class.

g) The overshoot turns out to be too high, but you like the speed. What should you try to do first?, second?, third?

Answers

1. a)	Approximately 0.8 & 2.08			2. a)	No answer the "why" yourself		
b)	0.6			b)	4.1		
c)	0.54	2.67		c)	3.69	0.208	
d)	0.54	0.202		d)	3.69	17.7	
e)	0.72	1.6	0.4	e)	4.92	0.125	0.031
g)	0.72	0.45	0.288	f)	4.92	39.4	0.154

g) 1st, Try turning up the differential gain

2nd, Try turning down the integer gain, but keep an eye on the steady-state error

3rd, Try turning down the proportional gain, but this could slow down the system

ECE 3510 hw FC2 Programmable Logic Controller Due: Mon, 4/5 A.Stolp
Ladder Logic

1. Use any one of the computers in any one of the ECE labs, or download and install the following program on your own computer.

   To download and install:

   You can get the program from our class website or the source website.
   Our website, download and run: www.ece.utah.edu/~ece3510/SetupTL6Edu.exe (or see below).
   Setup Password: LadderBasic2009

   Source website: www.tri-plc.com/trilogi.htm. You can download the ladder logic simulation software and install it on your computer, or you may be able to get by with just the the web version. Either way, you will need to have Java Runtime Environment on your computer. You may need a setup password different from that given above

2. Start --> i-TRiLOGI 6 (Educational) --> Internet TRiLOGI Helps
   Read through "Ladder Logic Fundamentals: Contact & Coils" under section 3
   Note: the same help page is available from the Help --> Content menu of the program itself.
   Open the "Ladder Logic Programming Tutorial" under section 1

3. Start --> i-TRiLOGI 6 (Educational) --> TRiLOGI 6.2 (Education Version)
   Work through the "Ladder Logic Programming Tutorial". Save your work.

4. Start --> i-TRiLOGI 6 (Educational) --> Internet TRiLOGI Helps
   Be ready to to read through some of the other sections under "Ladder Logic Language Reference" to complete step 5, below.

5. Change the automatic (Manual is turned off) mode of your sequencer so that it takes 4 seconds per step.

   I did this by adding two circuits (ladder rungs) between the original Circuit#1 and Circuit#2. These each operated a 2 second timer (set value at 20). When the first timer hit 2 seconds (counted down from 20) it started the second timer. When the second timer reached 2 seconds, it cut off current to the first timer which then cut off current to the second timer which then restored current to the first timer and started the whole process again. Finally, I replaced the Clk0.5s switch (in what is now Circuit#4) with one that was operated by the first timer.

   You may choose a different method. Save your work.

6. Make at least two significant changes to your ladder so that it will do something new that you find interesting.
   You may add or change circuits (ladder rungs).

   For example, you might try to modify the delay at some points in the sequence, say double the time it spends at 3rd step. Or you might add two circuits that implement something completly different, using different inputs and outputs. This entirely your choice, but try to make it interesting and learn about some new function.

7. Save your work, using your last name as the file name. Attach it to an email and send it to:
   "Bhawna Lakshimpathy" <B.Lakshmipathy@utah.edu>
   Subject: ECE 3510 homework FC2

   You may uninstall the TRiLOGI program if you wish. We will not use it again in this class.

Other sources of information:

http://openbookproject.net//electricCircuits/Digital/DIGI_6.html

ECE 3510 homework # FC2