Cairo University
Faculty of Engineering
Mech. Power Department

ACC Virtual Labs
Automatic Control Circuits & Virtual Labs for Mechanical Power Systems

دبلوم تطبيقات التحكم الأوتوماتيكى فى نظم القوى الميكانيكية

MEP 599 Diploma Design Project-Spring Term 2017/2018

Automatic control of liquid filling machine using PLC

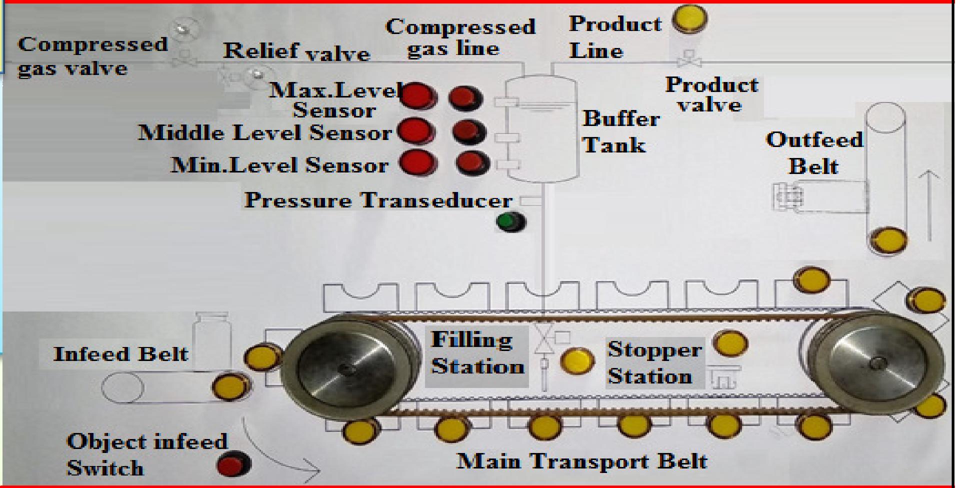
by Eng. Ahmad Mohamed Abdel Hai Fouda

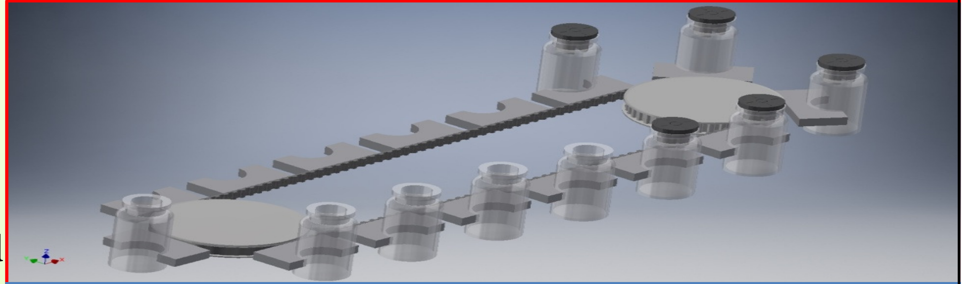
Supervised by

Assoc.Prof.Mohsen S.Soliman,ACC Manager&Dr.Amro Abdel Raouf

Mechanical Power Engineering Department

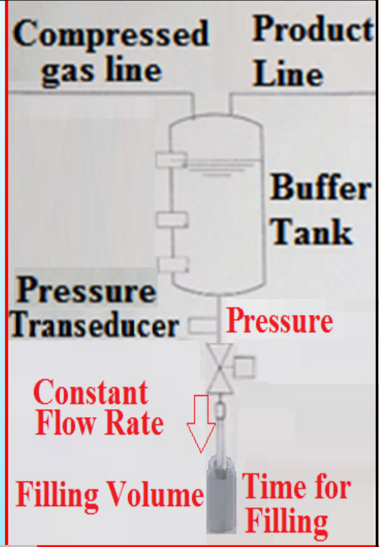
Abstract: The goal of project is to develop a control scheme to an automated sequential filling process of liquids into vials used in pharmaceutical fields incorporating the time/pressure technique & intermittent motion transport system, using a PLC. Different filling techniques&the time/pressure filling method are discussed then the development of full control system is done including the code used &how to use Mode bus TCP in order to communicate with different systems&expand I/O of PLC. The project included a practical simulation Kit with stepper motor(as shown in upper fig.) to realize the system &experimentally investigate/test both the design & simulation of a control system for automation of liquid filling using time-pressure filling method. The PLC model used is Siemens S7-1214. A motion control unit &object tracking technique using shift registers were used for quality control & to synchronize the whole process between filling station, stopper station and object transport belt. Arduino microprocessor board and Mod-bus TCP were used as cheap alternative I/O expansion module for the PLC. Finally, a simple HMI was also designed to operate simulation model on PC. Documentation includes full flow charts and the control LAD for each step in TPF process.



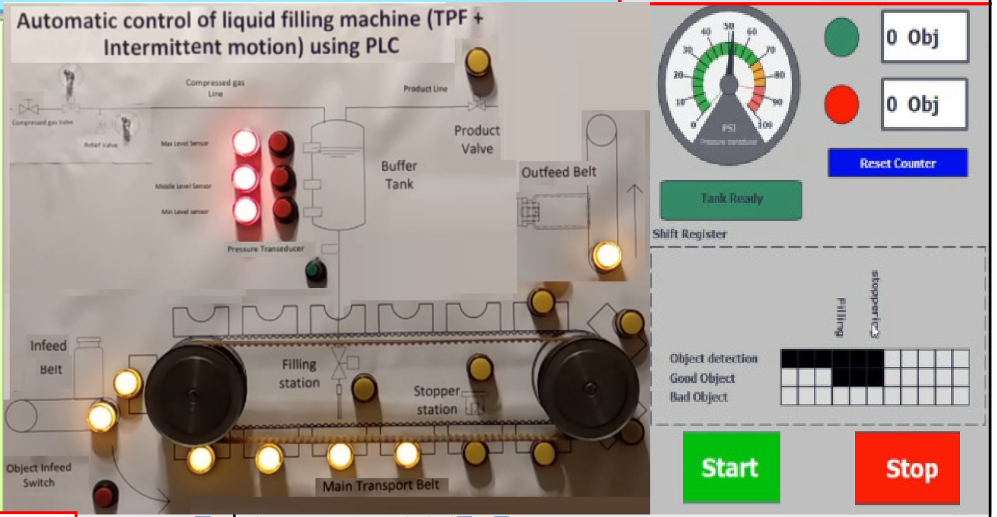


Vial filling and closing machines

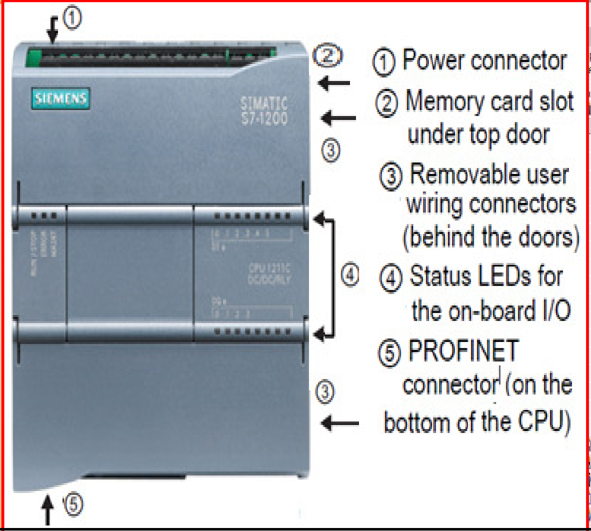
Time-Pressure Filling TPF Principle: A liquid of given temperature & viscosity will flow at constant rate through fixed opening, providing the pressure is constant. Having established a flow rate, time is the necessary variable to produce a specified fill volume. System pressure should be controlled to ±0.01 PSI by an inline pressure transducer located between the pressurized storage vessel and the filler. Depending on the size of the containers being filled and the speed of the filling equipment the size of buffer tank from several liters up to 20 L. To minimize the effect of head pressure in a larger Tank, automatic level control system is integrated into the system. A pressure transducer provides a signal to pressure control valve which regulates the pressurizing gas. The pressurized product flows through flexible tubing through a flow control pinch valve. Controlled by a PLC, the pinch valve will remain open for a specified time based on configurable parameters that include product viscosity, diameter of flexible tubing and the target pressure. For TP filling system critical process parameters in the process are: System pressure, Diameter of tubing, Viscosity, Equipment operating speed and Product flow rate (target fill volume X equipment speed) unites per minute.

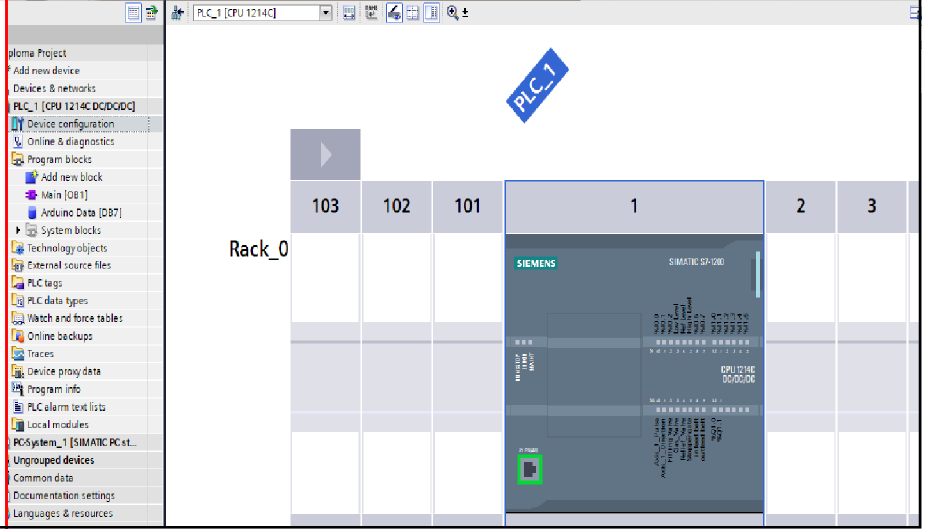


System layout& Function specification(as in fig.): Vials are supplied to main transport belt via infeed belt, Outfeed belt is used to transport Vials from main transport. An object switch is pressed to let the vials get in the main transport belt cells. A shift register is used to determine the position of all vials through the main transport. Product is fed via a valve to a level-controlled buffer tank. From the buffer tank, the distribution of the contents takes place at the filling points of the filling system. The filling is done via a time-pressure filling system. The filling station consists of the following functional units: Product tank with pressure control, Filling valve is a Pinch vale, Squeeze tube, Throttle (for setting a defined flow resistance), Filling hose and a Filling needle.

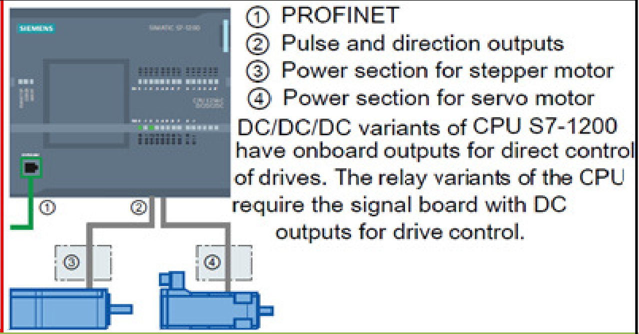


The opening time of the Pinch valve is calculated based on Pressure and fill volume.Vials are sealed by inserting a stopper using stopper station. Good and bad vials are counted. The filling will not begin until the tank is ready for production otherwise all vials will be counted bad, only filled vials will be counted as good. A PLC is used to automate the process. Arduino based microcontroller is also used as an I/O expansion module for the PLC unit.

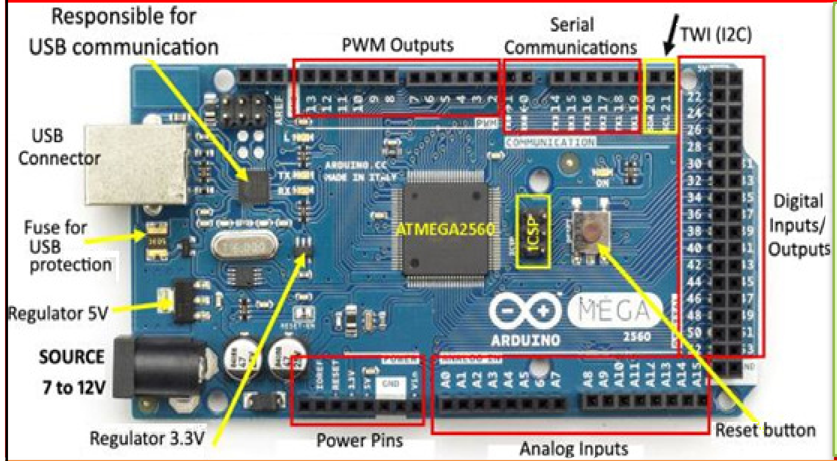

SIEMENS
SIMATIC S7-1200
① Power connector
② Memory card slot under top door
③ Removable user wiring connectors (behind the doors)
④ Status LEDs for the on-board I/O
⑤ PROFINET connector (on the bottom of the CPU)


ploma Project
Add new device
Devices & networks
PLC_1 [CPU 1214C DC/DC/DC]
Device configuration
Online & diagnostics
Program blocks
Add new block
Main [OB1]
Arduino Data [DB7]
System blocks
Technology objects
External source files
PLC tags
PLC data types
Watch and force tables
Online backups
Traces
Device proxy data
Program info
PLC alarm text lists
Local modules
PC-System_1 [SIMATIC PC st...
Ungrouped devices
Common data
Documentation settings
Languages & resources

Motion control function: PLC CPU provides a motion control function for operation of stepper motors & servomotors with pulse interface. Motion control takes over control & monitoring of the drives. The "Axis" technology object configures mechanical drive data, the drive interface, dynamic parameters & other drive properties. You configure pulse & direction outputs of CPU for controlling the drive. Your user program uses the motion control instructions to control the axis and to initiate motion tasks. Use the PROFINET interface to establish the online connection between the CPU and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.


① PROFINET
② Pulse and direction outputs
③ Power section for stepper motor
④ Power section for servo motor

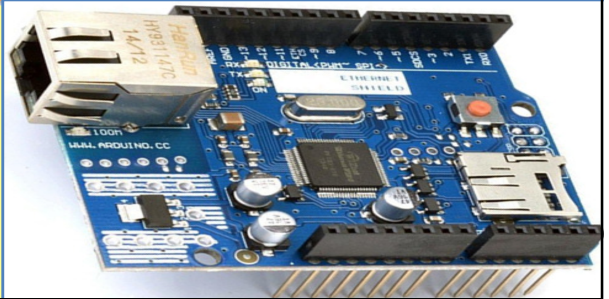
DC/DC/DC variants of CPU S7-1200 have onboard outputs for direct control of drives. The relay variants of the CPU require the signal board with DC outputs for drive control.



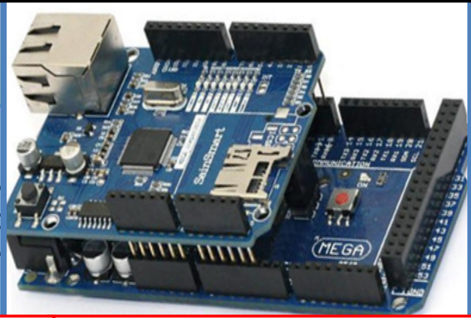
Arduino: is open-source platform used for building electronic projects. Arduino consists of both a programmable circuit board (microcontroller)& a software, or IDE (Integrated Development Environment)that runs on PC, used to write and upload code to the board. Arduino board designs use variety of microprocessors &controllers. The boards are equipped with sets of digital &analog input/output (I/O) pins that may be interfaced to various expansion boards or Breadboards (*shields*)& other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from PCs. The microcontrollers are typically programmed using a dialect of features from programming languages C &C++. In addition to using traditional compiler tool-chains, Arduino project provides an integrated development environment (IDE) based on the Processing language project.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller with varying amounts of flash memory, pins, and features. The 32-bit Arduino Due, based on Atmel SAM3X8E introduced in 2012.The boards use single, double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*. Multiple and possibly stacked shields may be individually addressable via an I²Cserial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to on-chip flash memory. The default boot-loader of Arduino UNO is opti-boot boot-loader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, as Arduino Mini &unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of Arduino IDE, standard AVR in-system programming (ISP) is used.

Arduino Mega 2560 Specifications: It has 54 digital input/output pins (of which 15 can be used as Pulse-Width Modulation (PWM) outputs), 16 analog inputs, 4 Universal Asynchronous Receiver-Transmitter (UART), a 16 MHz crystal oscillator, Universal Serial Bus (USB) connection, a power jack, ICSP header, and a reset button. It contains everything needed to support the microcontroller simply connect it to a computer with a USB cable or power it with an Alternating Current / Direct Current (AC/DC) adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Arduino Uno. Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header).



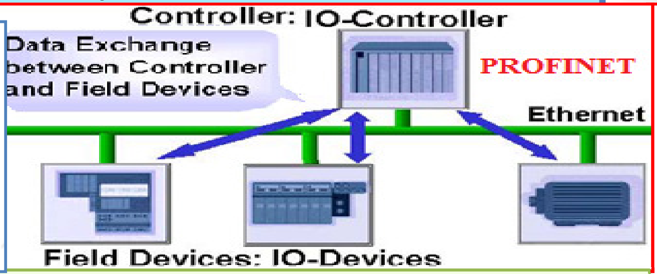
Mod-busTCP with Arduino &Ethernet shield: This device is intended to be an output expansion for the PLC, it will receive data via Mod-bus connection.

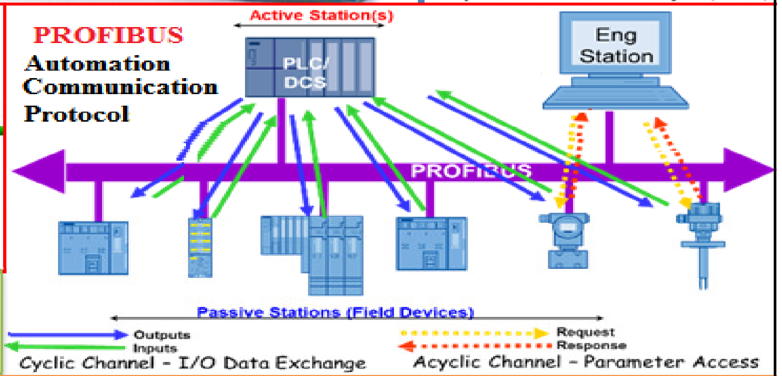


Relay Module used in the Project (3units)

Relay Module: it is a LOW Level 5V 4-channel relay interface board,& each channel needs a 15-20mA driver current. It is used to control various appliances&equipment with large current. It is equipped with high-current relays work under AC250V 10A or DC30V 10A. It has a standard interface that controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement & prevent ground loop when interface to microcontroller.

Pressure Transeducer

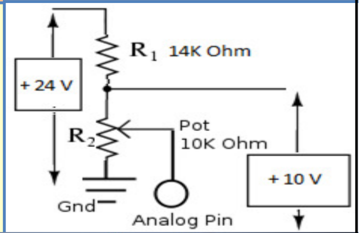

Controller: IO-Controller
Data Exchange between Controller and Field Devices
PROFINET
Ethernet
Field Devices: IO-Devices


PROFIBUS Automation Communication Protocol
Active Station(s)
PLC/ DCS
Eng Station
PROFIBUS
Passive Stations (Field Devices)
Outputs
Inputs
Cyclic Channel - I/O Data Exchange
Request
Response
Acyclic Channel - Parameter Access

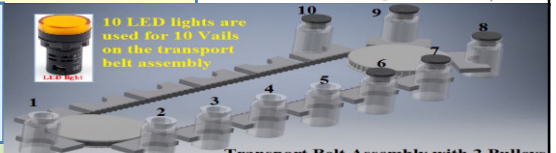
Ethernet switch: It is used as a connection between the PLC and the Arduino and the PC station (HMI) for data exchange.

Pressure transducer (or transmitter): is a transducer converts pressure into an analog electrical signal. One of the most common type is strain-gage type. The conversion is achieved by deformation of strain gages bonded into the diaphragm of the transducer and wired into a Wheatstone bridge. Pressure applied produces a strain to the gage. The strain produces resistance change proportional to pressure.

Voltage divider circuit: Simulation of a pressure transducer is done by a potentiometer, but since analog input of PLC is 0-10V & the supply voltage is 24V, a voltage divider circuit was used to limit the voltage across the potentiometer connected to the PLC input. Voltage dividers find wide application in electric meter circuits, where specific combinations of series resistors are used to "divide" a voltage into precise proportions as part of a voltage measurement device. As shown, if The voltage between R2, the divided voltage, will be % of the input voltage.

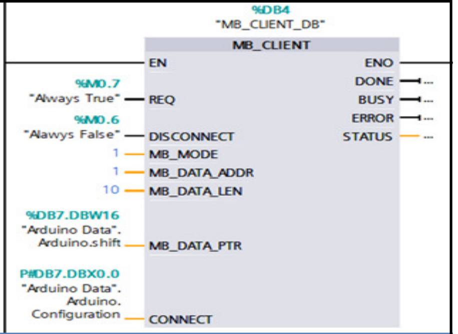


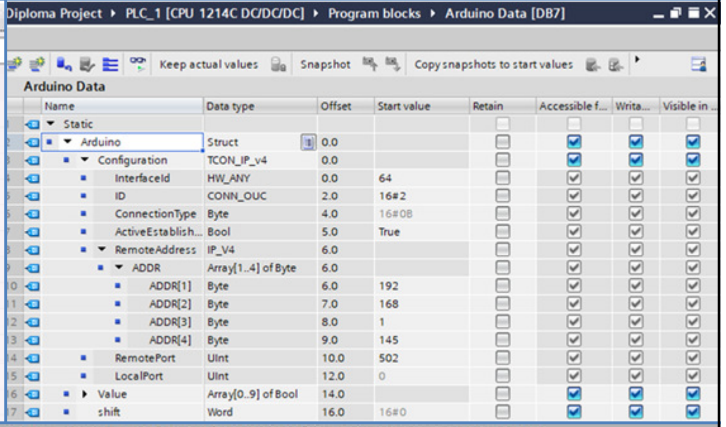
Transport belt assembly: A timing belt T5/750 & two M40 pulleys was used in the assembly to simulate the transport of the vials during the filling and the stoppering processes. As on the figure, Ten led lights were used to simulate movement of vials on transport belt assembly & to track the current position of the vials visually.


10 LED lights are used for 10 Vails on the transport belt assembly
LED light
1
2
3
4
5
6
7
8
9
10
Transport Belt Assembly with 2 Pulleys

Modbus TCP/IP: is Modbus RTU protocol with TCP interface that runs on Ethernet. Modbus messaging structure is *application protocol* that defines rules for organizing &interpreting data independent of data transmission medium. TCP/IP refers to Transmission Control Protocol/ Internet Protocol, which provides transmission medium for Modbus TCP/IP messaging. TCP/IP allows blocks of binary data to be exchanged between computers. It is a world-wide standard serves as foundation for World Wide Web. Primary function of TCP is to ensure that all packets of data are received correctly, while IP makes sure that messages are correctly addressed &routed. Note that TCP/IP is merely a *transport protocol,* &does not define what the data means or how it to be interpreted (this is the job of the application protocol, Modbus in this case).So, Modbus TCP/IP uses TCP/IP and Ethernet to carry the dataof the Modbus message structure between compatible devices. That is,Modbus TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data (Modbus as application protocol). Essentially, Modbus TCP/IP message is simply a Modbus communication encapsulated in an Ethernet TCP/IP wrapper.

How to create a connection between PLC and Arduino: In order to establish a connection between the PLC and Arduino first you have to configure PLC to use Profinet interface for the Mod-bus connection. First you have add the shown instruction to a new LAD network, go to instructions>communication>others>Modbus TCP>Mod-bus client. Then define: *(MB_Mode) is the mode of the Mod-bus request (read, write or diagnostics) in our case it will write (1). *(MB_DATA_ADDR) is address of the remote of Arduino. *(MB_DATA_LEN) is length of bits /words to be written. *(MB_DATA_PTR) is data to be sent to arduino, which is the shift-register bits. The combination of the MB_MODE, MB_DATA_ADDR and MB_DATA_LEN parameters define the Mod-bus function code used. The current combination is Function 15 (write multiple coils). After this we create a data block so we can write the remote IP configuration of the Arduino. We go to program blocks>Add new Block, then we right click on the new created block and uncheck (optimized block access). Then we add the shown configuration. And we compile the block. After that add the configuration to (connect) of the Mod-bus instruction.



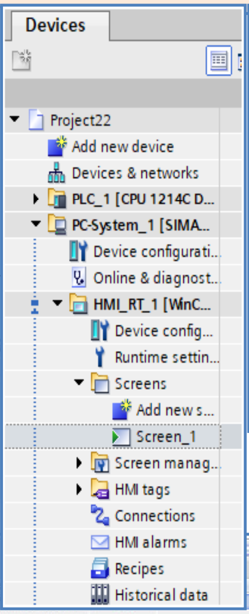

Diploma Project ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Arduino Data [DB7]

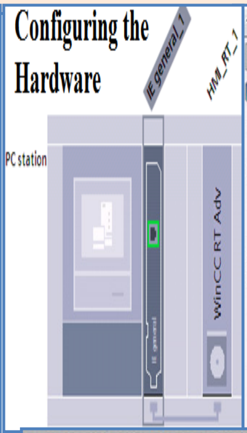
Keep actual values   Snapshot   Copy snapshots to start values

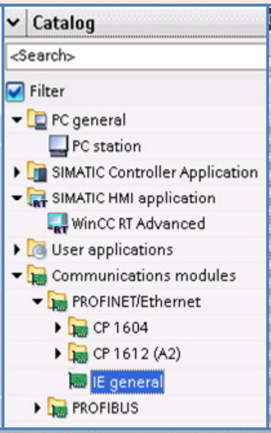
Arduino Data

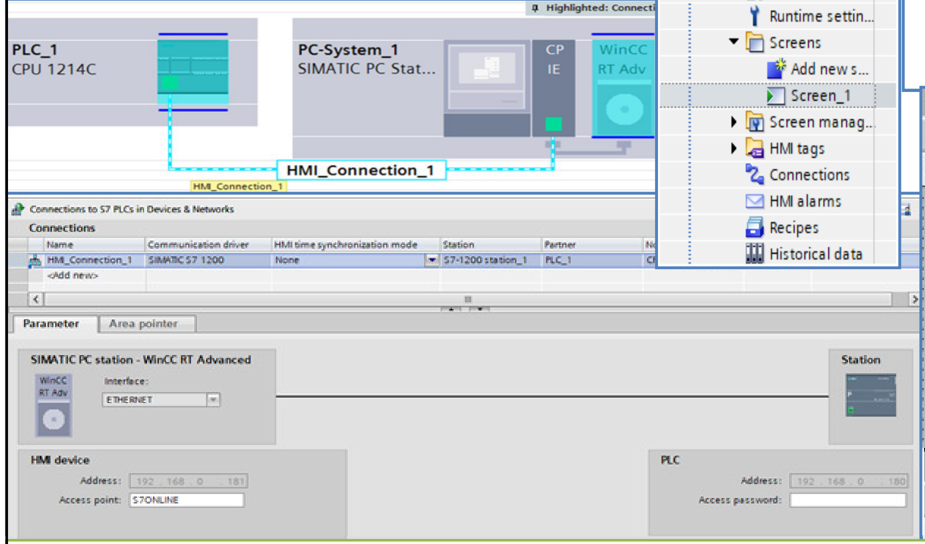
	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...
1	▾ Static							
2	▾ Arduino	Struct	0.0			✓	✓	✓
3	▾ Configuration	TCON_IP_v4	0.0			✓	✓	✓
4	InterfaceId	HW_ANY	0.0	64		✓	✓	✓
5	ID	CONN_OUC	2.0	16#2		✓	✓	✓
6	ConnectionType	Byte	4.0	16#0B		✓	✓	✓
7	ActiveEstablish...	Bool	5.0	True		✓	✓	✓
8	▾ RemoteAddress	IP_V4	6.0			✓	✓	✓
9	▾ ADDR	Array[1..4] of Byte	6.0			✓	✓	✓
10	ADDR[1]	Byte	6.0	192		✓	✓	✓
11	ADDR[2]	Byte	7.0	168		✓	✓	✓
12	ADDR[3]	Byte	8.0	1		✓	✓	✓
13	ADDR[4]	Byte	9.0	145		✓	✓	✓
14	RemotePort	UInt	10.0	502		✓	✓	✓
15	LocalPort	UInt	12.0	0		✓	✓	✓
16	▸ Value	Array[0..9] of Bool	14.0			✓	✓	✓
17	shift	Word	16.0	16#0		✓	✓	✓

HMI (Human Machine Interface) generated on PC by Wincc: Wincc Advanced (Windows Control Center) is windows based software for configuring panels & PC based HMI's, WinCC flexible RT (Real Time) is used for PC based solutions. It is Integrated in TIA Portal (WinCC RT Advanced). Some features of WinCC are: *Operator system works under Windows OS. *Visualization and control of the process . *Alarm and event handling. *Trends.









Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	HMI time synchronization mode	Station	Partner
HMI_Connection_1	SIMATIC S7 1200	None	S7-1200 station_1	PLC_1
<Add new>				

Parameter | Area pointer

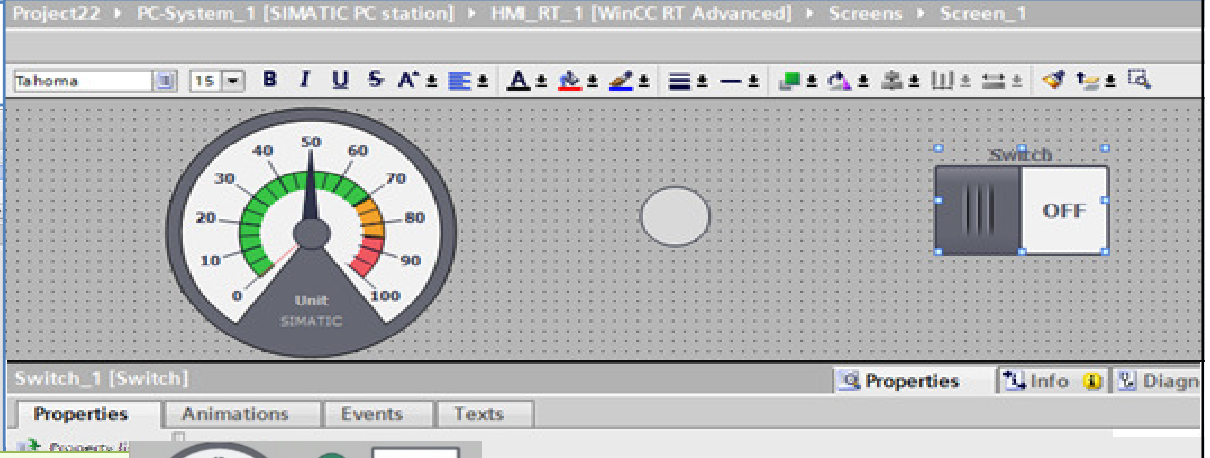
SIMATIC PC station - WinCC RT Advanced

Interface: ETHERNET

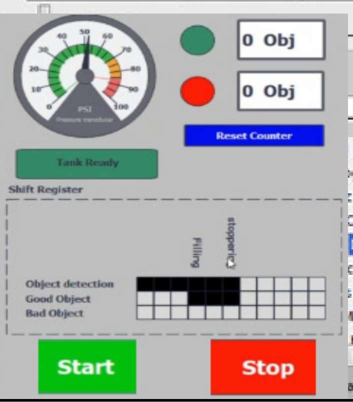
Station

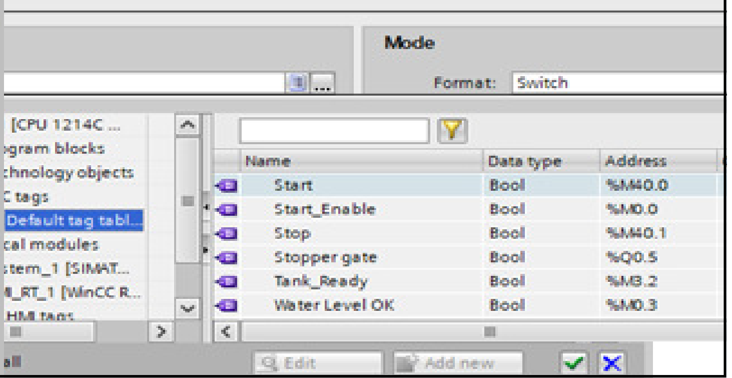
HMI device
Address: 192 . 168 . 0 . 181
Access point: S7ONLINE

PLC
Address: 192 . 168 . 0 . 180
Access password:



System procedure and to how to operate the Simulation kit: In order to operate the kit you do these steps: 1-Switch on the main power for the kit. 2-Power on the PC station. 3-Launch the Wincc RT (HMI) as in fig. 4-Press the start Button, the main transport belt will begin moving. 5-Rest the counter. 6-Product Valve will open automatically if the level sensor are not energized. 7-Press the switches for level sensors in the following order (low, Ref, High). 8-The product valve will close. 9-Adjust pressure by turning the potentiometer knob so that Tank is ready for production. If the pressure is low, compressed gas valve will open. If the pressure is high, the relief valve will open. Set point should be between 40-50 PSI. 10-Press object infeed switch to let the vials enter (writes 1 bit to the shift register every belt movement). 11-Filling and stoppering station will begin filling when the vials reach the appropriate position. 12-You can track the vials through the shift register on the HMI or through the LEDs on the kit.

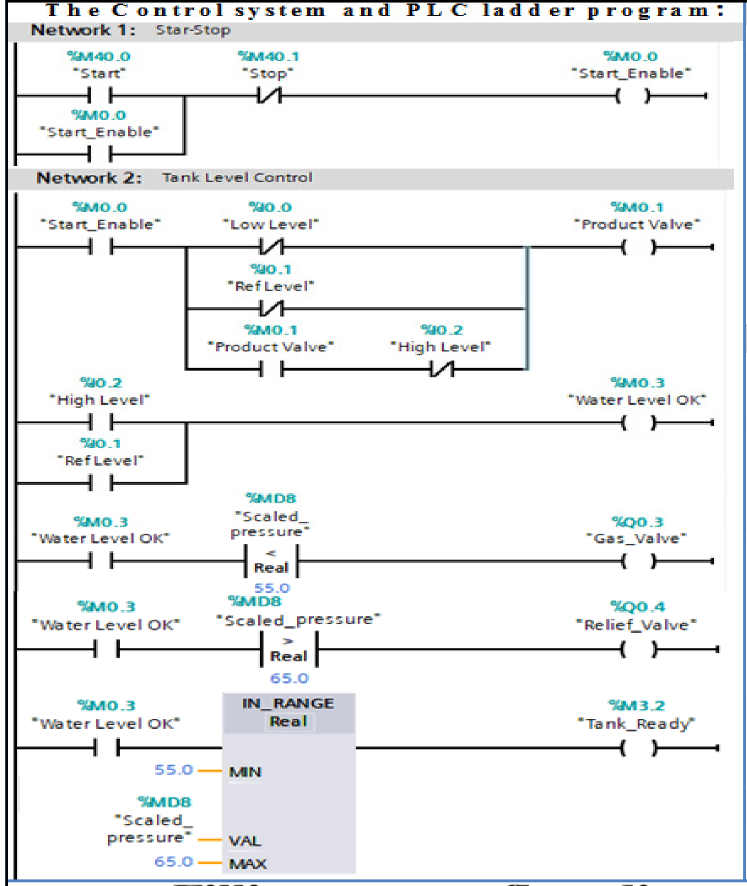


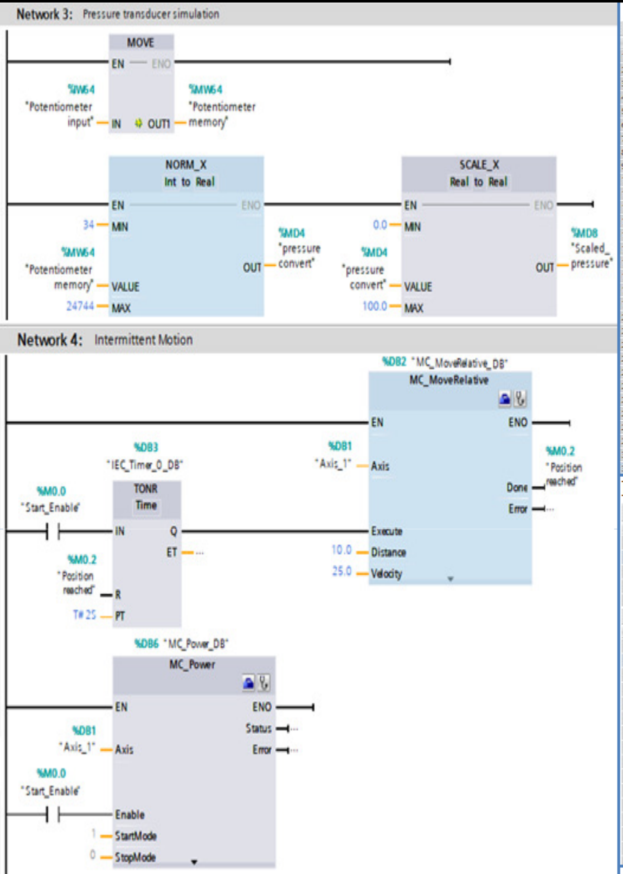

Mode

Format: Switch

Name	Data type	Address
Start	Bool	%M40.0
Start_Enable	Bool	%M0.0
Stop	Bool	%M40.1
Stopper gate	Bool	%Q0.5
Tank_Ready	Bool	%M3.2
Water Level OK	Bool	%M0.3

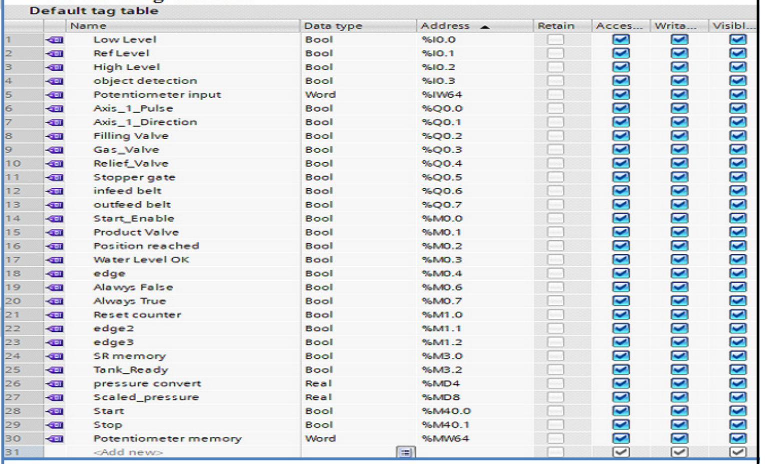
The Control system and PLC ladder program:





Network 3: Pressure transducer simulation
MOVE
EN ENO
%IW64 "Potentiometer input" IN
OUT1 %MW64 "Potentiometer memory"
NORM_X Int to Real
EN ENO
34 MIN
%MW64 "Potentiometer memory" VALUE
24744 MAX
OUT %MD4 "pressure convert"
SCALE_X Real to Real
EN ENO
0.0 MIN
%MD4 "pressure convert" VALUE
100.0 MAX
OUT %MD8 "Scaled_ pressure"
Network 4: Intermittent Motion
%DB2 "MC_MoveRelative_DB"
MC_MoveRelative
EN ENO
%DB1 "Axis_1" Axis
Done %M0.2 "Position reached"
Error
Execute
10.0 Distance
25.0 Velocity
%DB3 "IEC_Timer_0_DB"
TONR Time
%M0.0 "Start_Enable" IN
Q
ET
%M0.2 "Position reached" R
T# 2S PT
%DB6 "MC_Power_DB"
MC_Power
EN ENO
%DB1 "Axis_1" Axis
Status
Error
%M0.0 "Start_Enable" Enable
1 StartMode
0 StopMode

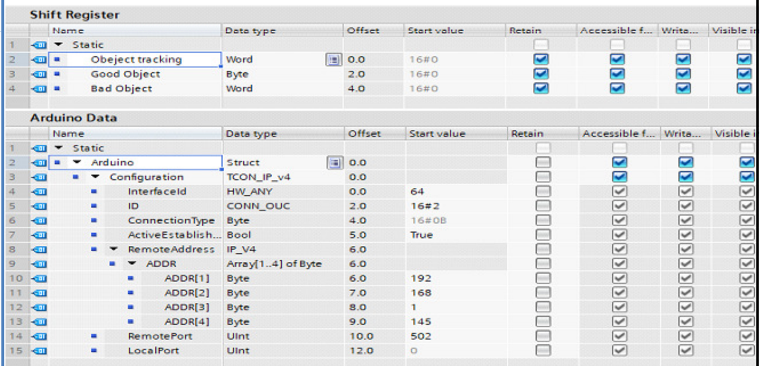
PLC Tag Table:



Default tag table

	Name	Data type	Address ▲	Retain	Acces...	Writa...	Visibl...
1	Low Level	Bool	%I0.0		✓	✓	✓
2	Ref Level	Bool	%I0.1		✓	✓	✓
3	High Level	Bool	%I0.2		✓	✓	✓
4	object detection	Bool	%I0.3		✓	✓	✓
5	Potentiometer input	Word	%IW64		✓	✓	✓
6	Axis_1_Pulse	Bool	%Q0.0		✓	✓	✓
7	Axis_1_Direction	Bool	%Q0.1		✓	✓	✓
8	Filling Valve	Bool	%Q0.2		✓	✓	✓
9	Gas_Valve	Bool	%Q0.3		✓	✓	✓
10	Relief_Valve	Bool	%Q0.4		✓	✓	✓
11	Stopper gate	Bool	%Q0.5		✓	✓	✓
12	infeed belt	Bool	%Q0.6		✓	✓	✓
13	outfeed belt	Bool	%Q0.7		✓	✓	✓
14	Start_Enable	Bool	%M0.0		✓	✓	✓
15	Product Valve	Bool	%M0.1		✓	✓	✓
16	Position reached	Bool	%M0.2		✓	✓	✓
17	Water Level OK	Bool	%M0.3		✓	✓	✓
18	edge	Bool	%M0.4		✓	✓	✓
19	Alawys False	Bool	%M0.6		✓	✓	✓
20	Always True	Bool	%M0.7		✓	✓	✓
21	Reset counter	Bool	%M1.0		✓	✓	✓
22	edge2	Bool	%M1.1		✓	✓	✓
23	edge3	Bool	%M1.2		✓	✓	✓
24	SR memory	Bool	%M3.0		✓	✓	✓
25	Tank_Ready	Bool	%M3.2		✓	✓	✓
26	pressure convert	Real	%MD4		✓	✓	✓
27	Scaled_pressure	Real	%MD8		✓	✓	✓
28	Start	Bool	%M40.0		✓	✓	✓
29	Stop	Bool	%M40.1		✓	✓	✓
30	Potentiometer memory	Word	%MW64		✓	✓	✓
31	<Add new>				✓	✓	✓

Data Blocks:



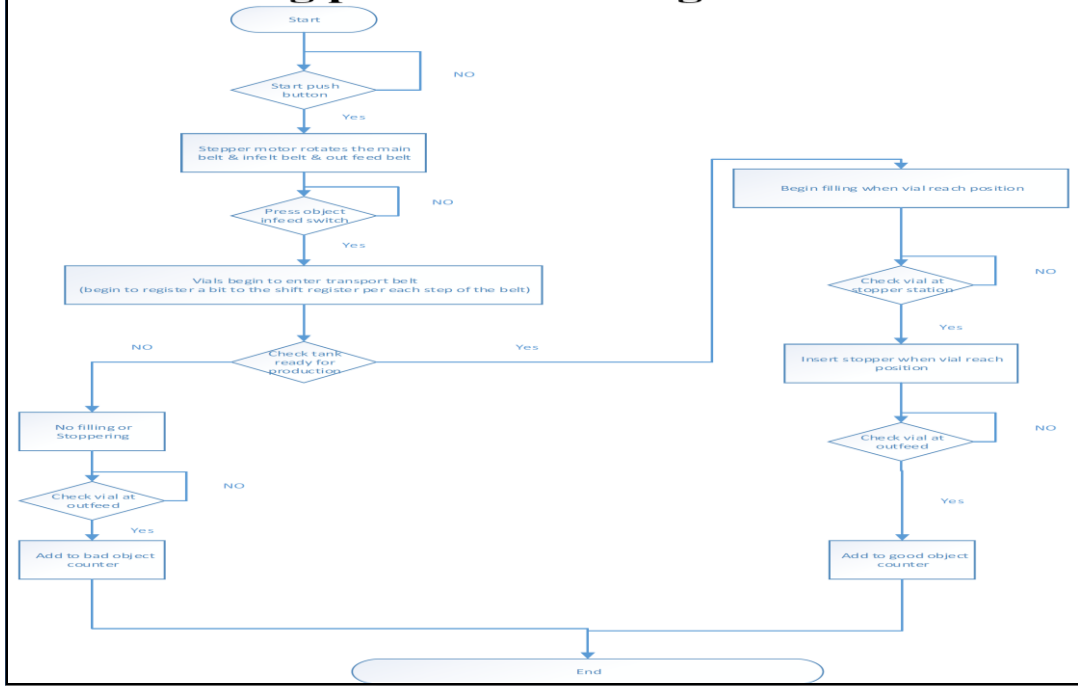
Shift Register

	Name	Data type	Offset	Start value	Retain	Accessible f...	Write...	Visible i
1	Static							
2	Obeject tracking	Word	0.0	16#0	✓	✓	✓	✓
3	Good Object	Byte	2.0	16#0	✓	✓	✓	✓
4	Bad Object	Word	4.0	16#0	✓	✓	✓	✓

Arduino Data

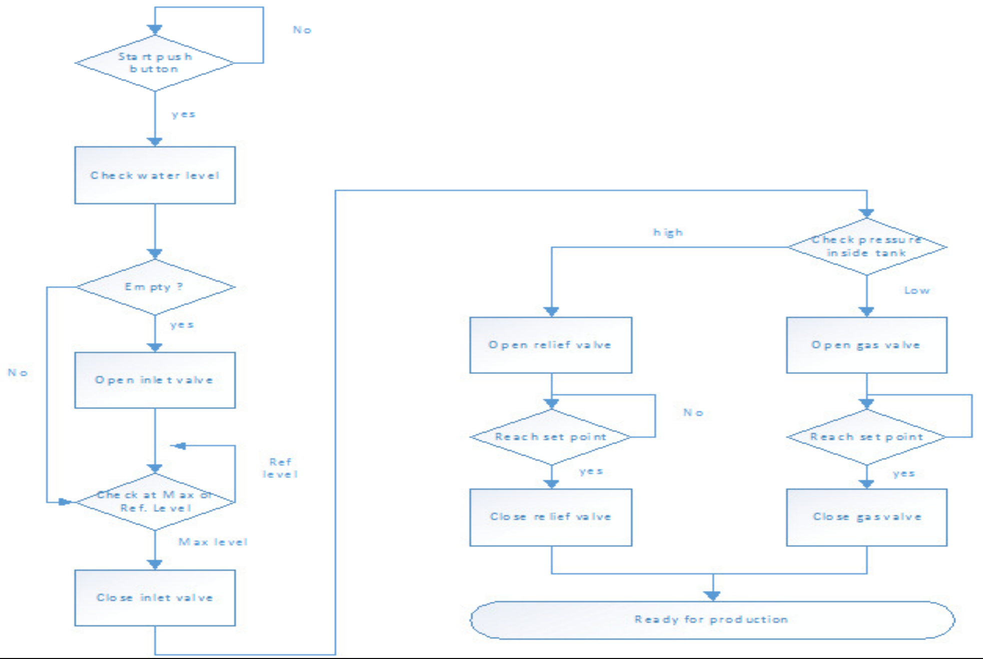
	Name	Data type	Offset	Start value	Retain	Accessible f...	Write...	Visible i
1	Static							
2	Arduino	Struct	0.0			✓	✓	✓
3	Configuration	TCON_IP_v4	0.0			✓	✓	✓
4	InterfaceId	HW_ANY	0.0	64		✓	✓	✓
5	ID	CONN_OUC	2.0	16#2		✓	✓	✓
6	ConnectionType	Byte	4.0	16#0B		✓	✓	✓
7	ActiveEstablish...	Bool	5.0	True		✓	✓	✓
8	RemoteAddress	IP_V4	6.0			✓	✓	✓
9	ADDR	Array[1..4] of Byte	6.0			✓	✓	✓
10	ADDR[1]	Byte	6.0	192		✓	✓	✓
11	ADDR[2]	Byte	7.0	168		✓	✓	✓
12	ADDR[3]	Byte	8.0	1		✓	✓	✓
13	ADDR[4]	Byte	9.0	145		✓	✓	✓
14	RemotePort	UInt	10.0	502		✓	✓	✓
15	LocalPort	UInt	12.0	0		✓	✓	✓

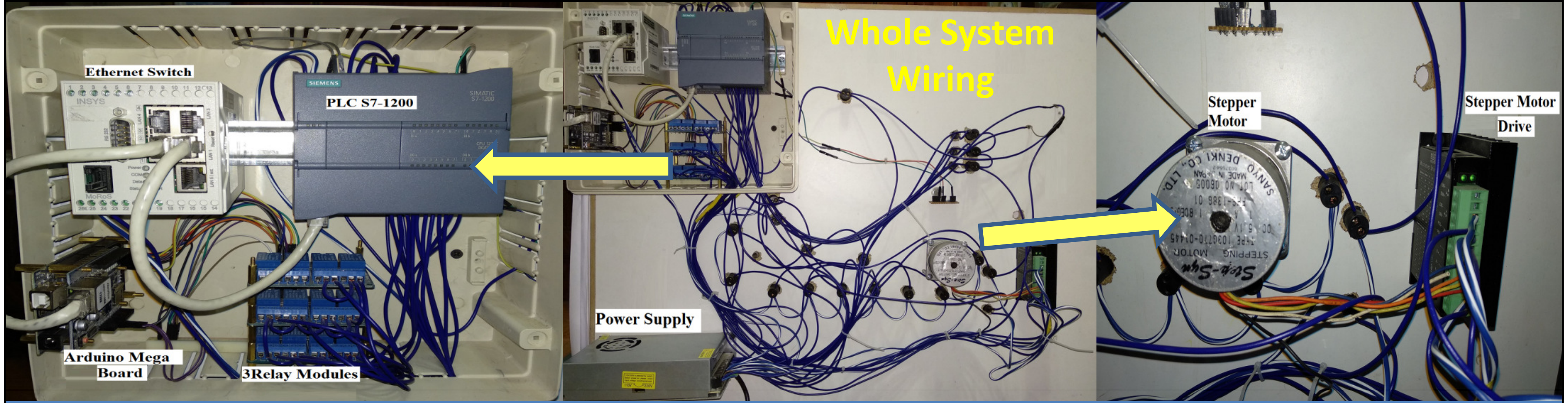
Filling process flow diagram:

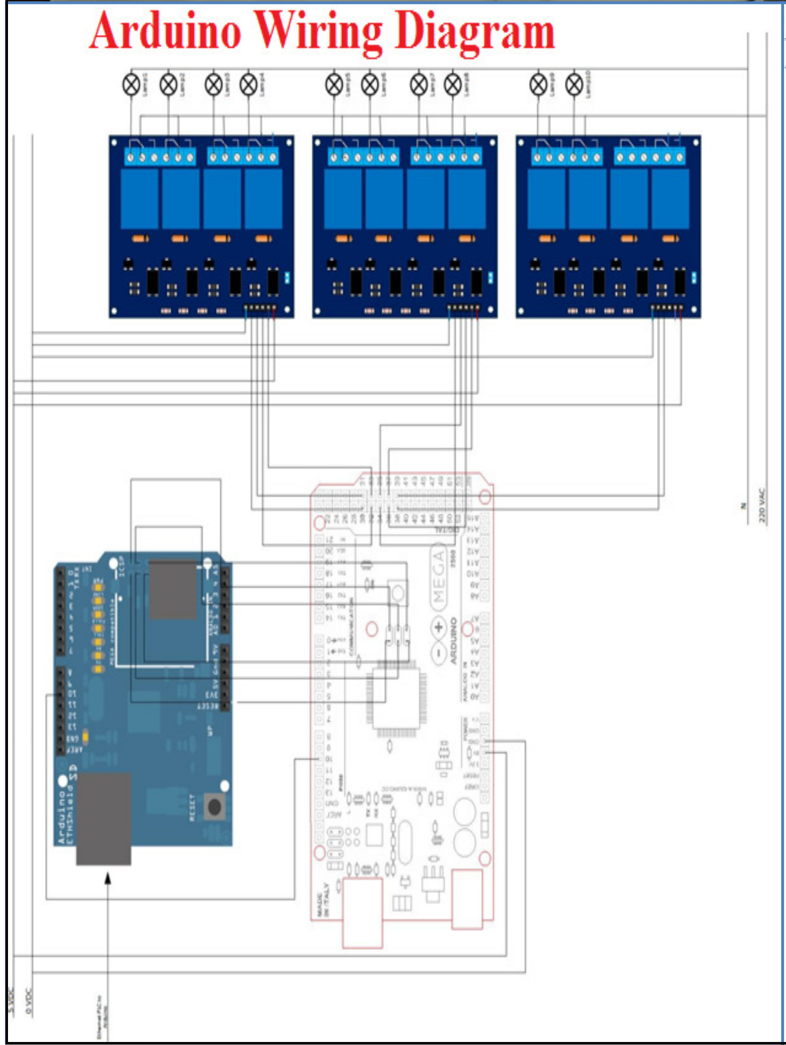


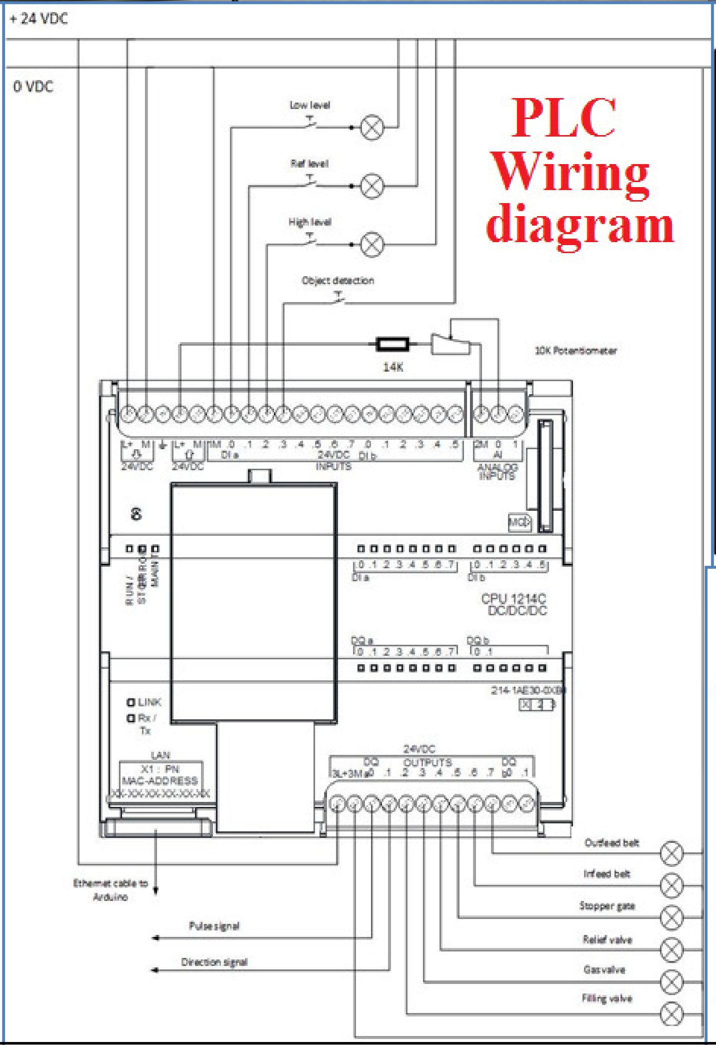
Start
Start push button
NO
Yes
Stepper motor rotates the main belt & infelt belt & out feed belt
Press object infeed switch
NO
Yes
Vials begin to enter transport belt (begin to register a bit to the shift register per each step of the belt)
Check tank ready for production
NO
Yes
No filling or Stoppering
Check vial at outfeed
NO
Yes
Add to bad object counter
Begin filling when vial reach position
Check vial at stopper station
NO
Yes
Insert stopper when vial reach position
Check vial at outfeed
NO
Yes
Add to good object counter
End

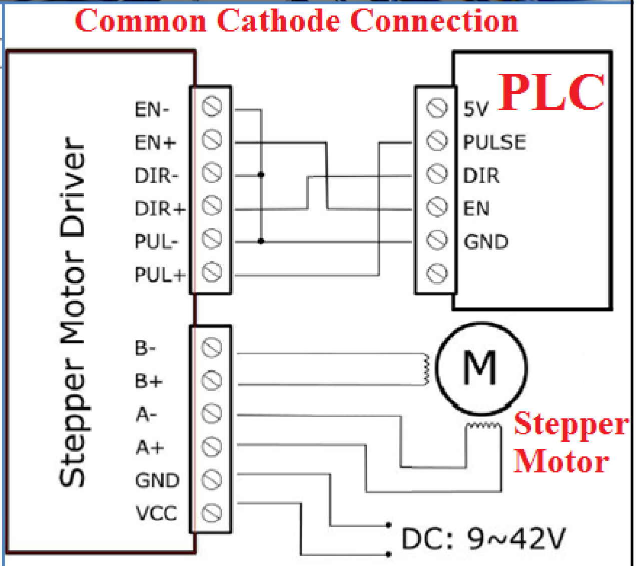
Tank level process flow diagram:







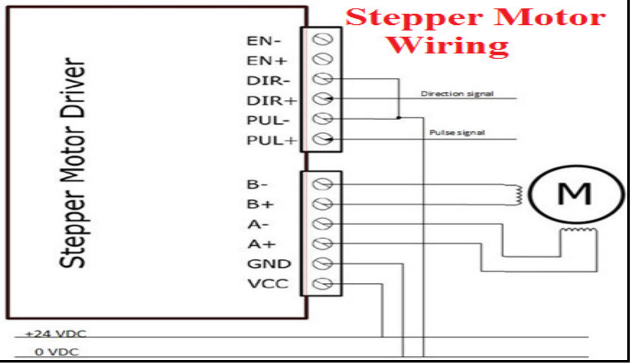




DIP Switch Setting

Micro Step	Pulse/Rev	S1	S2	S3
4	800	ON	OFF	OFF

Current (A)	S4	S5	S6
2.0	ON	OFF	OFF


Stepper Motor Wiring
Stepper Motor Driver
EN-
EN+
DIR-
DIR+
PUL-
PUL+
Direction signal
Pulse signal
B-
B+
A-
A+
GND
VCC
M
+24 VDC
0 VDC