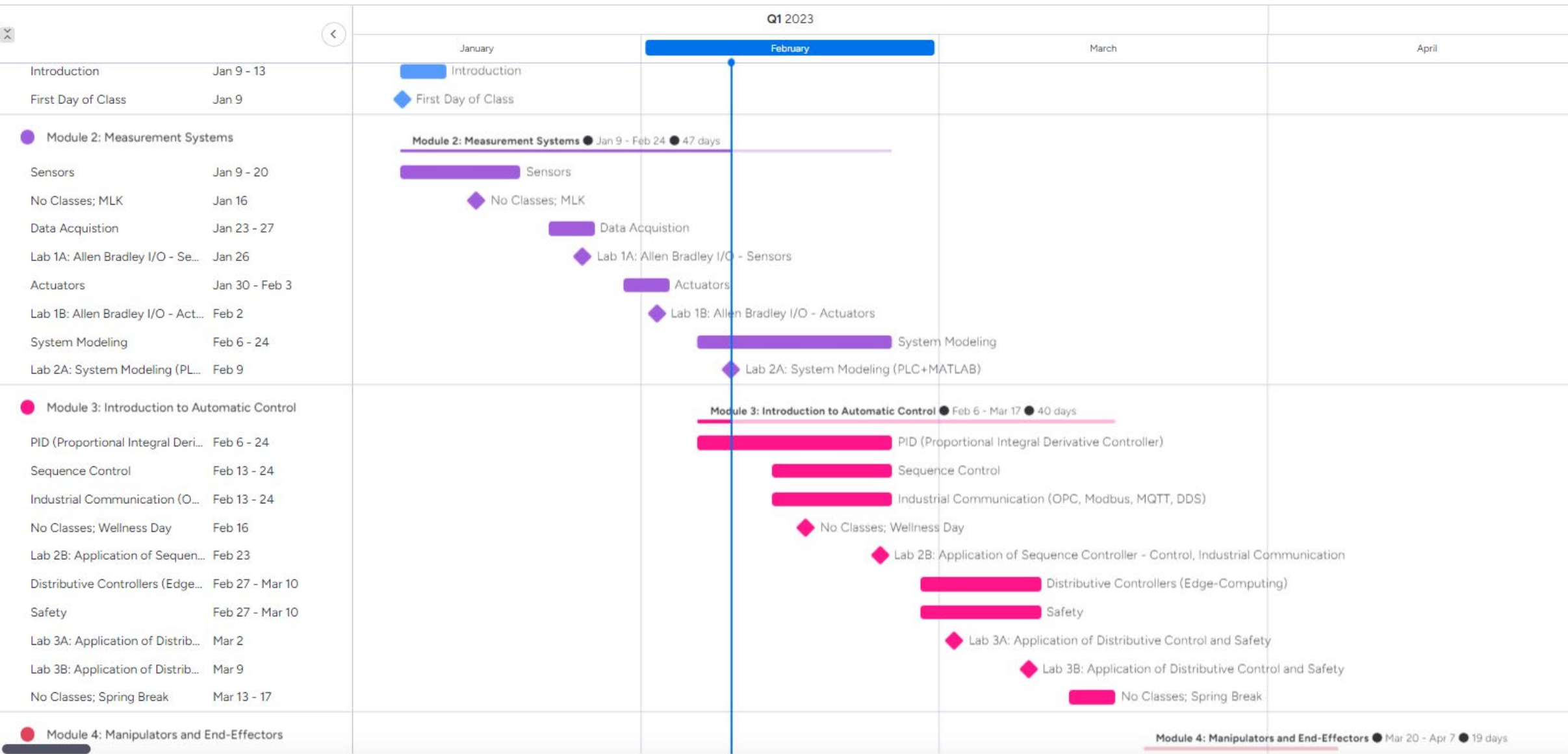




ISE 589-006: INTRODUCTION TO MODERN INDUSTRIAL AUTOMATION

LECTURE 005
Fred Livingston, PhD



LECTURE 005

- Sensors and Actuators (Review)
- Homework 1 Solutions
- System Modeling
- Open-Loop Control
- Introduction to Industrial Communication Protocols (Modbus)
- Laboratory Assignment

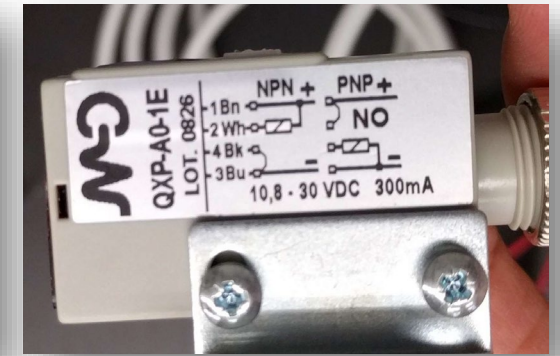
SENSORS



Inductive sensor



Inductive sensor



Rectangle head photo sensor

- If you are using an inductive sensor, connect the black wire to the PLC's Input0 (I-00), connect the brown wire the positive terminal block and connect the blue wire the negative terminal block.
- If connecting a photo sensor with a round head, connect the blue and pink wires to the negative terminal block. Connect the brown wire to the positive terminal block. Connect the black wire to the PLC's Input0 (I-00).
- If connecting a photo sensor with a rectangle head, connect the blue and black wires to the negative terminal block. Connect the brown wire to the positive terminal block. Connect the white wire to the PLC's Input0 (I-00)

ACTUATORS

- Actuator is hardware device that usually converts a controller command signal into a change in position or velocity of a mechanical device
- Classification based on the type of power used:
 - Electrical
 - Hydraulic
 - Pneumatic

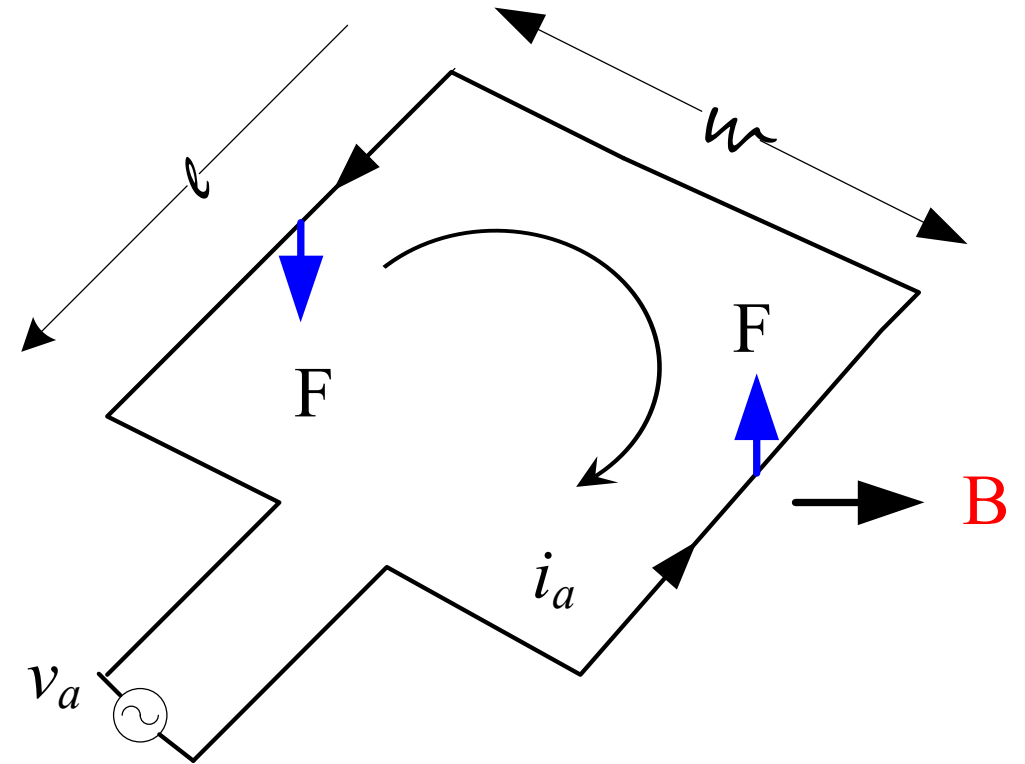
DC MOTOR — ELECTRIC MECHANICAL CONNECTIONS

Electric torque:

- $T_e = Fw = NBlwi_a = K_e i_a$

Back e.m.f:

- $v_b = k_2 B\omega = K_b \omega$



$$F = Bli_a \text{ for one coil}$$

$$F = NBli_a \text{ for } N \text{ coils}$$

DC MOTOR DYNAMICS

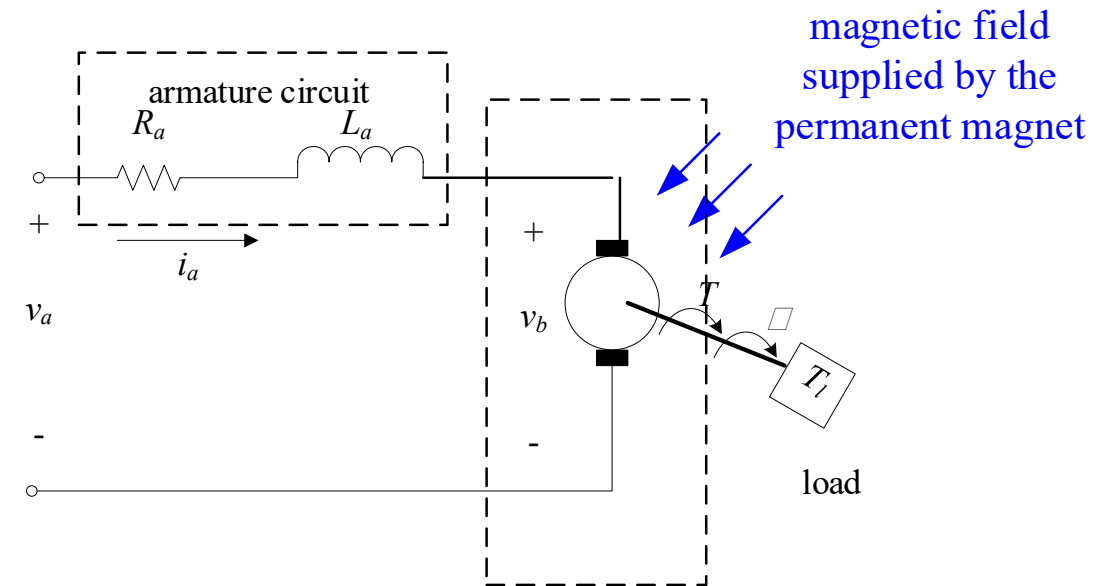
Armature circuit:

- $v_a - v_b = L_a \frac{di_a}{dt} + R_a i_a$
- $\frac{i_a}{v_a - v_b} = \frac{1}{L_a s + R_a}$

Rotating Motion:

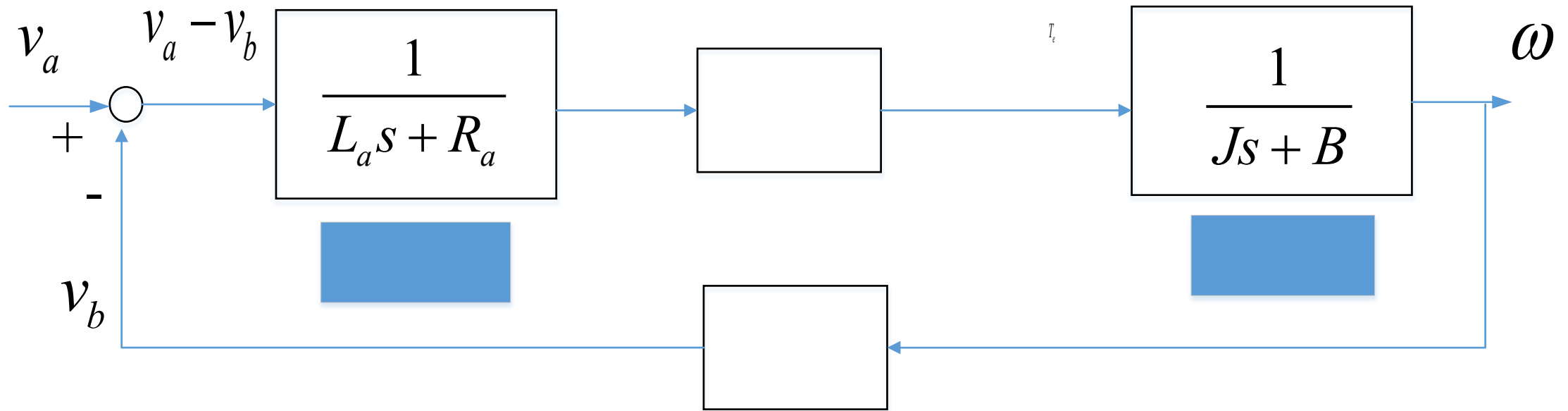
- $J \frac{d\omega}{dt} + B\omega = T_e$
- $J \frac{d^2\theta}{dt^2} + B \frac{d\theta}{dt} = T_e$

$$\frac{\omega}{T_e} = \frac{1}{Js + B}$$
$$\frac{\theta}{T_e} = \frac{1}{s(Js + B)}$$

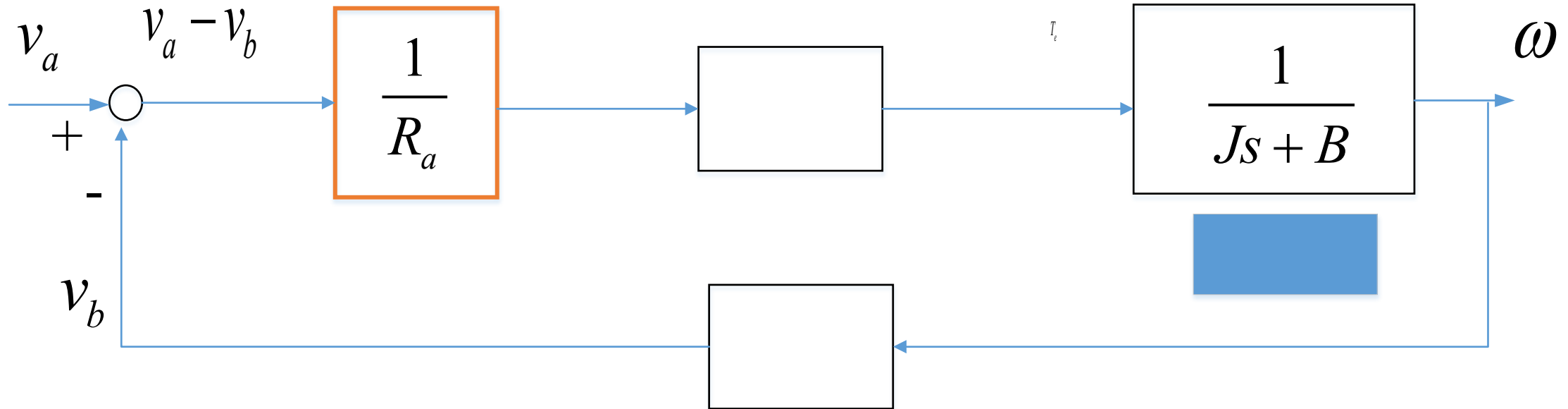


DC MOTOR TRANSFER FUNCTION

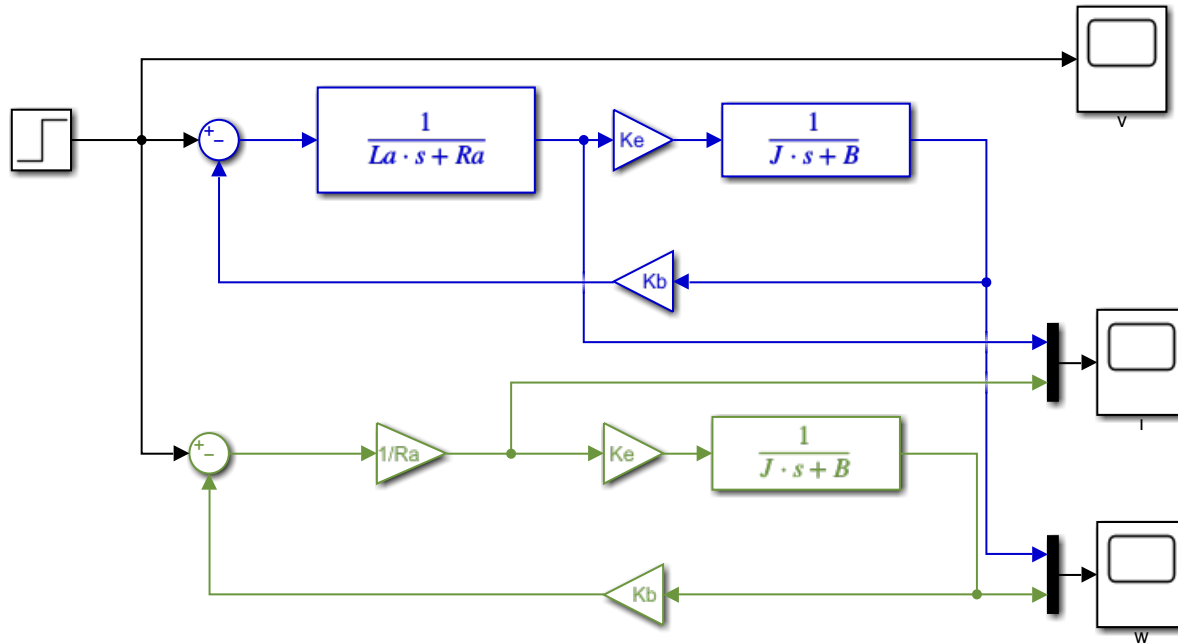
- DC motor dynamic block diagram:



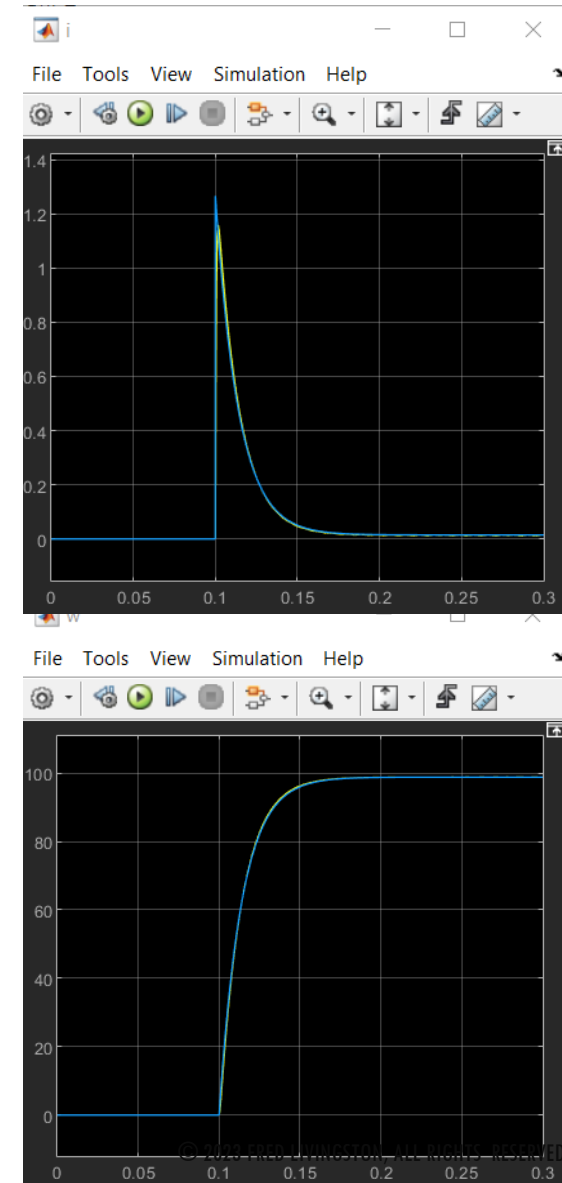
DC MOTOR SIMPLIFIED TRANSFER FUNCTION



DC MOTOR MODEL PERFORMANCE COMPARISON



$L_a = 0.37e-3$ % inductance
 $R_a = 0.79$ % terminal resistance
 $J = 1.62e-6$ % Rotor inertia
 $B = 1.34e-6$ % Viscous damping factor
 $K_e = 0.009$ % Torque constant
 $K_b = 0.01$ % Motor constant

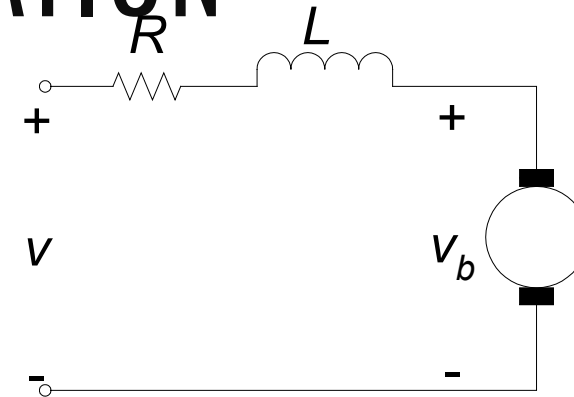


DC MOTOR STEADY-STATE OPERATION

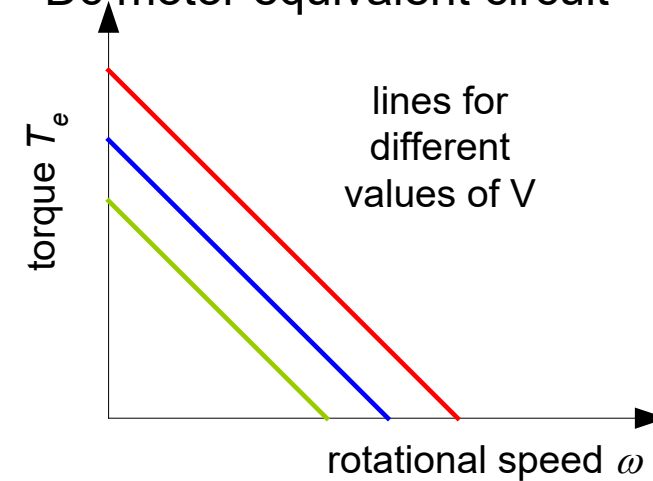
At steady-state: $di/dt = 0$ and $d\omega/dt = 0$

$$\Rightarrow i_a = \frac{v_a - v_b}{R_a} = \frac{v_a - K_b \omega}{R_a}$$

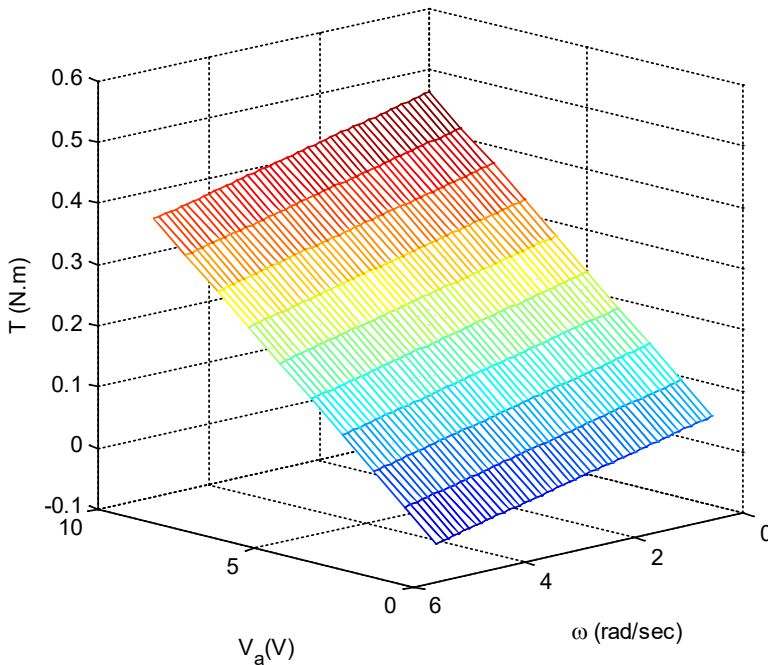
$$T_e = K_e i_a = K_e \frac{(v_a - K_b \omega)}{R_a}$$



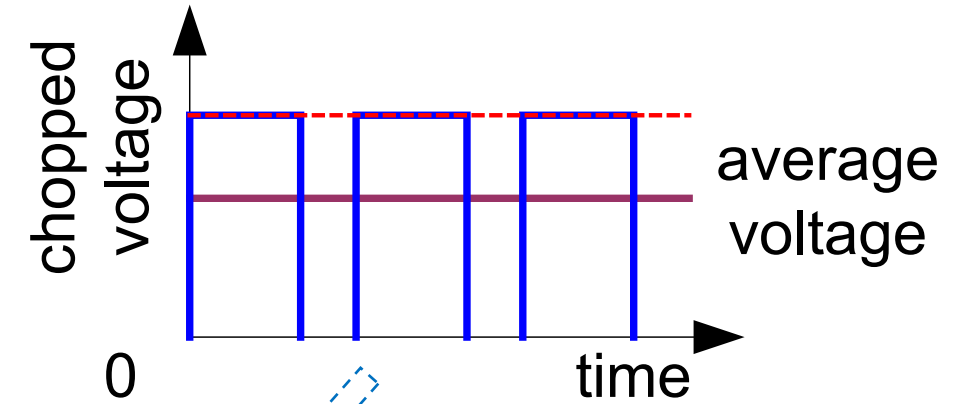
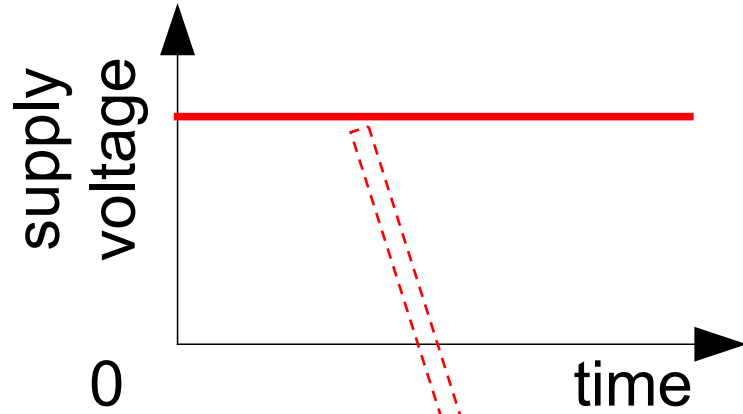
Dc motor equivalent circuit



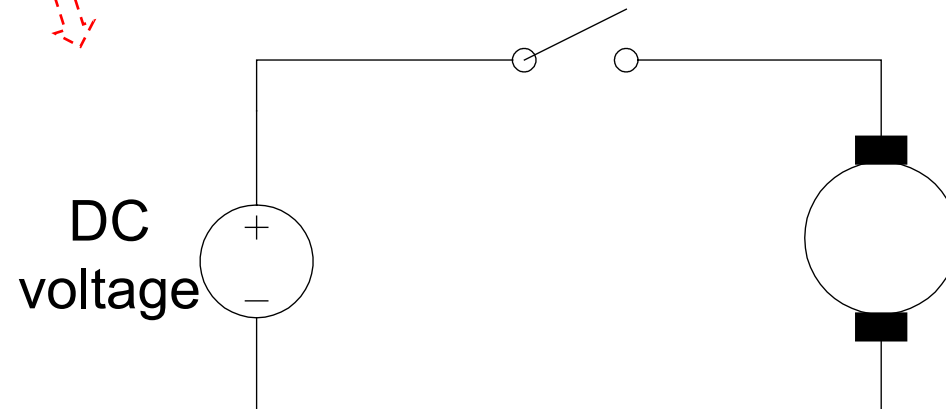
torque-speed characteristic with different input voltage



CONTROL OF D.C. MOTORS

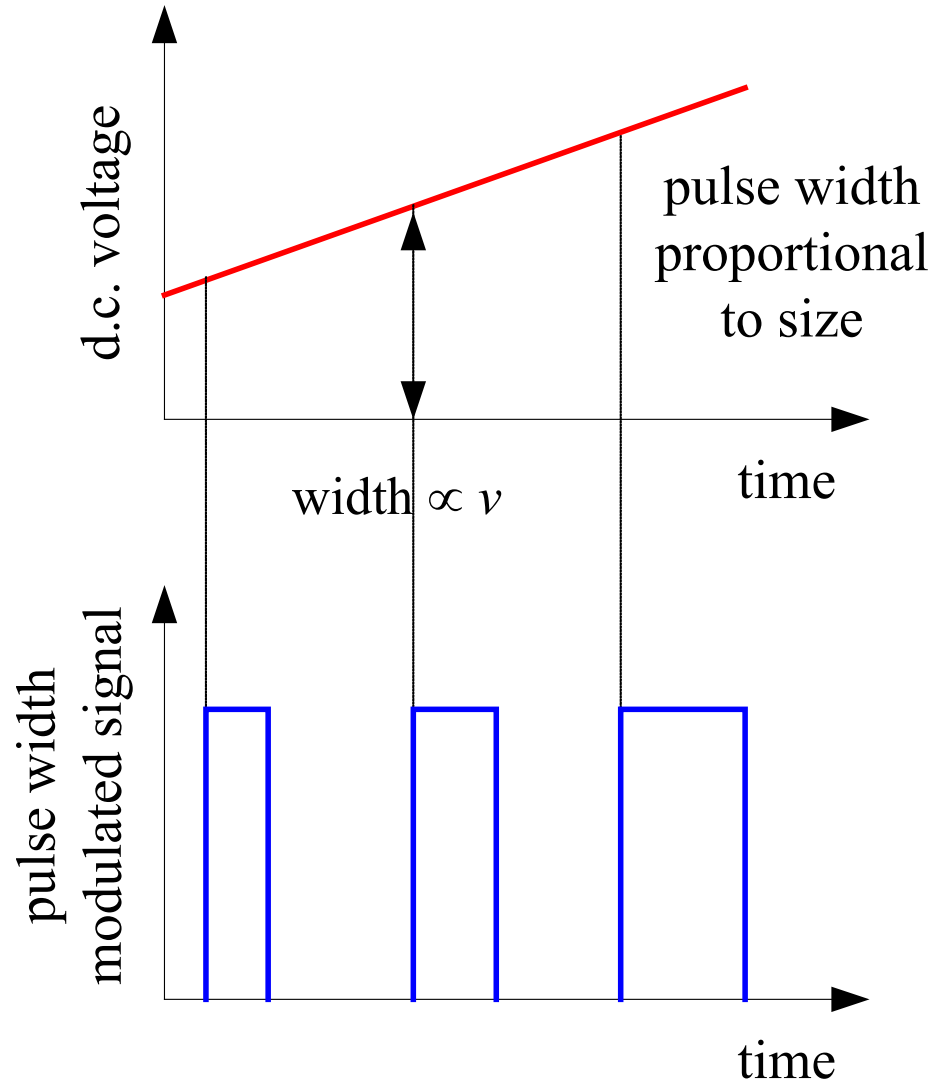


electronically controlled
high frequency switch
to chop the d.c. voltage

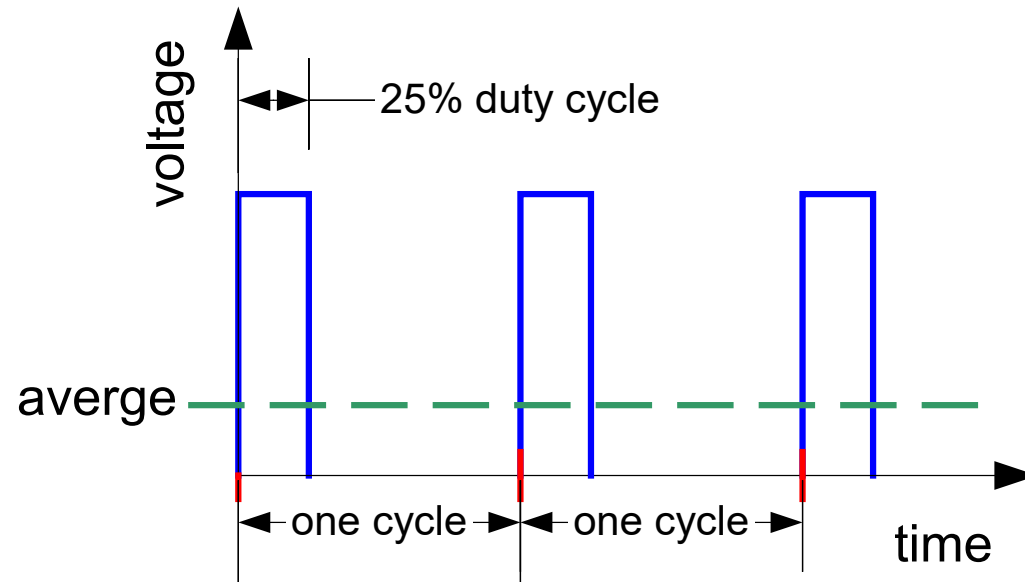
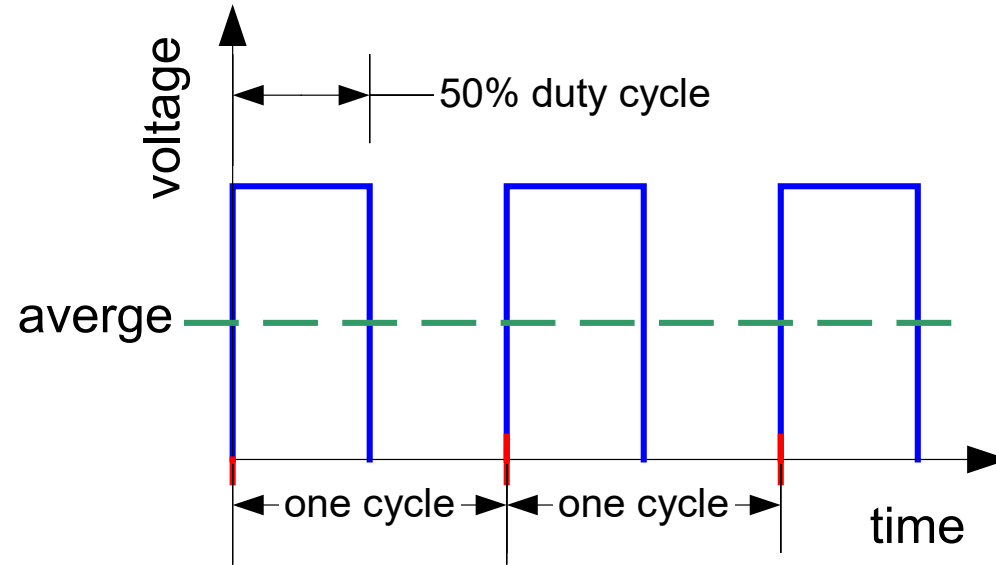


$$\bar{v}(t) \cong \frac{1}{t - t_0} \int_{t_0}^t v(t) dt$$

PULSE WIDTH MODULATION (PWM)



PWM — DUTY CYCLE



EXAMPLE: DC-MOTOR CONROL

A DC Motor operating at 24V DC is connected to a Micro850 PLC. The Micro850 PLC Pulse Timer Output (PTO) is set to 10 Hz. It is desired to rotated the armature at 20% of its maximum velocity.

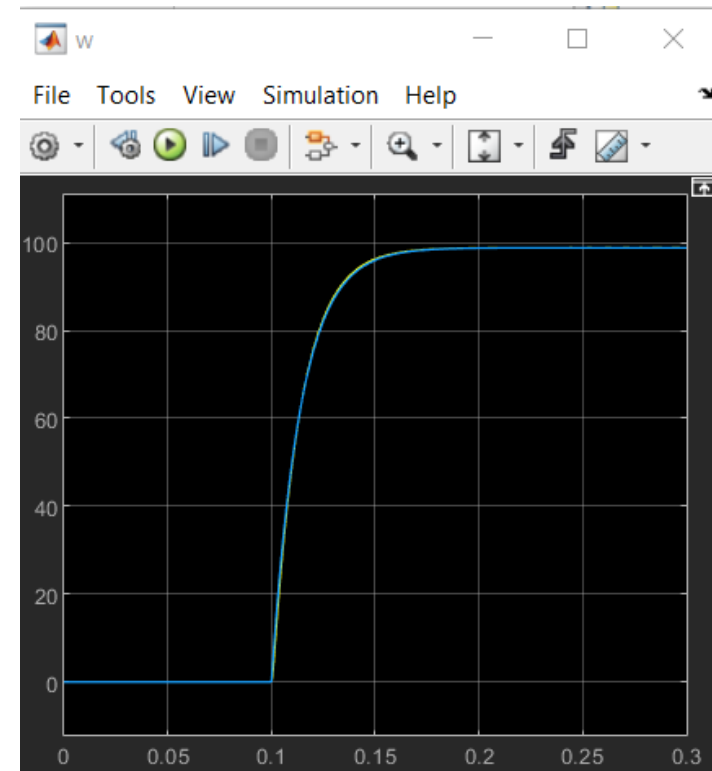
What is the on-pulse width of the control signal?

What is the average voltage of the signal?

EXAMPLE: DC-MOTOR CONROL

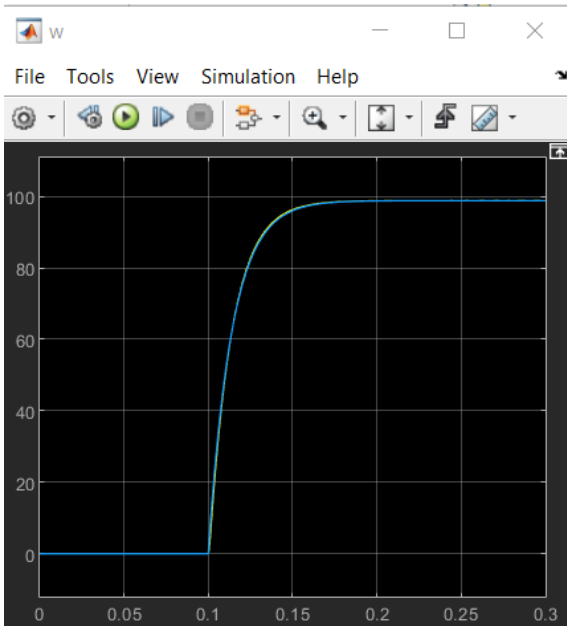
Given the characteristics of the DC motor what is the angular velocity of the motor at 20%?

```
La = 0.37e-3 % inductance
Ra = 0.79    % terminal resistance
J = 1.62e-6  % Rotor inertia
B = 1.34e-6  % Viscous damping factor
Ke = 0.009   % Torque constant
Kb = 0.01    % Motor constant
```



EXAMPLE: SOLUTION

A DC Motor operating at 24V DC is connected to a Micro850 PLC. The Micro850 PLC Pulse Timer Output (PTO) is set to 10 Hz. It is desired to rotate the armature at 20% of its maximum velocity.



$$V_{average} = V_{cc} \cdot \text{Duty Cycle}$$

$$V_{average} = 24 \cdot 0.2 = 4.8$$

$$V_{rpm}(\max t \rightarrow \infty) = 100 \text{ rpm}$$

$$V_{rpm}(20\%) = 0.2 \cdot 100 \text{ rpm} = 20 \text{ rpm}$$

A red dual-arm robot with a screen face is positioned over a conveyor belt in a factory setting. The robot's arms are extended, and it appears to be working on a component on the belt. The background shows industrial equipment, including large white pipes and a metal structure. The text "HW 1 SOLUTIONS" is overlaid on the image in a white, sans-serif font, with a thin blue horizontal line underneath it.

HW 1 SOLUTIONS

SYSTEM IDENTIFICATION

- *System identification* is to find the parameter values of the model (not necessarily the physical parameter values, e.g., damping coefficient) by knowing the input-output performance of the system, and the underlined mathematical model (e.g., first-order system).

- In another words, with given a system dynamics:

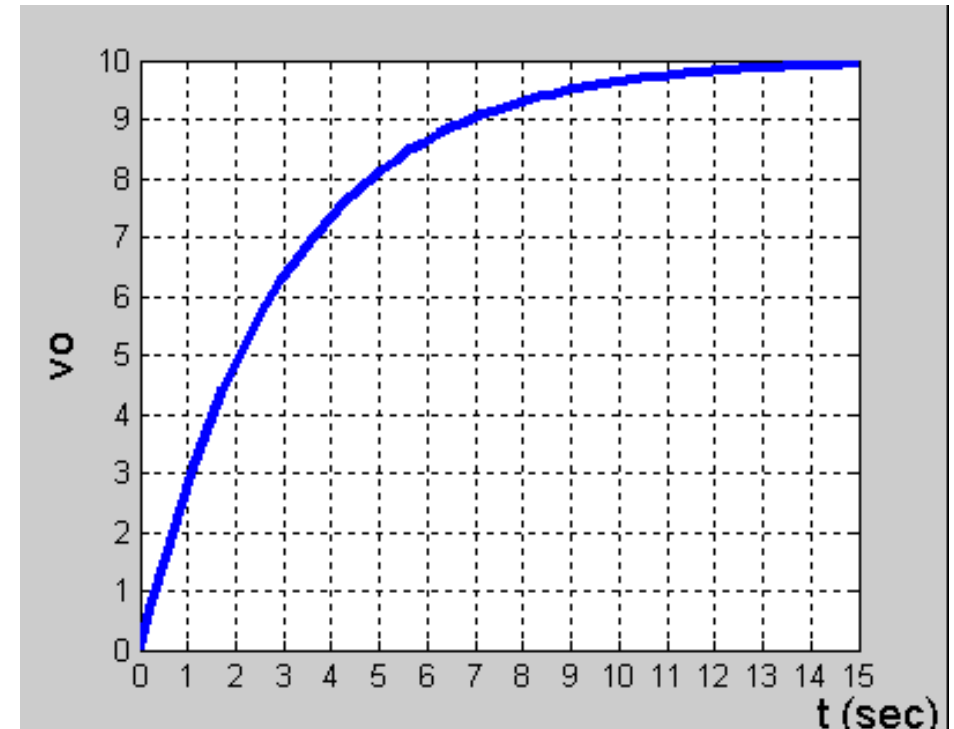
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p})$$

where \mathbf{x} is the state variables, and \mathbf{p} is the system parameters

- Question: What are the values of \mathbf{p} ?

EXAMPLE: SYSTEM IDENTIFICATION

- *Question:* We know that the dc motor speed output and voltage input is well represented by a first order system dynamics, yet we do not know the exact parameter values. In such a case, we can go to the lab to do a step response on the dc motor with inputting a step input voltage (e.g., 5 V) and measure the output motor speed as a function of time. The output as a function of time (capture by the oscilloscope) when subject to a step input of 5 V.
- What is the dynamics of the dc motor?



SOLUTION

- Since we know that the dc motor is a first-order system, we can assume that its dynamics can be described as:

$$\tau \frac{dy}{dt} + y = G_{ss} u$$

where τ and G_{ss} are the system dynamics parameters that we want to find.

- Since $\tau \cong 3$ sec

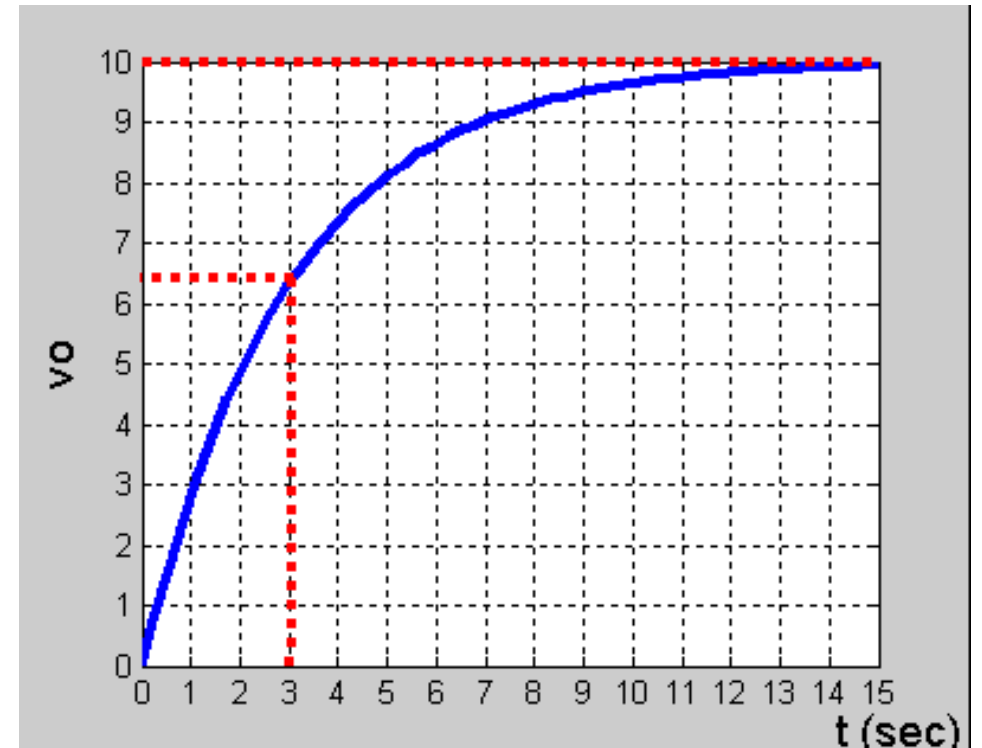
$$v_i = 5, \quad v_o(\infty) = G_{ss} v_i \cong 10$$

⇒ the steady-state gain:

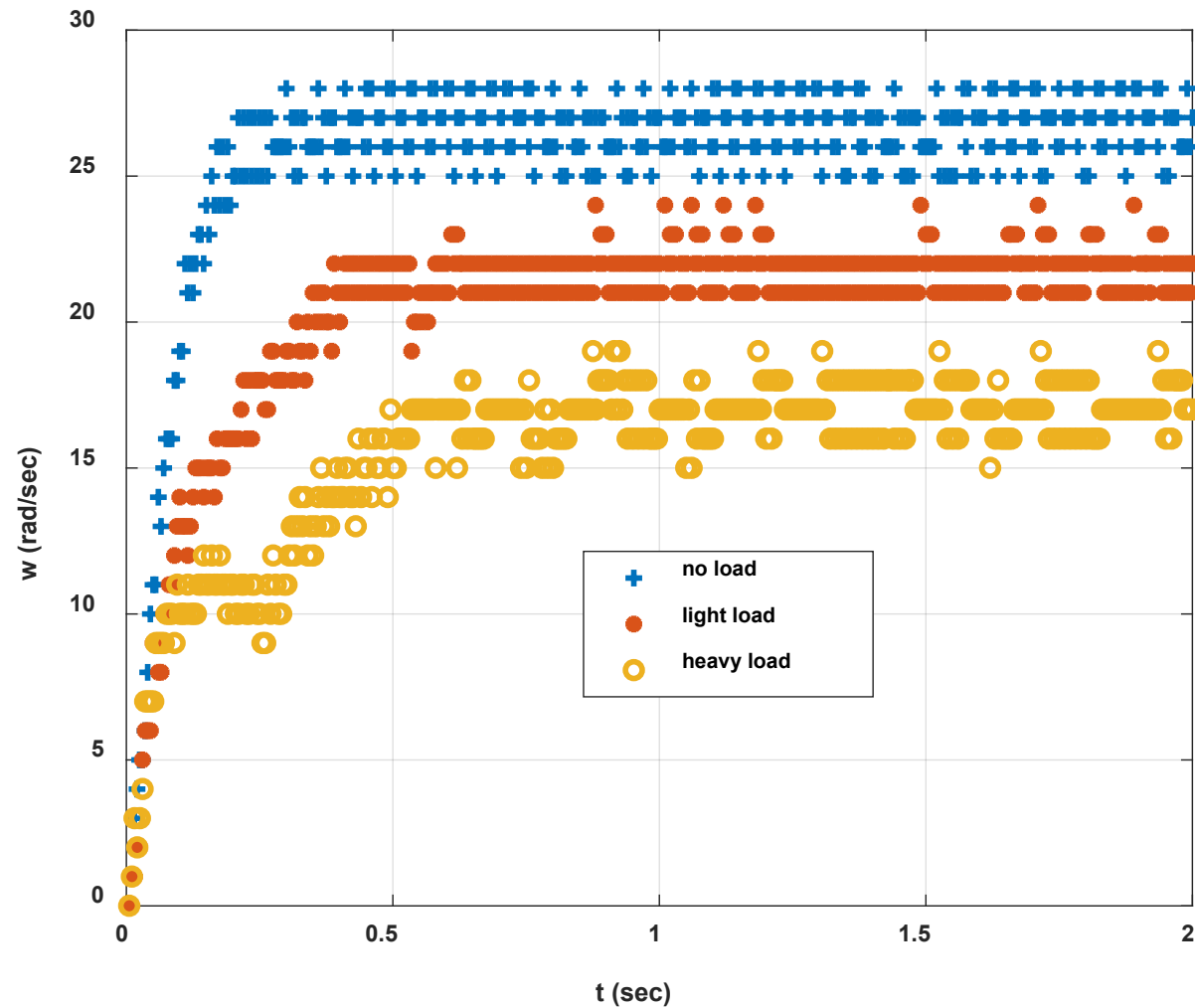
$$G_{ss} = \frac{V_{o,ss}}{V_{i,ss}} = \frac{10}{5} = 2$$

- Thus, the system dynamics can be described as:

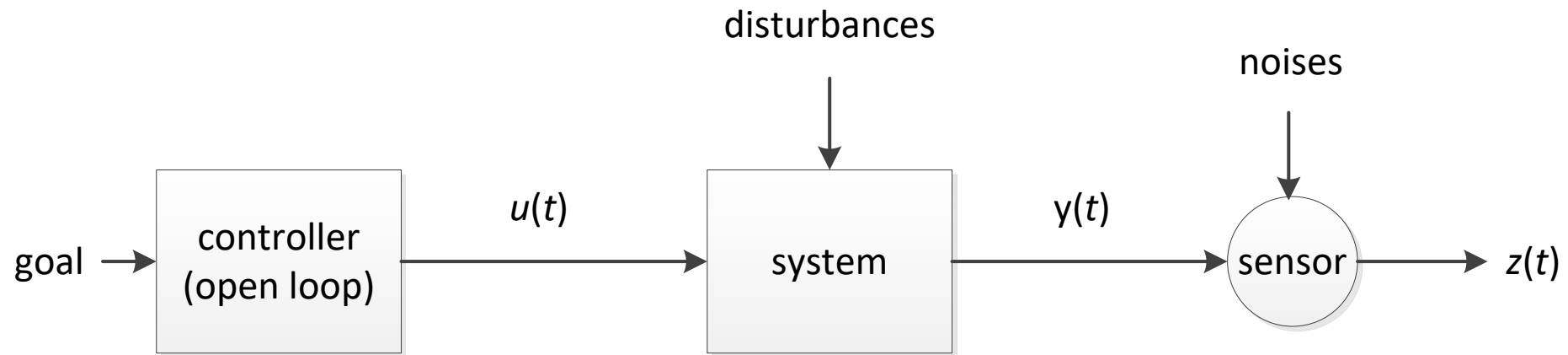
$$3 \frac{dv_0}{dt} + v_0 = 2v_i$$



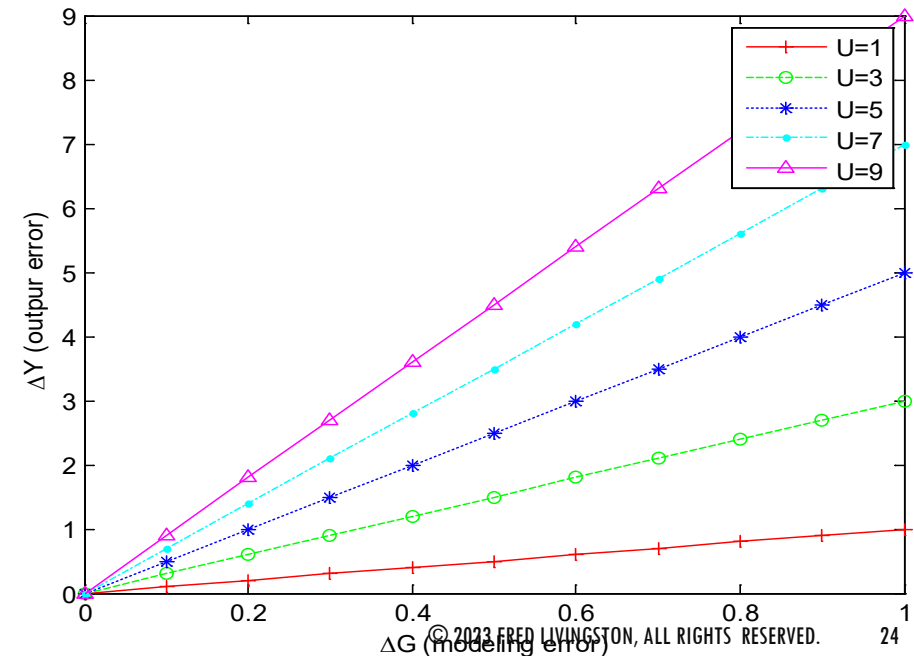
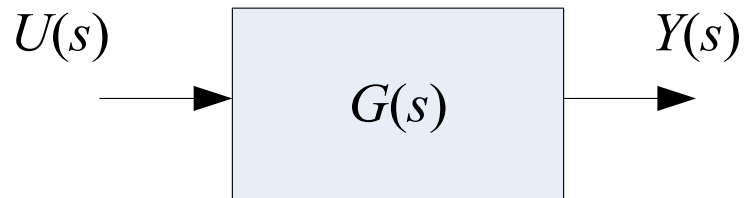
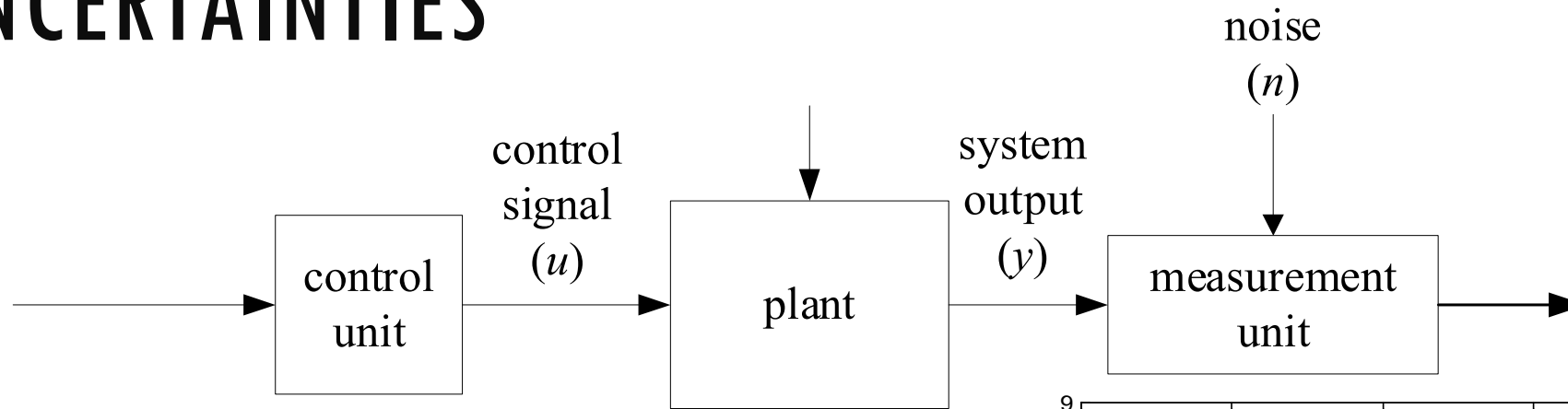
OPEN LOOP CONTROL RESPONSE



OPEN-LOOP CONTROL



OPEN-LOOP CONTROL — SENSITIVE TO UNCERTAINTIES



The background of the slide features a close-up, slightly blurred image of a pen writing on a piece of paper. The paper has a grid and some handwritten numbers, including '2.5' and '2.47'. A solid blue overlay covers the entire image, and a thin white vertical line is positioned to the right of the main title.

MATLAB + PLC

Desired to use high-computing device, such as MATLAB, to control automation process

INDUSTRIAL COMMUNICATION PROTOCOLS

Historically, many industrial components have been connected through different serial fieldbus protocols such as Control Area Network (CAN), **Modbus®**, PROFIBUS® and CC-Link. In recent years, industrial Ethernet has gained popularity, becoming more ubiquitous and offering higher speed, increased connection distance, and the ability to connect more nodes. There are many different industrial Ethernet protocols driven by various industrial equipment manufacturers. These protocols include Ether-CAT®, PROFINET®, EtherNet/IP™, and Sercos® III, among others.

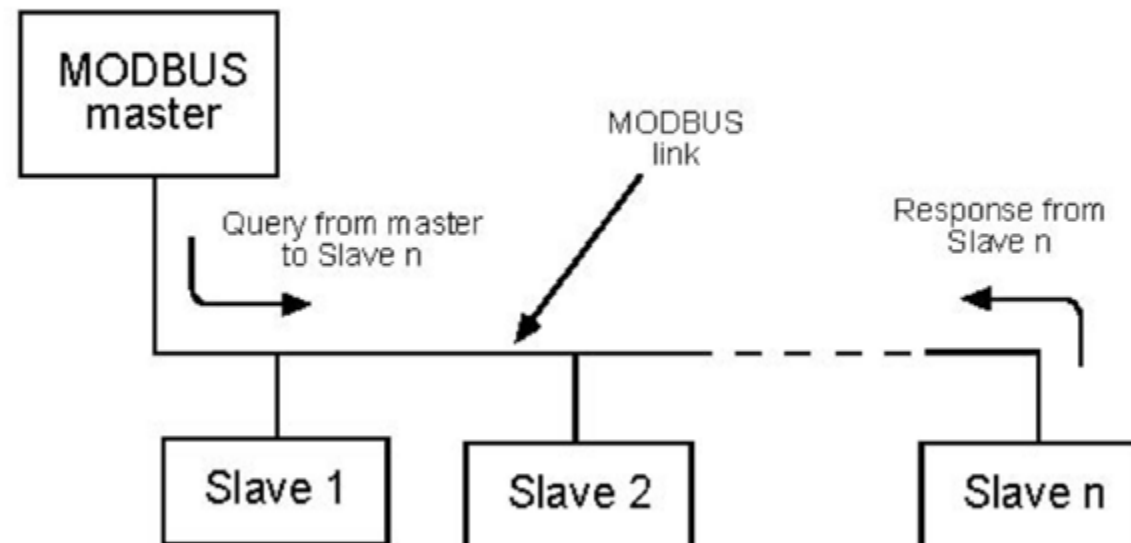
MODBUS

Modbus is a communication protocol developed by AEG-Modicon, and was devised initially for use with their Programmable Logic Controllers. It has, subsequently, become widely accepted as a communications standard, and many products have now been developed which use this protocol.

MODBUS TRANSACTIONS

Modbus controllers communicate using a master-slave technique, in which only one device (the master) can initiate a communication sequence.

Simple master - slave communication



MODBUS TRANSACTIONS

The sequence begins with the master issuing a request or command on to the bus (a 'query'), which is received by the slaves. The slaves respond by:

- 1) Taking appropriate action,
- 2) Supplying requested data to the master, or
- 3) Informing the master that the required action could not be carried out.

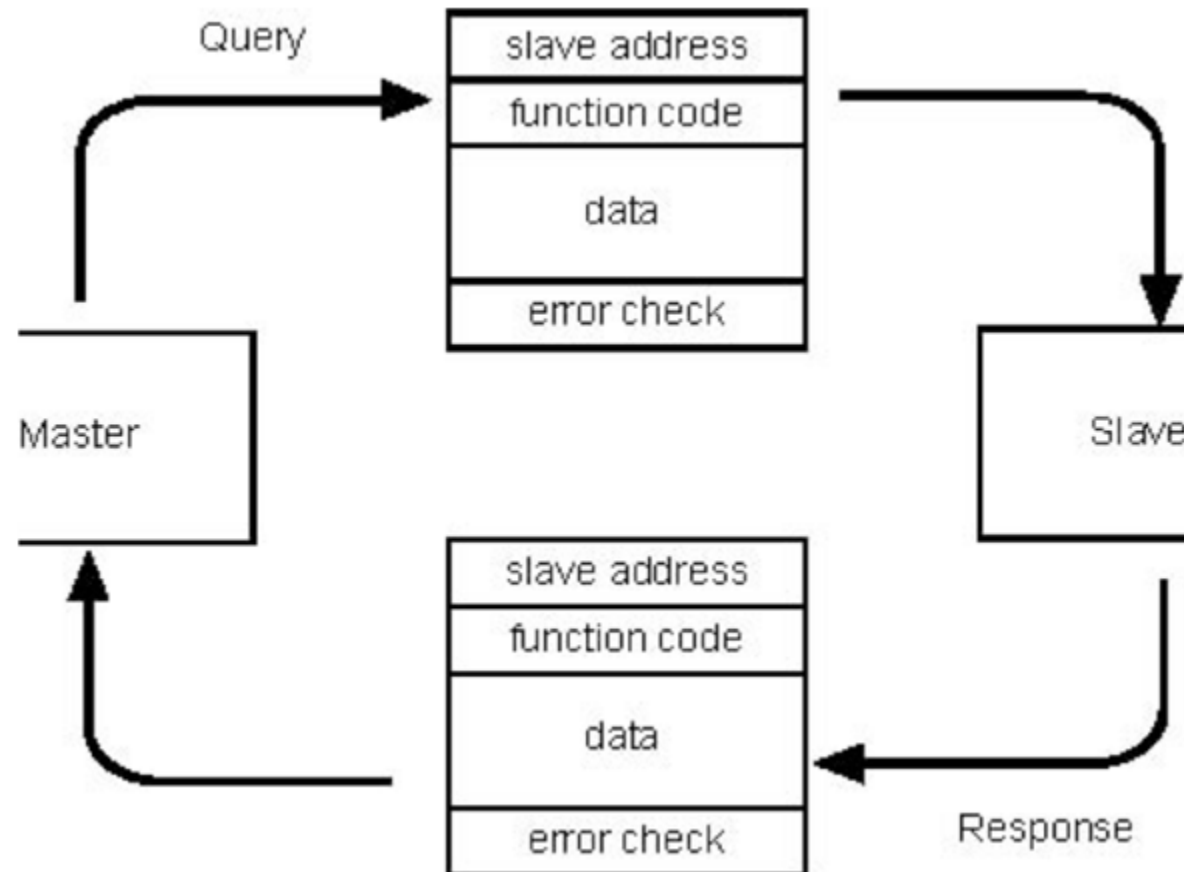
The master can address individual slaves or can transmit a message to be received by all slaves - through a 'broadcast' message.

When a slave receives a message addressed specifically to that slave, it will return a message to the master called a 'response'. The response confirms:

1. that the message was received, understood and acted upon, or
2. informs the master that the action required could not be carried out.

THE QUERY-RESPONSE CYCLE

The query-response cycle forms the basis of all communication on a Modbus network. In all situations it is the master that initiates the query and the slave that responds.



THE QUERY

The query is made up of four parts: the device address; the function code; eight bit data bytes; and an error check.

The ***device address*** - uniquely identifies a particular slave or indicates that the message is a 'broadcast' addressed to all slaves.

The ***function code*** - tells the slave what type of action to perform.

The ***data*** - bytes contain any data that the slave will require to carry out the requested function (this may be a register address within the slave, a value to be used by the slave, etc.).

The ***error check*** - field allows the slave to confirm the integrity of the message received from the master. If an error is detected, the slave ignores the query and waits for the next query to be addressed to that slave.

THE RESPONSE

A slave will normally be required to provide a response (when a query has been addressed to that slave

specifically, and not broadcast to all slaves), which will have the same overall structure format as was used for

the query: a device address; a function code; eight bit data bytes; and an error check.

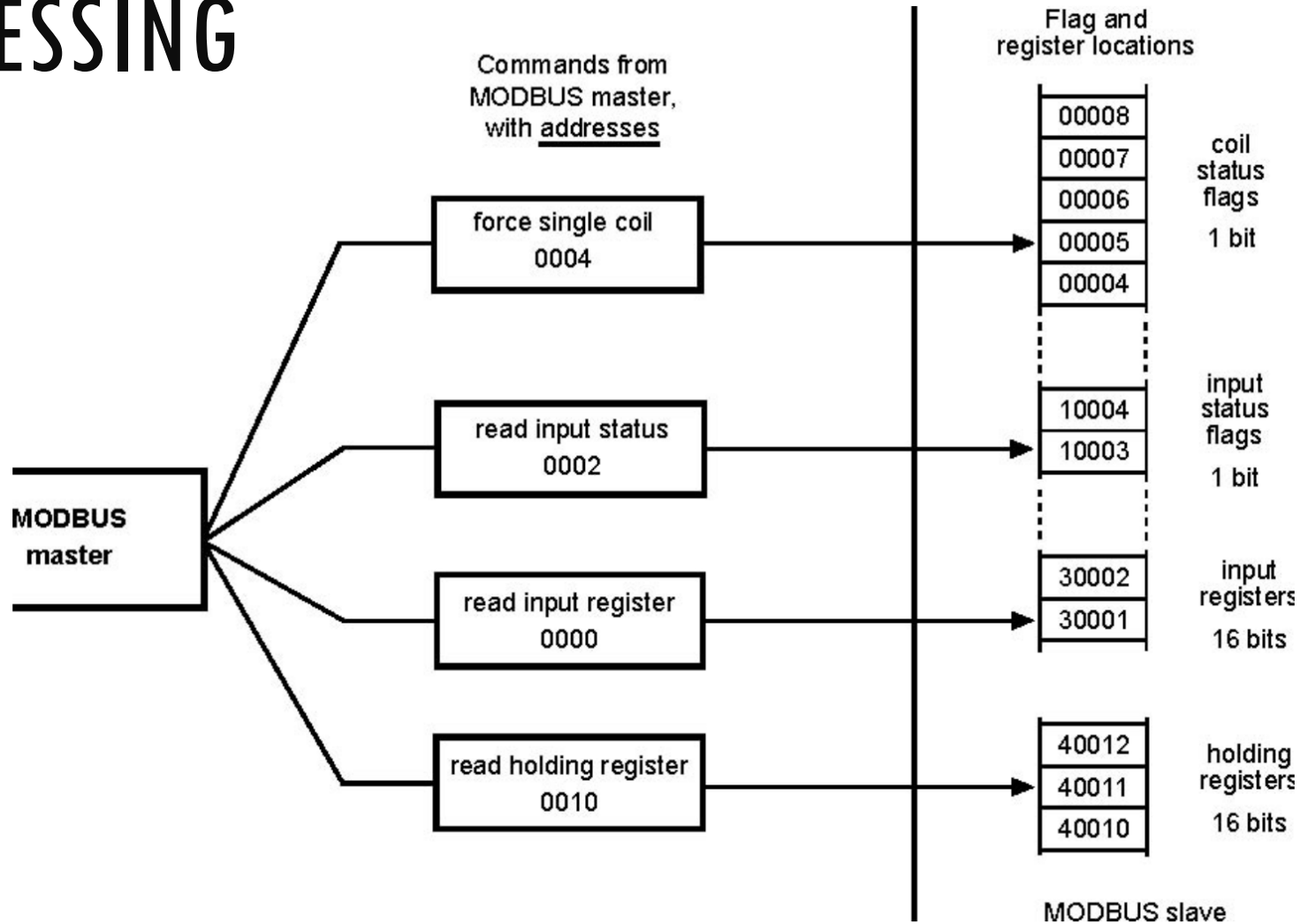
The ***device address*** - in the response is that of the addressed slave. This indicates to the master which slave is replying to its query, and allows it to confirm that the correct slave is responding.

The ***function code*** - in the response is normally an exact copy of the function code in the query, and will only vary if the slave is unable to carry out the requested function. In such circumstances the function code returned is a modified form of the original code - this then indicates which function the slave was unable to perform.

The ***data*** - contain any data requested in the query.

The ***error check*** - allows the master to confirm the integrity of the message received from the slave - if the error check is not correct, the response is ignored

REGISTER AND FLAG ORGANIZATION AND ADDRESSING





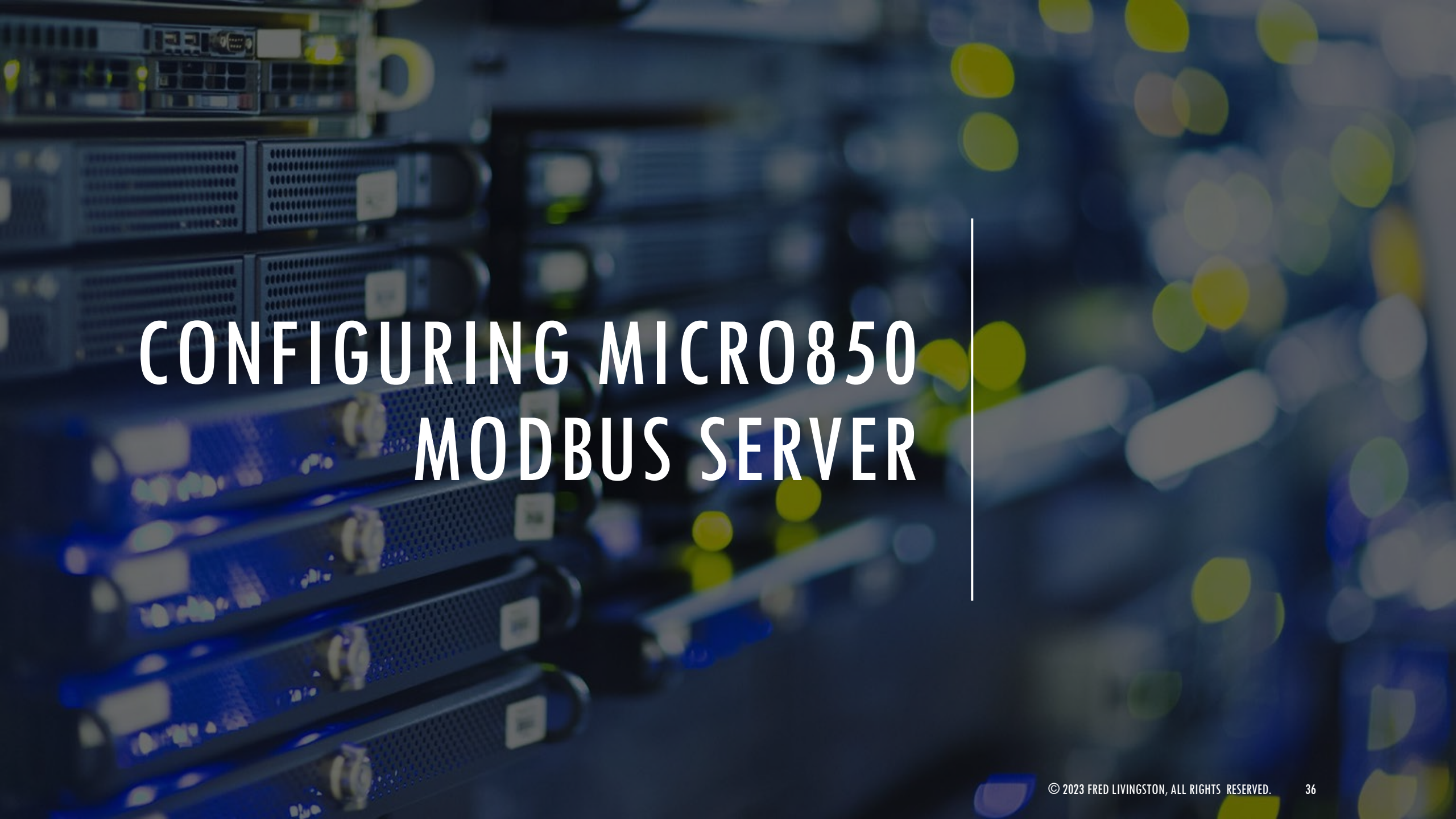
LABORATORY 3A

LAB 3A — PRE LAB

Objective:

Using the Modbus Protocol, design and implement a program, to enable an output device on the PLC

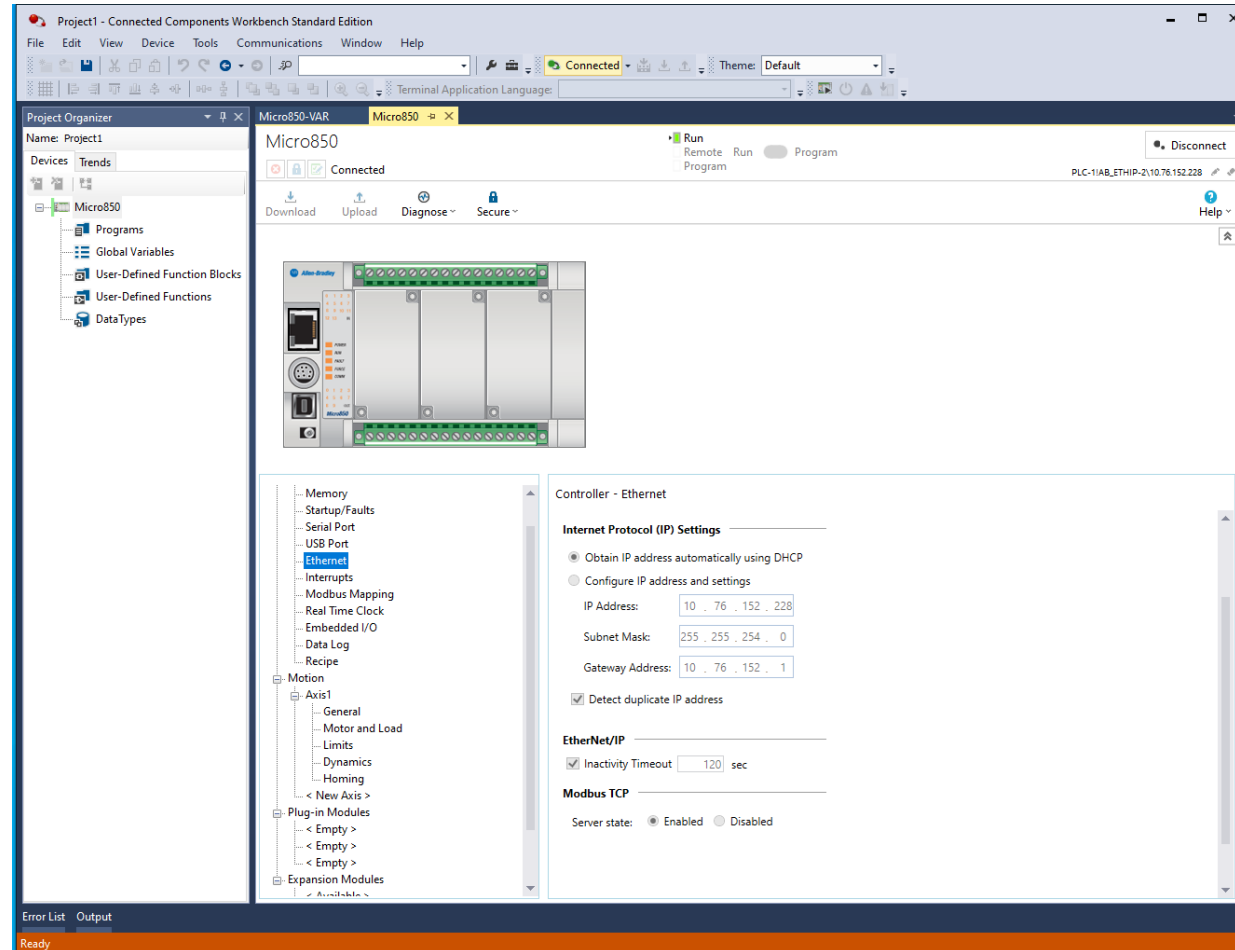
In Laboratory 3B we will expand this concept to develop an open-loop speed controller for a DC motor. NOTE: Formal report will be due at this time.



CONFIGURING MICRO850 MODBUS SERVER

MICRO 850 (2080-LC50-24QBB)

Enable Modbus Server



HOLDING REGISTERS

MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007

	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046

		...
	MB_REAL.30	HDR_40099
		HDR_40100

Project1 - Connected Components Workbench Standard Edition

File Edit View Device Tools Communications Window Help

Connected Theme: Default

Terminal Application Language:

Project Organizer

Name: Project1

Devices Trends

Micro850

Programs

Global Variables

User-Defined Function Blocks

User-Defined Functions

DataTypes

Name	Alias	Logical Value	Physical Value	Initial Value	Lock	Data Type	Dimension	Comment	String Size
MB_INT						INT	[1..40]		
MB_INT[1]		0	N/A			INT			
MB_INT[2]		0	N/A			INT			
MB_INT[3]		0	N/A			INT			
MB_INT[4]		0	N/A			INT			
MB_INT[5]		0	N/A			INT			
MB_INT[6]		0	N/A			INT			
MB_INT[7]		0	N/A			INT			
MB_INT[8]		0	N/A			INT			
MB_INT[9]		0	N/A			INT			
MB_INT[10]		0	N/A			INT			
MB_INT[11]		0	N/A			INT			
MB_INT[12]		0	N/A			INT			
MB_INT[13]		0	N/A			INT			
MB_INT[14]		0	N/A			INT			
MB_INT[15]		0	N/A			INT			
MB_INT[16]		0	N/A			INT			
MB_INT[17]		0	N/A			INT			
MB_INT[18]		0	N/A			INT			
MB_INT[19]		0	N/A			INT			
MB_INT[20]		0	N/A			INT			
MB_INT[21]		0	N/A			INT			
MB_INT[22]		0	N/A			INT			
MB_INT[23]		0	N/A			INT			
MB_INT[24]		0	N/A			INT			
MB_INT[25]		0	N/A			INT			
MB_INT[26]		0	N/A			INT			
MB_INT[27]		0	N/A			INT			
MB_INT[28]		0	N/A			INT			

Error List Output

HOLDING REGISTERS

MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007

	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046

		...
	MB_REAL.30	HDR_40099
		HDR_40100

Project1 - Connected Components Workbench Standard Edition

File Edit View Device Tools Communications Window Help

Connected Theme: Default

Terminal Application Language:

Project Organizer

Name: Project1

Devices Trends

Micro850

- Programs
- Global Variables
- User-Defined Function Blocks
- User-Defined Functions
- DataTypes

Name	Alias	Logical Value	Physical Value	Initial Value	Lock	Data Type	Dimension	Comment	String Size
_MOTION_DIAG		<input type="checkbox"/>	MOTION_DIAG			
Axis1		<input type="checkbox"/>	AXIS_REF			
MB_INT		<input type="checkbox"/>	INT	[1..40]		
MB_INT[1]		1	N/A		<input type="checkbox"/>	INT			
MB_INT[2]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[3]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[4]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[5]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[6]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[7]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[8]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[9]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[10]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[11]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[12]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[13]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[14]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[15]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[16]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[17]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[18]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[19]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[20]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[21]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[22]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[23]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[24]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[25]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[26]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[27]		0	N/A		<input type="checkbox"/>	INT			
MB_INT[28]		0	N/A		<input type="checkbox"/>	INT			

Error List Output

Ready

Type here to search

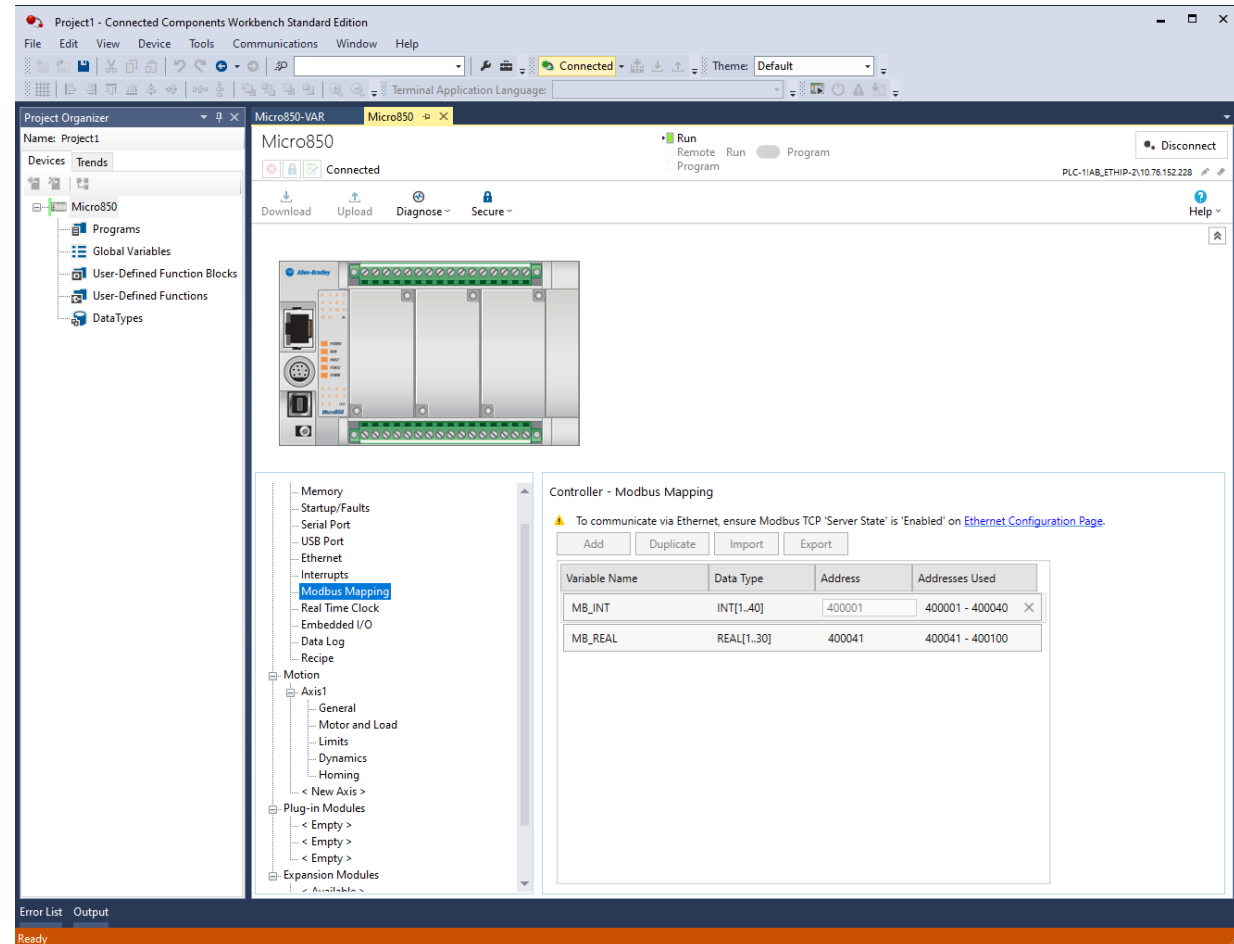
MICRO 850 (2080-LC50-24QBB)

Modbus Holding Registers

MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007

	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046

		...
	MB_REAL.30	HDR_40099
		HDR_40100



MODBUS CLIENTS

- Python Example
- MATLAB Example

MODBUS PYTHON



PYMODBUS3 LIBRARY

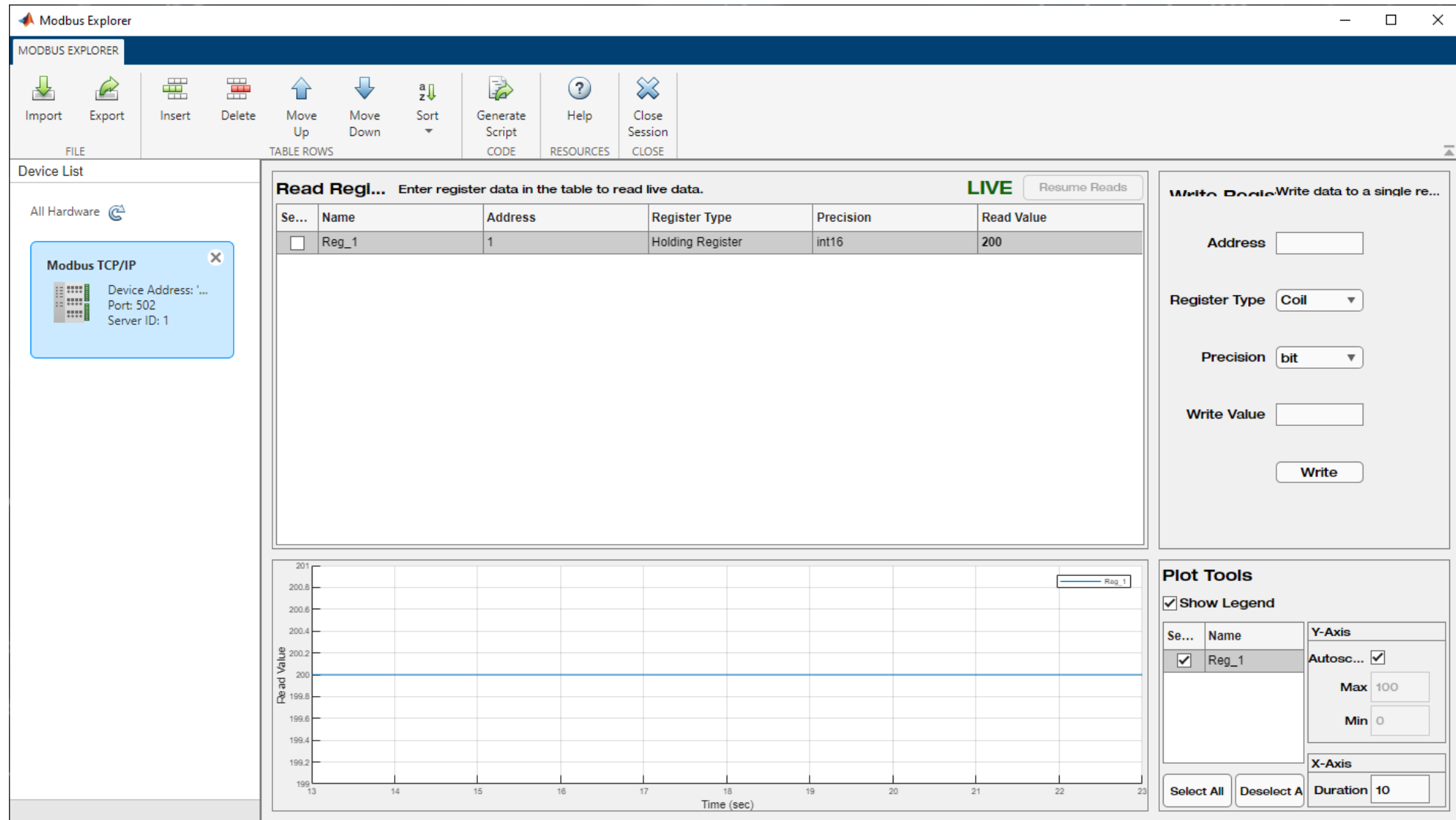
- <https://pymodbus.readthedocs.io/en/latest/index.html>
- `pip install -U pymodbus3`

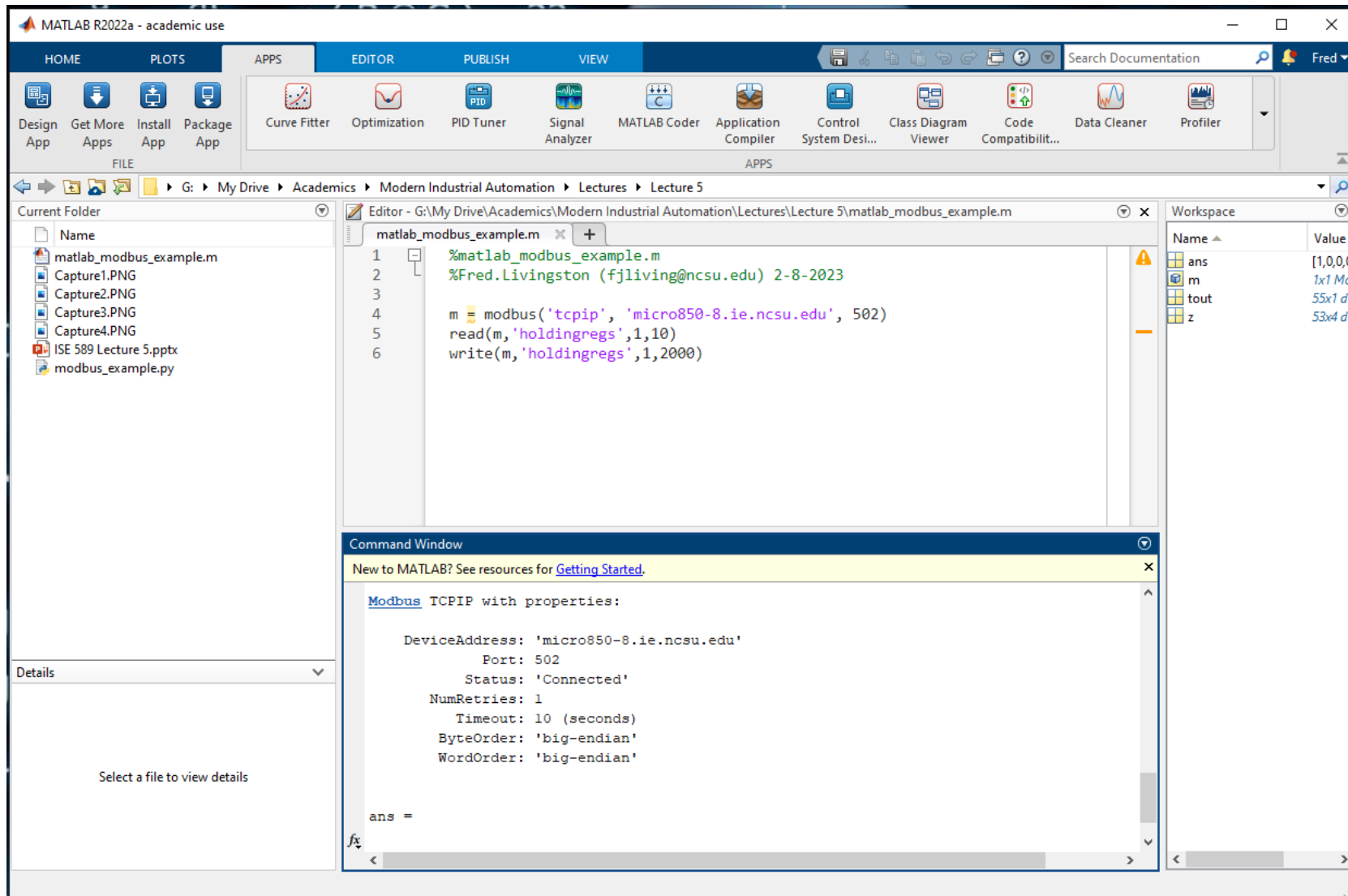
MODBUS PYTHON EXAMPLE

```
modbus_example.py x
1  #!/usr/bin/env python3
2  # modbus_example.py
3  # Fred Livingston (fjlliving@ncsu.edu) 2-8-2023
4
5  from pymodbus.constants import Endian
6  from pymodbus.payload import BinaryPayloadDecoder
7  from pymodbus.payload import BinaryPayloadBuilder
8  from pymodbus.client import ModbusTcpClient
9
10 client = ModbusTcpClient('micro850-8.ie.ncsu.edu')
11 client.connect()
12
13 # Read from Holding Registers
14 request = client.read_holding_registers(1,1)
15 result = request.registers
16 decoder = BinaryPayloadDecoder.fromRegisters(result, Endian.Big, wordorder=Endian.Big)
17
18 # 'string': decoder.decode_string(8),
19 # 'float': decoder.decode_32bit_float(),
20 # '16uint': decoder.decode_16bit_uint(),
21 # 'ignored': decoder.skip_bytes(2),
22 # '8int': decoder.decode_8bit_int(),
23 # 'bits': decoder.decode_bits(),
24 print("Value: %d" % decoder.decode_16bit_uint())
25
26
27 # Write to Hold Registers
28 client.write_registers(1,2000)
29
30 client.close()
```


MODBUS MATLAB







GROUP ASSIGNMENT

Using the Modbus Protocol, design and implement a program, to enable an output device on the PLC



END OF LECTURE