Cairo University
Faculty of Engineering
Credit Hour System Programs

Graduation Project Report
Spring 2021

BATCH INDUSTRIAL CONTROL SYSTEM

Submitted by:

Mohamed Khaled Abdo Hussein ID: 1162266
Ali Mahmoud Ali Mahmoud Ahmed ID: 1153105
Mahmoud Mohamed Mohamed Abd-elmotte ID: 11617113
Mahmoud Emad El-din El-sayed Elerian ID: 1162121

Supervisor

Prof. Dr. Magdy Aboelela

Electrical Energy
Engineering Program

Acknowledgments

A strong appreciation is passed to our faculty, faculty of engineering Cairo University for its role in building our personalities, and to the department of Electrical Engineering and all of our professors for our academic knowledge.

Our humble gratitude and deep regards are expressed towards everyone who supported us during this journey, to our families and friends, whom we dedicate this document to. A profound sense of gratitude goes to our supervisor **Prof. Dr. Magdy Aboelela** for his guidance, mentorship and motivation. This project would not have succeeded without his guidance and supervision.

Last but not least, vast gratitude to Almighty God for enlightening our minds and continuous protection to make this project see the light.

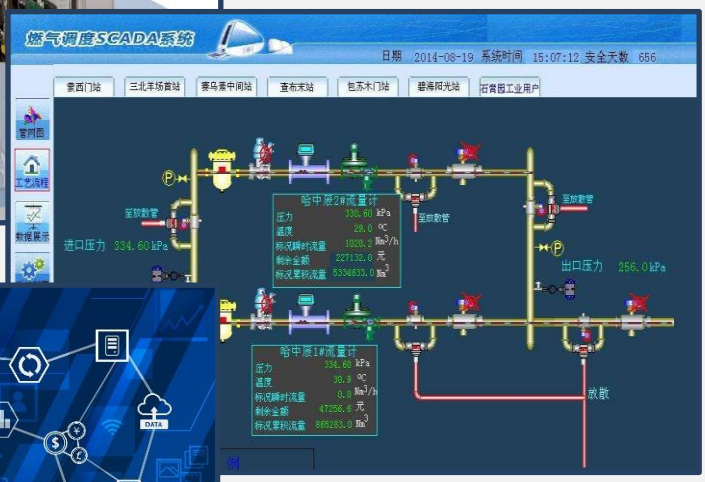
Abstract

The project is to mimic the batch process which is tank filling, heating, mixing and emptying. In our project we are using different technologies which are connected together to reach the best possible result. These technologies are PLC which we are using to control the process and is considered to be the brain of the whole process ”controller”, Sensors which used to give accurate data to the PLC to take appropriate action, SCADA which monitors the whole process and gives indication to the status of every device in our process and finally we are integrating the IOT technology to our project by using ARDUINO and an android application that control and monitor the whole process as an alternative of PLC and SCADA and the two solutions working together smoothly to give the supervisor more options to control the process.

Table of Contents

PART 1

Background



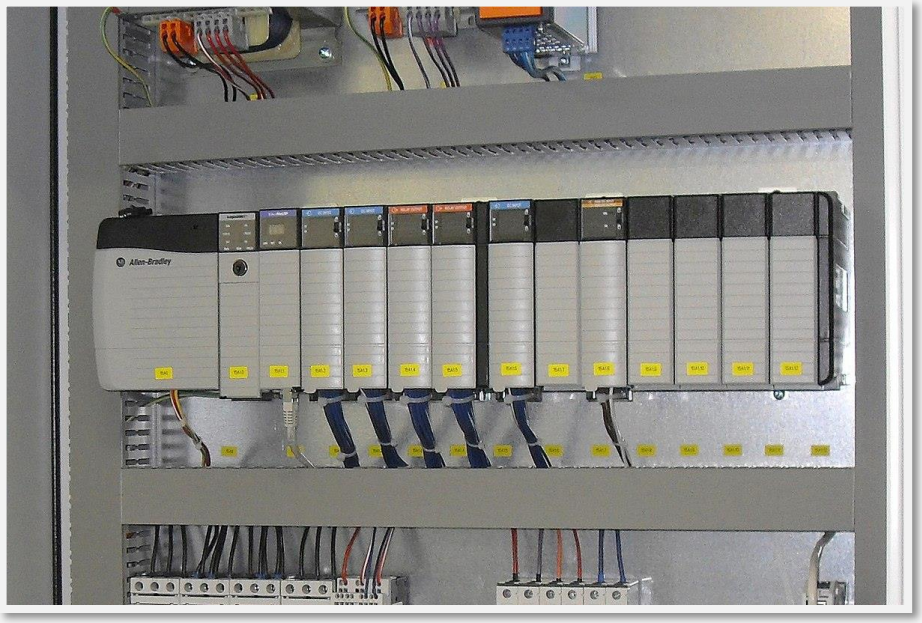
Chapter 1

Programmable logic Controller

Overview

In this chapter we will discuss:

- **History of PLC**
- **The hardware components of PLC**
- **The internal architecture of PLC**
- **Types of PLC**
- **PLC applications**
- **PLC programming techniques**



What is the definition of "PLC"?

A Programmable Logic Controller, or PLC, is a ruggedized computer used for industrial automation. These controllers can automate a specific process, machine function, or even an entire production line

Chapter 1: Programmable logic controller

Introduction

PLC: Stands for Programmable Logic Controller.

- A device was invented to replace the necessary sequential relay circuits for machine control.
- an industrial micro-Processor based device capable of dealing with different process inputs and outputs to achieve:
    - Sequential Logic Control Functions
    - Automatic Control Functions (digital control)

The first PLC can be traced back to 1968 when Bedford Associates, a company in Bedford, MA, developed a device called a Modular Digital Controller for General Motors (GM). The MODICON, as it was known, was developed to help GM eliminate traditional relay-based machine control systems.

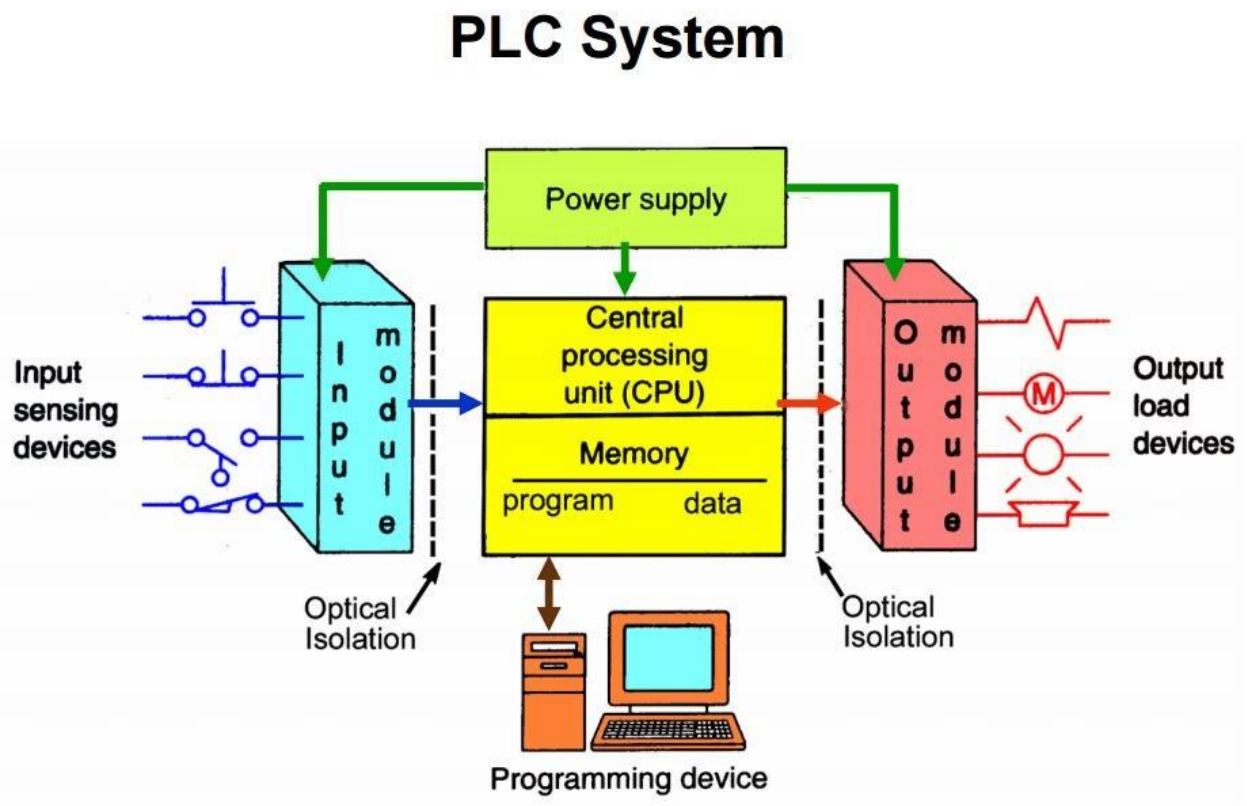
The advent of PLC began was to replace hard-wired controls. So, since PLC advent, it becomes to predominant choice for industrial controls. PLC is a small computer used for automation of real-world process such as control of machinery on factories assembly lines. where older automated systems would use hundreds or thousands of relays, a single PLC can be programmed as a replacement. So, since PLC advent, it becomes to predominant choice for industrial controls.

Before the days of the PLC the only way to control machinery was through the use of relays. Relays work by utilizing a coil that, when energized, creates a magnetic force to effectively pull a switch to the ON or OFF position. When the relay is de-energized, the switch releases and returns the device to its standard ON or OFF position.

A programmable logic controller (PLC) is a digital computer used for automation of typically industrial electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures. PLCs are used in many machines, in many industries. PLCs are designed for multiple arrangements of digital and analog inputs and outputs, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed-up or non-volatile memory. A PLC is an example of a "hard" real-time system since output results must be produced in response to input conditions within a limited time, otherwise unintended operation will result. The PLC is the brain of an automated system, while it’s inputs and outputs are the neurological system, connecting sensors and action-oriented parts to the brain.

PLC Hardware

Typically, a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface and the programming device.



PLC's consist of a few different components; **Input and output modules (I/O),** a **Power Supply**, **Programming Device** and a **Central Processing Unit (CPU).** The CPU also has memory for logic functions and data tables for I/O storage and will be covered later. So, let's start at the beginning and define exactly what inputs and output are and how they interact with a PLC.

PLC Input and output modules (I/O)

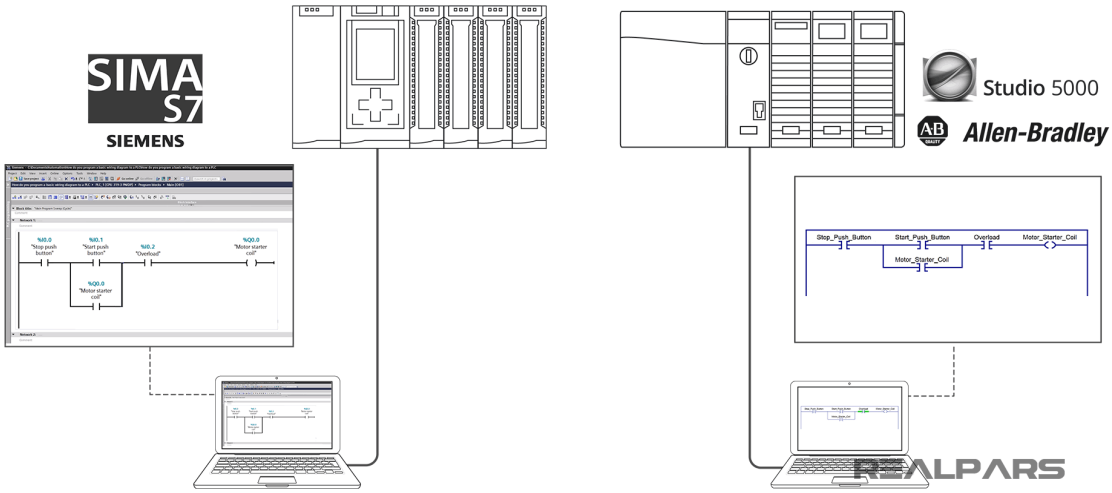
Inputs and Outputs can be digital or analog, depending on the field device needed. An input is a field device that updates the PLC on how the process is running. An input field device could be a pushbutton switch, pressure sensor, or photo sensor to name a few. There are hundreds of types of inputs that monitor a process and report the data back to the PLC. The input field device tied to the PLC is not controlled; only read by the PLC to update the status of the process input. These

inputs can be of different signal types, such as 4-20 milliamps, 24 volts-dc, or 110 volts-ac. A PLC is versatile enough to accept multiple signal types by having different input cards installed that accept these signals.
An output is a command supplied by the CPU Logic to control an output field device. These field devices may include, relays, timers, motor starters, lights, counters or displays. The output command is driven by the updated input tables, the CPU logic, and a signal sent to the output field device to be controlled.
These inputs and outputs are not directly connected to the PLC; therefore, if one failed or shorted electrically, it may short-circuit the entire PLC. A protection system was put into place to prevent this from happening with both inputs and outputs. These field devices are connected to an input/output card which has opto-isolators installed.
The opto-isolator transfers the input signal through circuitry to a signal the PLC can understand. Likewise, the output card works just the opposite of an input card. It converts the PLC language into a signal that is useable for an output field device. This is usually accomplished by an LED (light emitting diode) giving off light when an input signal is active. The light is then detected by a photo diode and converted into a useable signal for the PLC. Input tables are updated continuously and outputs can be changed in real time as the program is scanned for changes from the input field devices and processed to an output.

Programming Device

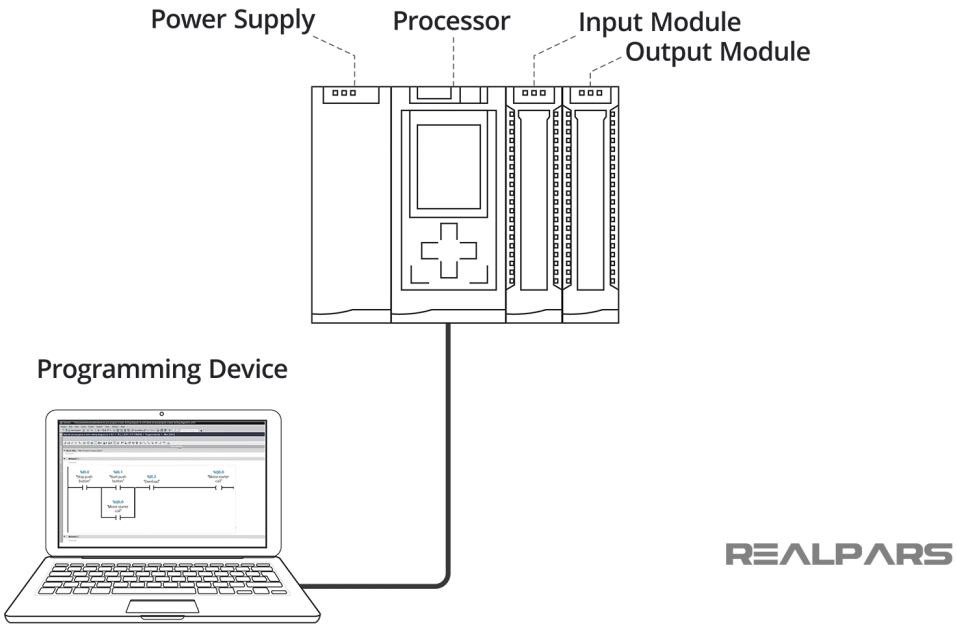
The Programming Device in today's industrial applications is usually a laptop or a desktop computer that facilitates the creation of decision-making programs destined for the PLC.

Examples of the programming software residing on the laptop are Studio 5000 for Allen Bradley PLCs or SIMATIC Step 7 for Siemens PLC's.



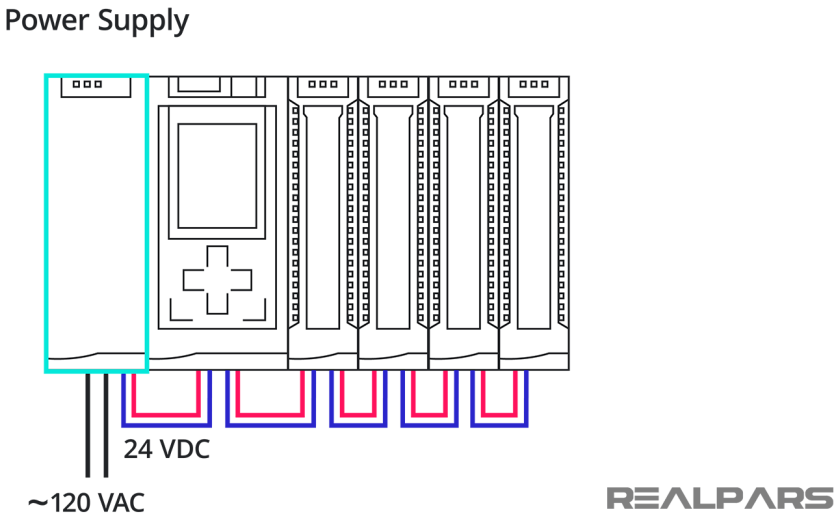
Power Supply

The Power Supply is connected to AC mains for the supply voltage.



The output of the Power supply is a DC voltage used to power all of the other modules associated with the PLC.

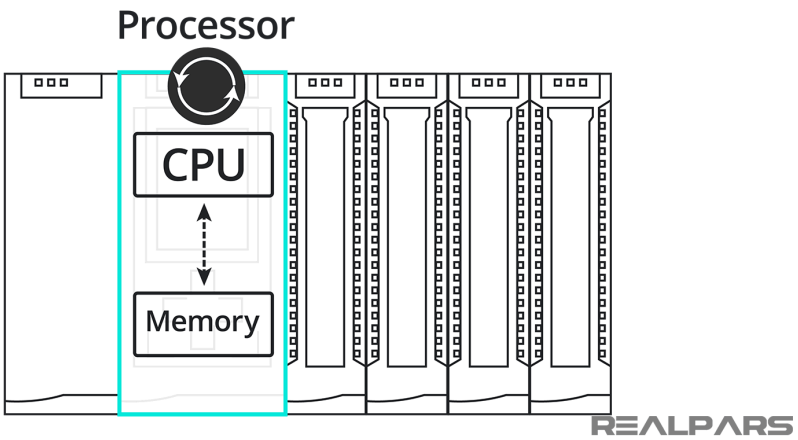
The Power supply DOES NOT provide power for field devices.



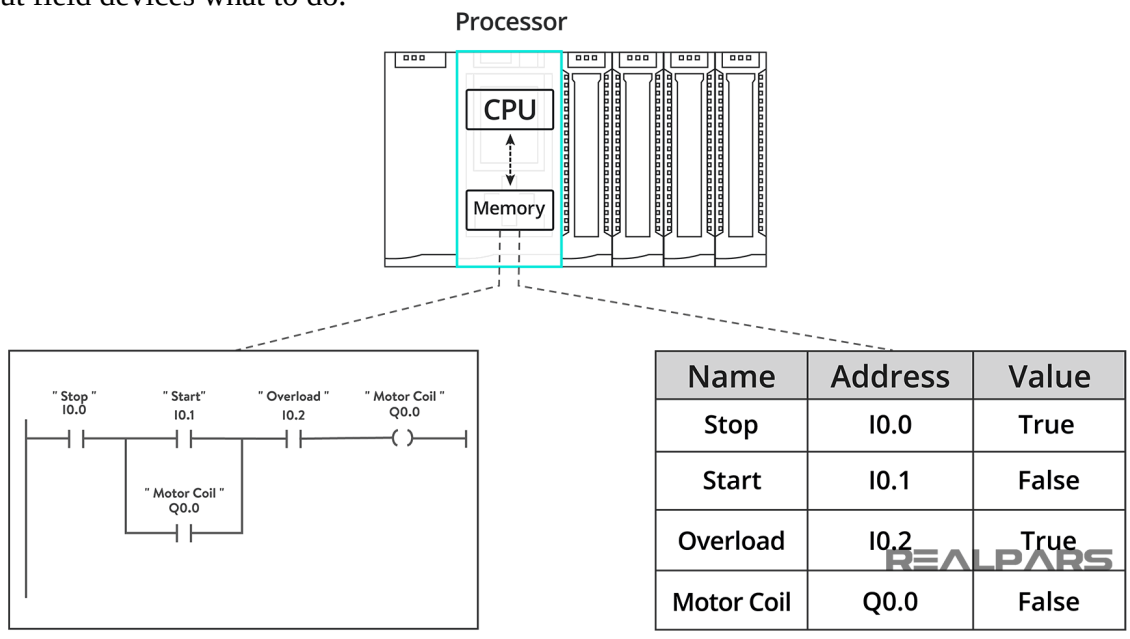
Processor

The Processor consists of the CPU (central processing unit) and memory.

The processor section makes decisions needed to observe and operate the field devices connected to the Input/Output modules.



The decisions are based upon a user-created program saved in the memory. The memory also stores data representing the condition of all input field devices and contains the data telling the output field devices what to do.



Name	Address	Value
Stop	I0.0	True
Start	I0.1	False
Overload	I0.2	True
Motor Coil	Q0.0	False

Communication Interface Module

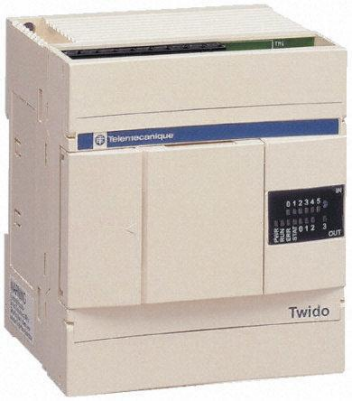
To transfer information between CPU and communication networks, intelligent I/O modules are used. These communication modules help to connect with other PLCs and computers which are placed at a remote location.

Types of PLCs

The two main types of PLC are fixed / compact PLC and modular PLC.

Compact PLC

Within a single case, there would be many modules. It has a fixed number of I/O modules and external I/O cards. So, it does not have the capability to expand the modules. Every input and output would be decided by the manufacturer.



Modular PLC

This type of PLC permits multiple expansion through “modules”, hence referred to as Modular PLC. I/O components can be increased. It is easier to use because each component is independent of each other.

- PLC are divided into three types based on output namely Relay output, Transistor output, and Triac Output PLC. The relay output type is best suited for both AC and DC output devices. Transistor output type PLC uses switching operations and used inside microprocessors.

- According to the physical size, a PLC is divided into Mini, Micro, and Nano PLC.

Some of the manufacturers of PLCs include:

- Allen Bradley
- ABB
- Siemens
- Mitsubishi PLC
- Hitachi PLC
- Delta PLC
- General Electric (GE) PLC
- Honeywell PLC

PLC Applications

PLCs have a variety of applications and uses, including:

1. Process Automation Plants (e.g. mining, oil &gas)
2. Glass Industry
3. Paper Industry
4. Cement Manufacturing
5. In boilers – Thermal Power Plants

PLC Programming

When using a PLC, it's important to design and implement concepts depending on your particular use case. To do this we first need to know more about the specifics of PLC programming.

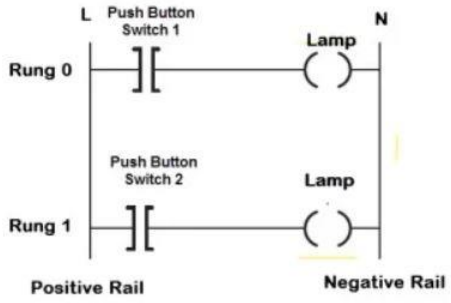
A PLC program consists of a set of instructions either in textual or graphical form, which represents the logic that governs the process the PLC is controlling. There are two main classifications of PLC programming languages, which are further divided into many sub-classified types.

1. **Textual Language**
   - Instruction list
   - Structured text
2. **Graphical Form**
   - Ladder Diagrams (LD) (i.e. Ladder Logic)

- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)\

Ladder Logic

Ladder logic is the simplest form of PLC programming. It is also known as “relay logic”. The relay contacts used in relay-controlled systems are represented using ladder logic.
The below figure shows a simple example of a ladder diagram.

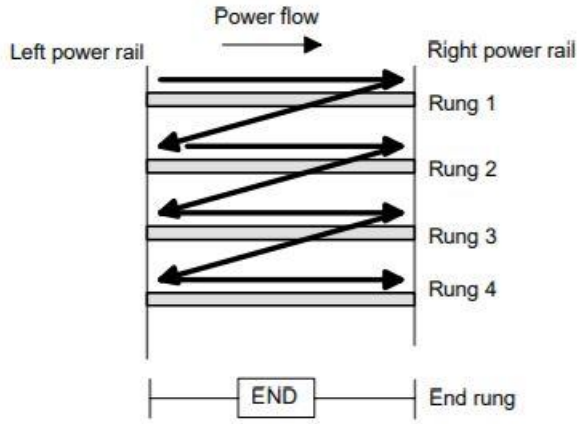


In the above-mentioned example, two pushbuttons are used to control the same lamp load. When any one of the switches is closed, the lamp will glow.

The two horizontal lines are called rungs and the two vertical lines are called rails. Every rung form the electrical connectivity between Positive rail (P) and Negative rail (N). This allows the current to flow between input and output devices.

In drawing a ladder diagram, certain conventions are adopted:

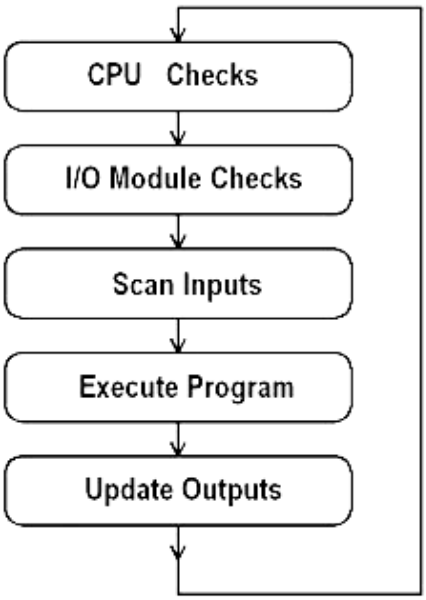
- The vertical lines of the diagram represent the power rails between which circuits are connected. The power flow is taken to be from the left-hand vertical across a rung.
- Each rung on the ladder defines one operation in the control process.
- A ladder diagram is read from left to right and from top to bottom, Figure showing the scanning motion employed by the PLC. The top rung is read from left to right. Then the second rung down is read from left to right and so on. When the PLC is in its run mode, it goes through the entire ladder program to the end, the end rung of the program being clearly denoted, and then promptly resumes at the start. This procedure of going through all the rungs of the program is termed a cycle. The end rung might be indicated by a block with the word END or RET for return, since the program promptly returns to its beginning.



- Each rung must start with an input or inputs and must end with at least one output. The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC. The term output is used for a device connected to the output of a PLC, e.g. a motor.
- Electrical devices are shown in their normal condition. Thus, a switch which is normally open until some object closes it, is shown as open on the ladder diagram. A switch that is normally closed is shown closed.
- A particular device can appear in more than one rung of a ladder. For example, we might have a relay which switches on one or more devices. The same letters and/or numbers are used to label the device in each situation.
- The inputs and outputs are all identified by their addresses, the notation used depends on the PLC manufacturer. This is the address of the input or output in the memory of the PLC

PLC Scan

Any changes in the status of input devices during the program or output scan are not recognized until the next input scan.



PLCs operate by continually scanning programs and repeat this process many times per second. When a PLC starts, it runs checks on the hardware and software for faults, also called a self-test. If there are no problems, then the PLC will start the scan cycle. The scan cycle consists of three steps: input scan, executing program(s), and output scan.

Input Scan: A simple way of looking at this is the PLC takes a snapshot of the inputs and solves the logic. The PLC looks at each input card to determine if it is ON or OFF and saves this information in a data table for use in the next step. This makes the process faster and avoids cases where an input changes from the start to the end of the program.

Execute Program (or Logic Execution): The PLC executes a program one instruction at a time using only the memory copy of the inputs the ladder logic program. For example, the program has the first input as ON. Since the PLC knows which inputs are ON/OFF from the previous step, it will be able to decide whether the first output should be turned ON.

Output Scan: When the ladder scan completes, the outputs are updated using the temporary values in memory. The PLC updates the status of the outputs based on which inputs were ON during the first step and the results of executing a program during the second step. The PLC now restarts the process by starting a self-check for faults.

Furthermore, data changes in the output table are not transferred to the output terminals during the input and program scans

Chapter 2

SCADA ,I/O devices and communication protocols

Overview

In this chapter we will discuss:

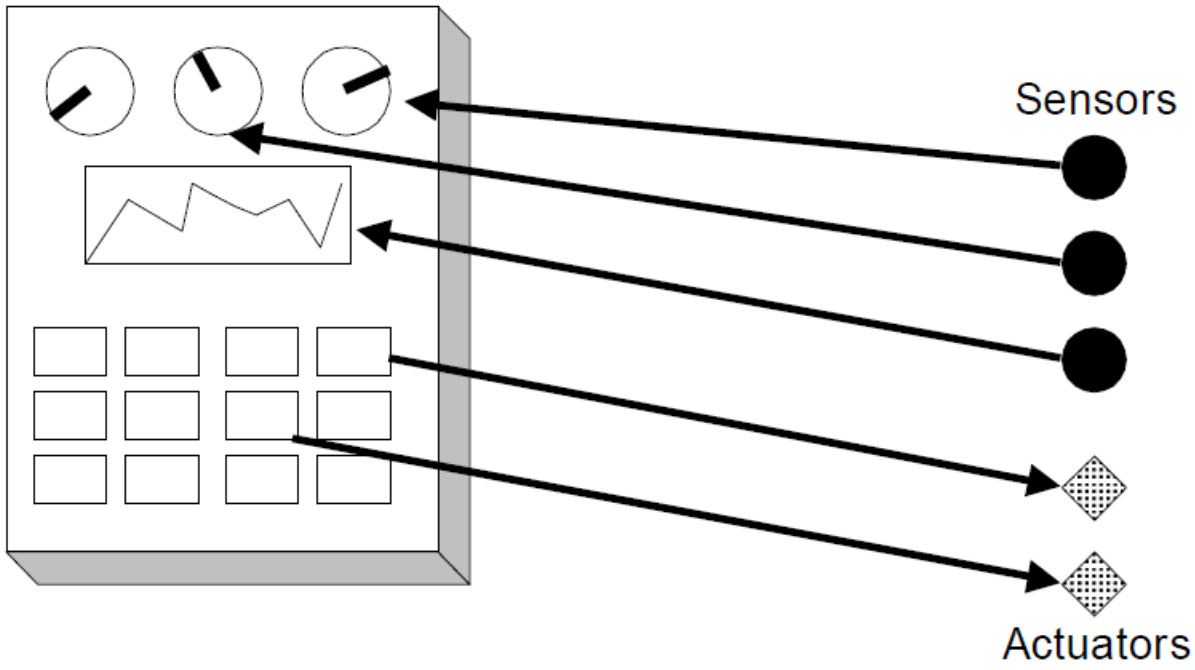
- **Introduction & Brief History of SCADA**
- **Fundamental principles of modern SCADA systems**
- **TAGs in SCADA software**
- **Signal Journey in SCADA Systems**
- **SCADA Applications**
- **Communication protocols**
- **I/O devices**

SCADA systems are crucial for industrial organizations since they help to maintain efficiency, process data for smarter decisions, and communicate system issues to help mitigate downtime.

Chapter 2 : SCADA ,I/O devices and communication protocols

Introduction & Brief History of SCADA

This chapter is designed to provide a thorough understanding of the fundamental concepts and the practical issues of SCADA systems. Particular emphasis has been placed on the practical aspects of SCADA systems with a view to the future. Formulae and details that can be found in specialized manufacturer manuals have been purposely omitted in favor of concepts and definitions. This chapter provides an introduction to the fundamental principles and terminology used in the field of SCADA.
It is a summary of the main subjects to be covered throughout the manual. SCADA (supervisory control and data acquisition) has been around as long as there have been control systems. The first 'SCADA' systems utilized data acquisition by means of panels of meters, lights and strip chart recorders. The operator manually operating various control knobs exercised supervisory control. These devices were and still are used to do supervisory control and data acquisition on plants, factories and power generating facilities. The following figure shows a sensor to panel system.



The sensor to panel type of SCADA system has the following advantages:

- It is simple, no CPUs, RAM, ROM or software programming needed
- The sensors are connected directly to the meters, switches and lights on the panel
- It could be (in most circumstances) easy and cheap to add a simple device like a switch or indicator

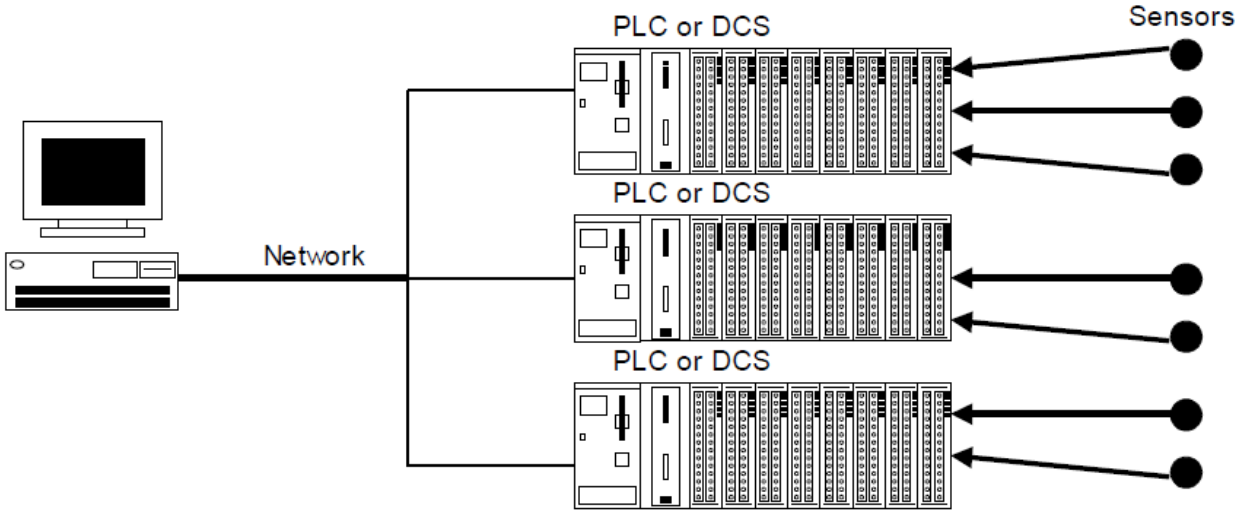
The disadvantages of a direct panel to sensor system are:

- The amount of wire becomes unmanageable after the installation of hundreds of sensors
- The quantity and type of data are minimal and rudimentary.
- Installation of additional sensors becomes progressively harder as the system grows
- Re-configuration of the system becomes extremely difficult
- Storage of data is minimal and difficult to manage
- No off-site monitoring of data or alarms
- Someone has to watch the dials and meters 24 hours a day

Fundamental principles of modern SCADA systems

In modern manufacturing and industrial processes, mining industries, public and private utilities, leisure and security industries telemetry is often needed to connect equipment and systems separated by large distances. This can range from a few meters to thousands of kilometers. Telemetry is used to send commands, programs and receives monitoring information from these remote locations. SCADA refers to the combination of telemetry and data acquisition. SCADA encompasses the collecting of the information, transferring it back to the central site, carrying out any necessary analysis and control and then displaying that information on a number of operator screens or displays.
The required control actions are then conveyed back to the process. In the early days of data acquisition, relay logic was used to control production and plant systems. With the advent of the CPU and other electronic devices, manufacturers incorporated digital electronics into relay logic equipment. The PLC or programmable logic controller is still one of the most widely used control systems in industry. As need to monitor and control more devices in the plant grew, the PLCs were distributed and the systems became more intelligent and smaller in size.

PLCs and DCS (distributed control systems) are used as shown below.



The advantages of the PLC / DCS SCADA system are:

- The computer can record and store a very large amount of data
- The data can be displayed in any way the user requires
- Thousands of sensors over a wide area can be connected to the system
- The operator can incorporate real data simulations into the system
- Many types of data can be collected from the RTUs
- The data can be viewed from anywhere, not just on site

The disadvantages are:

- The system is more complicated than the sensor to panel type
- Different operating skills are required, such as system analysts and programmer
- With thousands of sensors there is still a lot of wire to deal with
- The operator can see only as far as the PLC

Components of Modern Automation Systems

Although Modern automation systems vary in their philosophies and architectures, we can define four separate basic components that form the automation systems. These components are:

a) Field devices
b) Automatic Control System
c) SCADA software
d) Communication Networks

Field Devices

Field devices are the inputs and outputs to the control system; they are its eyes and hands. They can be divided into two sections, sensors and actuators, sensors do the eyes part of the job, they provide instantaneous information about the status of the field or process whether these pieces of. Typical examples for sensors are flow meters, pressure sensors and temperature sensors (these ones provide an analog signal proportional to the physical value of the measured variable and may be called “Transmitters”), other examples of sensors are limit switches, photoelectric sensors and thermostats (these ones provide discrete signals that represents predefined process states). On the other hand, actuators do the hands part of the job, they receive control orders from the control system and intend to affect the process according to these orders. Typical examples of actuators are control valves, contactors and VFDs (Variable frequency drives).

Automatic Control System

Typically, a PLC (Programmable logic controller) or a DCS (distributed control system), these devices provides an extremely reliable structure to perform huge control functions in a relatively very short response time.
PLCs and DCSs are made to be connected directly to field devices (sensors and actuators), and thanks to their ability to store control programs and algorithms, these devices can run plants and utilities in a standalone manner with minimum dependency on humans.

SCADA Software

SCADA stands for supervisory control and data acquisition, and as it is clear from its name, its function is to acquire filed data and to issue supervisory control commands to a PLC or a DCS. SCADA software provides an interface between plant operators and field devices through graphical screens that represent the actual field, these screens is continuously updated with the latest field equipment status. Key features of SCADA software are:

- User interface
- Graphics displays
- Alarms
- Trends
- RTU (and PLC) interface
- Scalability
- Security
- Access to data
- Database
- Networking
- Client/server distributed processing

SCADA software can be divided into two types, proprietary or open. Companies develop proprietary software to communicate to their hardware. These systems are sold as 'turnkey' solutions. The main problem with this system is the overwhelming reliance on the supplier of the system. Open software systems have gained popularity because of the interoperability they bring to the system.

Communication Networks

Here comes the telemetry part to the picture, communication networks are the means by which data is transferred from RTUs (PLC or DCS) at remote site to the SCADA software stations. The communication network may be as short as 10 meters or as long as thousands of kilometers. Many communication technologies have been implemented successfully in modern automation systems such as landlines (copper wires and fiber optics), microwave, radio and even satellite. These technologies vary in their architectures, speeds, and most important, prices.

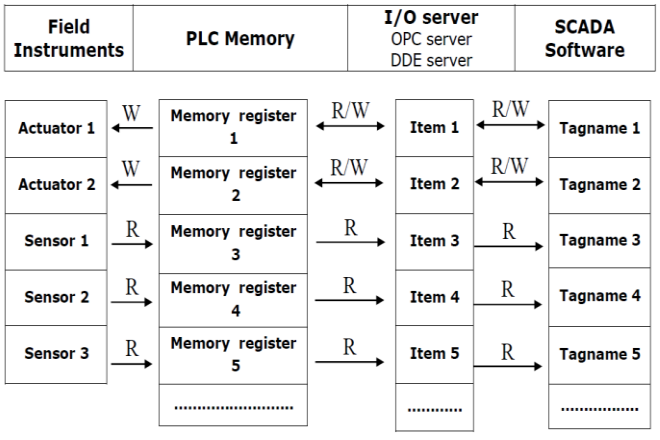
Signal Journey in SCADA Systems

This section provides the basic understanding of how the data is being moved through SCADA systems. We mean by the term "data" two things:

- The first one is the data being acquired using the sensors and transmitters from the field (Data acquisition).
- The second one is the data being instructed or ordered from the user through SCADA software (Supervisory control).

You should already know how the data being moved from measuring instruments to the PLC, and that is the first station of the signal journey. Following is a detailed description of the rest of the "signals journey" until it reaches its final station which is the SCADA software. It is clearly known that the PLC "holds" in its memory registers the data should be acquired by the SCADA software, so in this case, the SCADA software READs data from the PLC. And it's also clearly known that the SCADA software should be able to WRITE data to other PLC memory registers.

As shown in the figure, the data is transferred between the PLC and the SCADA software; you will notice that there is an intermediate step between the PLC and the SCADA software that is called "I/O server", and hence, the SCADA software will work as a "client" for that "server". The "I/O" server is software that "serves" the communication between the PLC and the SCADA software.



Typical Signal Journey

Let's take an example of a signal journey from the instruments to the SCADA software. Consider SCADA software that should display an oil tank level. Let's track the journey of this signal (value) from the beginning (i.e. transmitter) to the end (i.e. SCADA software)

The first step of the signal journey: (Field Instrument)

The PLC is continuously acquiring the tank's oil level through its analog channel and holds it in a pre-specified place. This place in PLCs is called a "memory register".

Second step of the signal journey: (I/O server

The I/O server will access this known "memory registers" and READs the value of the oil level and put it in a pre-specified place. This place in "I/O server" is called an "Item". For "items" in I/O servers, when do they are updated? Answer: the "item" is NOT continuously updated like "memory registers": in PLCs, but instead, it is only updated when the client asks for it (Remember the "client" is the SCADA software).

The third step of the signal journey: (SCADA software)

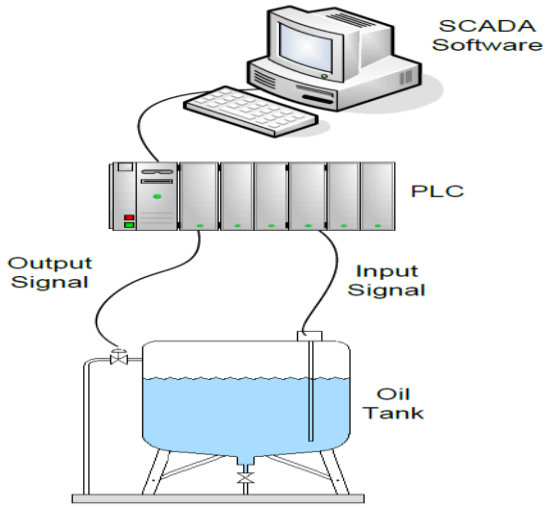
The SCADA software is connected to the "I/O server" and typically, will continuously access this Known "item" to READ the value of the oil level in the tank and put it in a pre-specified place. This place in SCADA software is called a "Tag". This "Tag" in turn is always updated with the oil level value.

The fourth (final) step of the signal journey: (manipulating data and animation)

Now, The SCADA software "knows" the value of the oil level in the tank, So It can be represented in graphics, stored in data base, trended, monitored for alarms and so on.

Reverse direction of a signal journey:

The same thing is applied for parameters entered by the user to the SCADA software (The reverse direction), for example a set point or an order of starting the production, The SCADA software holds the data entered by the user in a pre-specified "Tag" and WRITEs it to a pre-specified "Item" in the I/O server and then the I/O server Writes the this value or order to a pre-specified "Memory register", So the PLC now "Knows" the value of the required set point and will use it (i.e. output this value to a controller)



Basic Input Devices and Sensors types

The Input devices in PLC (Programmable Logic Controller refers to the required components to take the data or pass the data from physical world (i.e. field data) to the PLC. Input devices can be Transducer, Sensors, Switches, Push Button which is connected with PLC. This are the devices where PLC hardware talking with the physical actions in the form of data or information

to execute certain action against it. This data or information can be in the form of Digital signal or Analog Signal. The PLC uses this input information to make decisions based upon its program whether to energize and de-energize the outputs controlled by the PLC.

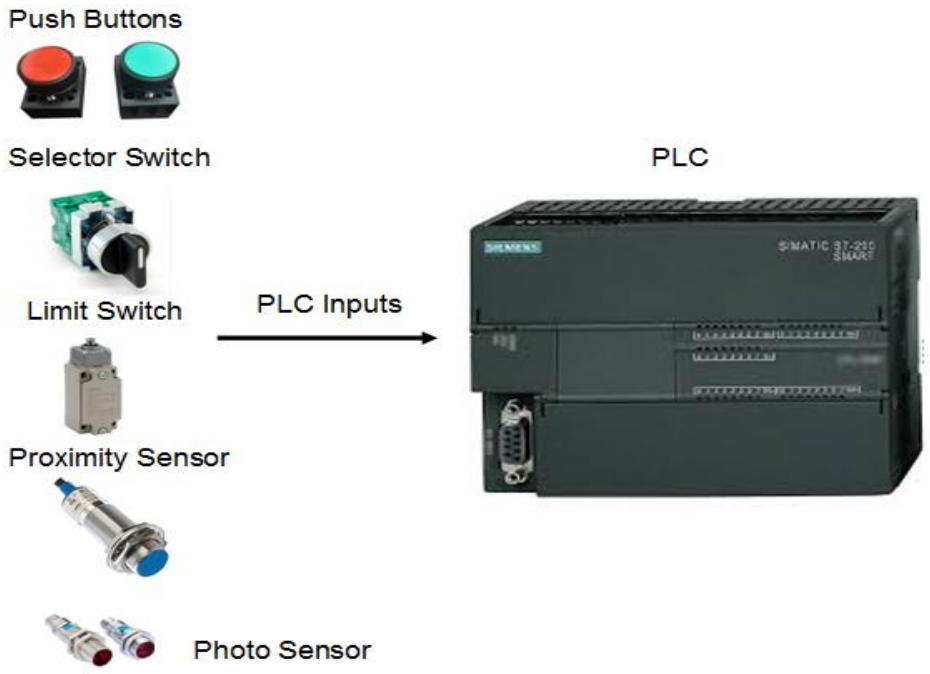
Digital Inputs (DI) are referred in simple two state 0 / 1 or High state /Low Stare or ON/OFF. This signal can be achieved through simple on/off or Digital type input sensors.
Analog Inputs are referred in the continuous electrical signal in the particular range like 0 t0 5V or 0 to 20ma from Transducer, Sensors which convert a physical quantity into a signal.

As PLC applications require that you know the basics requirements of automation itself before you can even start Programming your PLC. Automation is a technology by which a set of procedures is performed without any human intervention and Inputs devices performs major roles in executing complete Automation process. Let's discuss in brief about inputs devices and their functional behavior. It will help to understand basic requirement at the input side of PLC before your start design control circuit and programming.

Digital Input Devices

Push buttons

Push buttons are the devices which used make or break the contacts with mechanical arrangements. Based on the making and breaking contact push button categorized in two type. Normally open contact and Normally Closed Contact.



- Normally Open (NO) Contact push button

When we connect Normally Open Contact type push button the circuit becomes Open Contact Circuit and signal to the PLC will be always LOW. With the use of Normally Open Contact High level signal will pass to the PLC when user press the button. By mechanical arrangements with Normally open contact push button will momentary. Signal will break as soon user release the button.

- Normally Closed (NC) Contact push button

By connecting Normally Closed type push buttons circuit becomes always short circuited with PLC. The signal level with Normally closed push button is always. The contact will be break as the user pressed the push button. This means pushing the button will make its metal contacts touch with each other, closing the connection between the two connected terminals.
Push buttons will have arrangements to contain or connect both type contact normally open and normally closed. User can achieve normally open and normally closed signal from single push button operation.

Selector Switch:

Selector switches are still manually operated switches, however instead of being normally open or closed, there are more than two contacts to select from. The usual example is found in electric fans, where you can select a number that then dictates the speed of the fan's motor. This process actually selects a varying load for the motor in order to control its speed.

Limit Switches:

PLC Basics Limit Switch Limit switches, as the name implies, change state when a predetermined limit is reached. These are actually useful in automation because you can set a limit (using the limit switch) where a specific process stops. There are also different types of limit switches which allow us to choose the physical quantity to limit in our control system design.

Level Limit Switch:

PLC Basics Level Limit Switch Level switches—more commonly called Level Sensors, are used to control the height of a liquid inside a container, usually a tank. They are most commonly used in conjunction with inlet and outlet valves in a liquid level control system, or in a heating and mixing application.

Sensors:

In PLC automation sensor are devices which playing important roles for input devices. Sensors are the devices which measures or detect the physical quantity and covert into electrical signal. This electrical signal either in Analog or Digital format. PLC will receive this signal from sensors and performed the respective action based on program

uploaded in PLC to response it. Choosing sensor types is depends on many factors like, Input Supply Voltage, output Voltage, Operating Range, Type of output (PNP / NPN type), Accuracy, Sensitivity etc.

There are two types of sensors used in PLC Automation: Analog and Digital:

- **Analog sensors**: -

Analog sensors are devices that output a continuous voltage linear to the experienced change in the environment. They are most extensively used in Temperature Sensing, Distance Sensing, Luminance Sensing, Pressure Sensing, and basically in PLC applications where an exact, certain unit of measure is involved. In PLC automation, the typically used analog inputs vary from 0-20mA, 4-20mA, or 0-10V. Hence, the sensing may also become current sensing or voltage sensing.

- **Digital sensors**: -

Digital sensors are the device which gives outputs in a status HIGH or in a status LOW voltage only. These devices working on principle of Binary number system where output will be measured in 0 and 1 which can be represent LOW and HIGH, respectively. The High Level and Low Level will be defined based on the output voltage at sensor means for 24VDC supply voltage sensor high level output will 24VDC and Low-level output should be 0V. These digital sensors have internal switching circuits that classify them as either a sinking (NPN) or a sourcing device (PNP). Basically, PNP provides +24 V as input, and NPN provides -24V as input. Hence, the input modules must also be classified as sourcing or sinking.

PLC Actuators and Output Devices

Now that we know PLC Basics and some PLC input devices, we now go into the interface between the controller and the real world: output devices.
In PLC programming, they are the devices that are activated using conditional statements that the input devices form.
Output devices are operated by the PLC using the DC voltage at the output. A number of devices can be controlled by the PLC, but we can categorize them to **indicators/alarms** and **actuators**.

Indicators and Alarms

The indicators/alarms include devices that can represent a predetermined state in a basic PLC control system.
Indicators such as pilot lights—usually a green one, may represent that the PLC system is running properly. A red pilot light usually represents a PLC system that does not have ongoing operation.
Alarms provide notification with higher urgency. A blinking red light, a horn, or a buzzer may represent that a hazardous condition has been met in the system.
These are just the most common applications of indicators and alarms.

Actuators

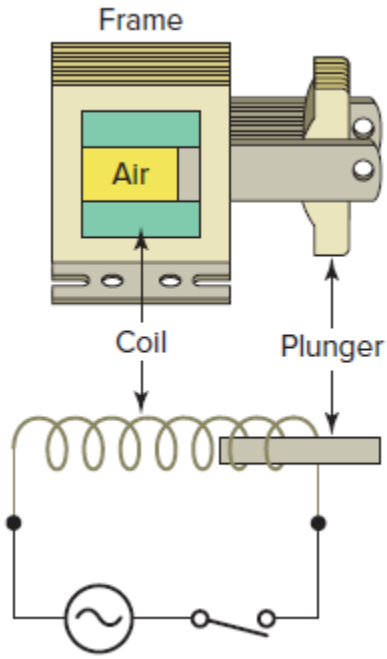
Actuators, on the other hand, convert an electrical signal (control signal) to mechanical movement.

Simply put, actuators **translate** the output signal into practical operations. They are usually connected to high-power sources, but they do have coils to allow DC voltage to control the switching.

Solenoids

In some cases, electrical devices—even the most powerful ones, are just not fit to perform a specific task over and over again. For applications like frequent heavy pressing, molding or lifting, hydraulic or pneumatic systems are much more suited for the task.

But ever since our logic controllers made their way to our industry, they make the use of hydraulic and pneumatic systems MUCH EASIER than before. The combination of these systems made a cost-effective solution for large-scale projects and/or assembly lines.
The solenoid consists of a fixed part: the coil and the frame, and a dynamic part: the plunger/armature as shown in the photo below.



When the DC voltage of the PLC closes the switch of the solenoid (which connects it to the other power source), the current passes through the coil—which creates a strong magnetic field. The magnetic field pulls the plunger into the frame.

When the electrical current is removed from the solenoid circuit, the plunger goes back to its original position (usually they have springs).

Usually, solenoids are used to control the flow directions of fluids in pneumatic and hydraulic systems.

Motors

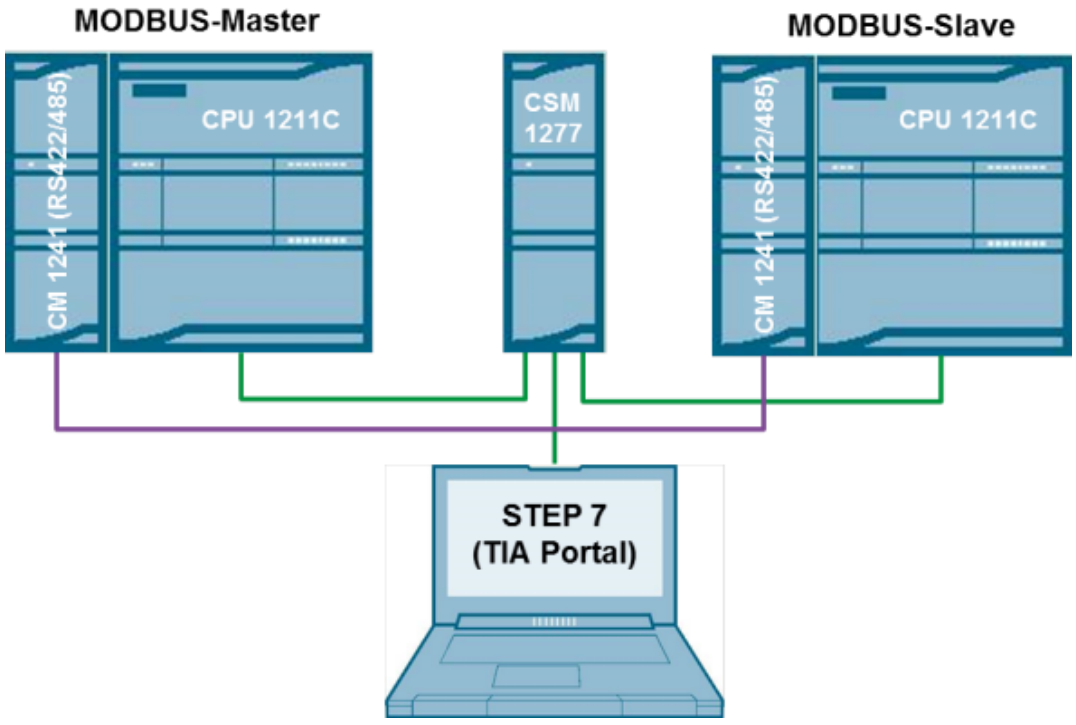
Motors are devices that convert electrical energy into mechanical energy using magnetic fields. They consist of a multitude of coils mounted on their frames, but the way they are connected varies depending on the type of motor.

DC motors

The DC motor's construction has two major components: the armature windings and the field windings.

When a voltage is applied to the field windings, a strong magnetic field is created in the DC motor. When the armature windings encounter the magnetic field, they experience a magnetic force directly proportional to the strength of the magnetic field (hence directly proportional to the applied power to the field windings). This makes the shaft rotate at a speed directly proportional to the DC voltage

communication protocols and Its Types!



The Communication protocols is the way to trans-receive the data with a set of rules that sends or receives between two or more devices. The communication protocol is the media or channel between two or more communicating devices. By using the communication protocols, two devices can connect and communicate with each other. Without communication protocol, devices can only be connected but not communicated.

The different types of available communication protocol help to expand the device network by connecting and communicating with each other. There are many old communication protocols still in use and widely getting used to expand network. These old, communication methods still have their place, as they allow quick and easy connections among a single supplier's products. Many PLC suppliers still support these older physical, wired connections—most often based on RS-232C, RS-422 and RS-485 connectivity—and the related proprietary communication protocols.

The communication protocols not only being used to expand the PLC network and Also its used to expand number of IO devices by connecting additional modules (Expansion modules).

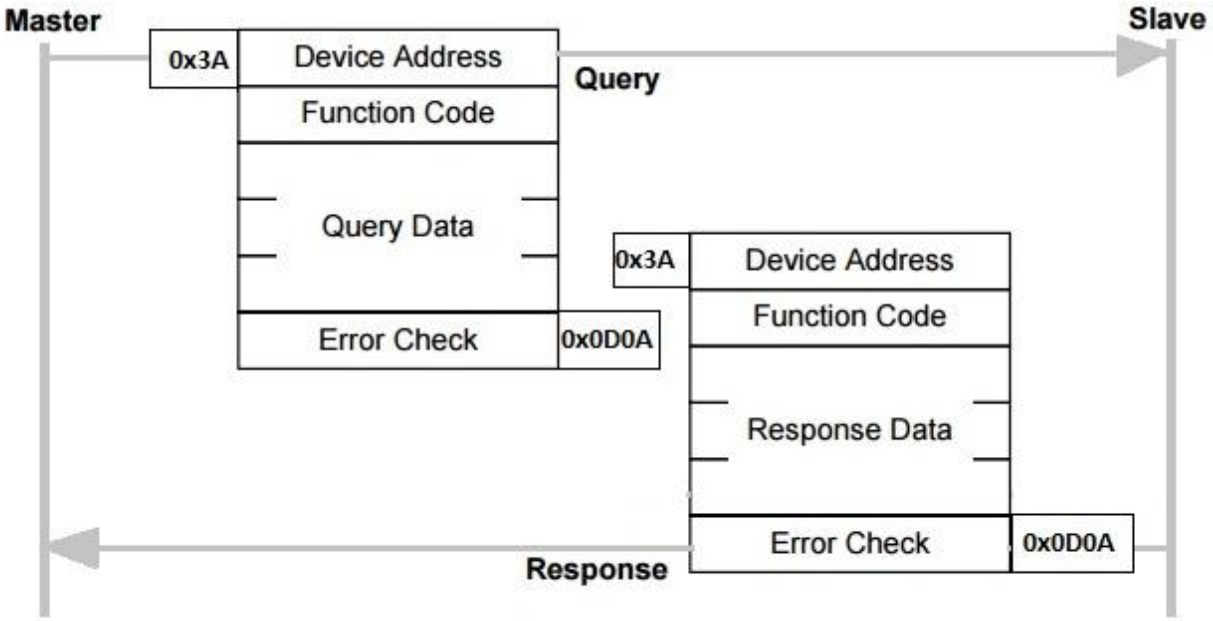
PLC networks provide you with a variety of networking options to meet specific control and communications requirements. Typical options include remote I/O, peer-to-peer, and host computer communications, as well as LANs. These networks can provide reliable and cost-effective communications between as few as two or as many as several hundred PLCs, computers, and other intelligent devices.
Many PLC manufacturers like Siemens, Allen-Bradley, Mitsubishi offer proprietary networking systems that are unique and will not communicate with another make of PLC. This is because of the different communications protocols, command sequences, error-checking schemes, and communications media used by each manufacturer.

In this tutorial we are going see different Standard PLC communication protocols and Its Types use in PLC communication and expanding its network. This protocol used for PLC-to-PLC communication, over a point-to-point or multipoint network using standard PLC instructions, PLC to other device like HMI, Variable Frequency Drives (VFD), GSM / GPRS modems, IOT devices and so on.
Most commonly available protocols with PLC in all industrial PLC are:

- Modbus RTU
- Ethernet/IP
- Ethernet TCP/IP
- Modbus TCP/IP
- Profinet

Modbus RTU

Modbus RTU is an open serial protocol derived from the Master/Slave architecture originally developed by Modicon (now Schneider Electric). It is a widely accepted serial level protocol due to its ease of use and reliability. Modbus RTU is widely used within Industrial Automation Systems (IAS), Home Automation, Building Management, Robotics etc.
Modbus RTU messages are a simple 16-bit structure with a CRC (Cyclic-Redundant Checksum). The simplicity of these messages is to ensure reliability. Due to this simplicity, the basic 16-bit Modbus RTU register structure can be used to pack in floating point, tables, ASCII text, queues, and other unrelated data.



Modbus Message Frames

This protocol primarily uses an RS-232 or RS-485 serial interfaces for communications and is supported by almost every commercial SCADA, HMI, OPC Server and data acquisition software program in the marketplace. This makes it very easy to integrate Modbus compatible equipment into new or existing monitoring and control applications.

The communication protocols are depended upon list of fundamental parts as mentioned below.

- Baud rate
- Start/Stop bits
- Parity bit
- Network length
- Number of nodes

Baud Rate is the number of bits per second that are being transmitted or received. Common values (speeds) are 1200, 2400, 4800, 9600, 19200, and 38400.

Start bit: This is a synchronizing bit added just before each character we are sending. This is considered a SPACE or negative voltage or a 0.

Stop bit: This bit tells us that the last character was just sent. This is considered a MARK or positive voltage or a 1.
Parity bit is added to check whether corruption has occurred. Common forms of parity are: None, Even, and Odd. During transmission, the sender calculates the parity bit and sends it. The receiver calculates parity for the character and compares the result to the parity bit received. If the calculated and real parity bits don't match, an error occurred and we act appropriately.

RS-232 Communications

RS-232 is an asynchronous communications method. We use a binary system to transmit our data in the ASCII format. PLCs serial port is used for transmission/reception of the data, it works by sending/receiving a voltage, With RS232, normally, a 1 bit is represented by a voltage -12 V, and a 0 by a voltage +12 V. (The voltage between +/- 3 volts is considered There are 2 types of RS-232 devices.) DTE – Data Terminal Equipment and a common example is a computer.
DCE – Data Communications Equipment and a common example is a modem.
PLC may be either a DTE or DCE device. When plc and external device are both DTE, (or both DCE) devices they can't talk to each other. The solution is to use a null-modem connection. Usually, the plc is DTE and the external device is DCE. Using RS-232 with PLC Some manufacturers include RS-232 communication capability inbuild with the main processor and Some uses the "programming port" or External additional pluggable RS232 expansion Module.

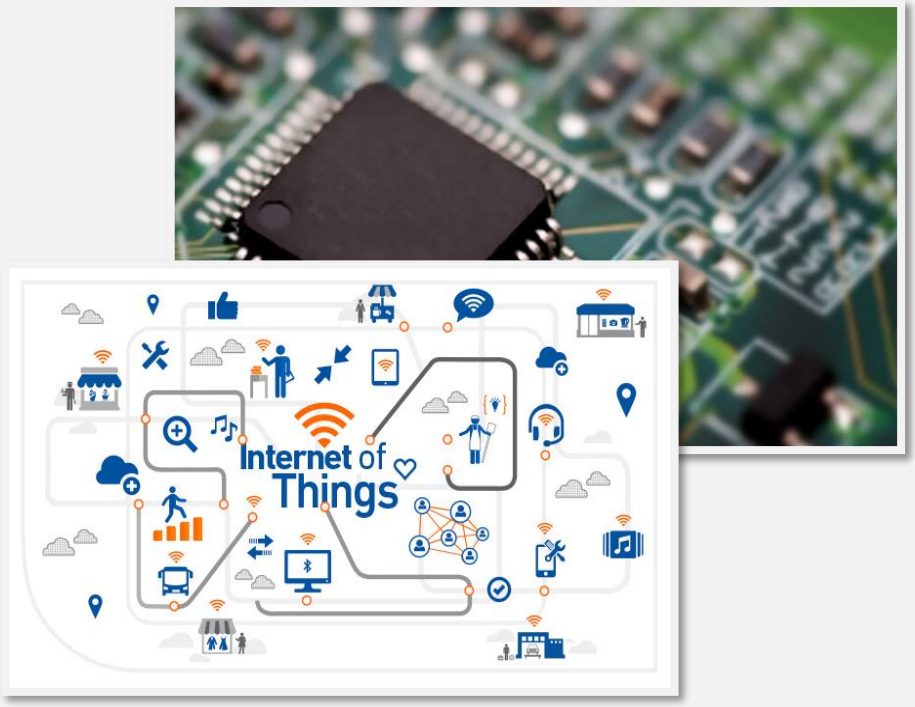
Chapter 3

Micro Controllers and IOT

Overview

In this chapter we will discuss:

- **Microcontrollers definition.**
- **Basics of Microcontrollers**
- **Basic Structure of a Microcontroller**
- **Advantages of Microcontrollers**
- **Disadvantages of Microcontrollers**
- **Definition and interpretation of IoT**

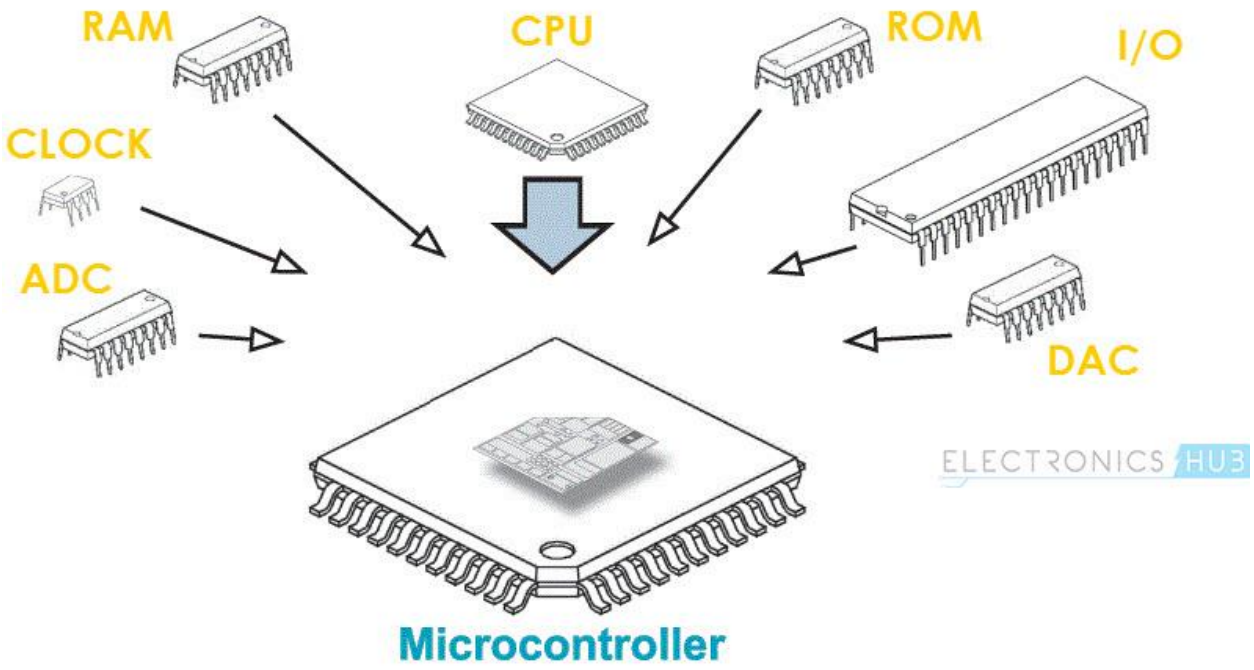


A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Chapter 3 : Microcontrollers and IOT

What is a Microcontroller?

A Microcontroller is a VLSI (Very Large-Scale Integration) Integrated Circuit (IC) that contains electronic computing unit and logic unit (combinedly known as CPU), Memory (Program Memory and Data Memory), I/O Ports (Input / Output Ports) and few other components integrated on a single chip.



Sometimes, a Microcontroller is also called as a Computer-on-a-Chip or a Single-Chip-Computer. Since the Microcontroller and its supporting circuitry are often embedded in the device it controls, a Microcontroller is also called as an Embedded Controller.

Microcontrollers are omnipresent. If a device or an application involves measuring, storing, calculating, controlling or displaying information, then device contains a Microcontroller in it. Let us see some of the areas where microcontrollers are used.

The biggest user of Microcontrollers is probably the Automobiles Industry. Almost every car that comes out of the assembly factory contains at least one Microcontroller for the purpose of engine control. You can find many more Microcontrollers for controlling additional systems.

Consumer Electronics is another area which is loaded with Microcontrollers. Microcontrollers are a part of Digital Cameras, Video Camcorders, CD and DVD Players, Washing Machines, Ovens, etc.

Microcontrollers are also used in test and measurement equipment like Multimeters, Oscilloscopes, Function Generators, etc. You can also find microcontrollers near your desktop computer like Printers, Routers, Modems, Keyboards, etc.

The above definitions of the Microcontroller might seem complicated or confusing to newbies in Electronics or Embedded Systems but the concept will become clear as we move forward.

First, we will see the Rise of Microcontrollers, where you can find how the development to the Microcontroller took place.

Rise of Microcontrollers

Microprocessor, the invention that took the field of computation by storm. A Microprocessor is an Integrated Circuit (IC) that contains the Central Processing Unit (CPU). The earliest known Microprocessors are the Intel's 4004 and the Texas Instruments' TMS1000.

Since then, the computational power, complexity and power consumption kept on increasing in order to provide ultimate performance (Power Consumption must be discussed separately due to developments such as Low Power VLSI etc.).

For a Microprocessor to work, it needs a bunch of supporting hardware that can be found on a mother board. The hardware includes memory, ICs for peripheral devices, etc.

In the beginning itself, the Microprocessors ability to control other electronic equipment like Photocopiers is realized. The emphasis here is not on the computational power of the Microprocessor but rather on a control mechanism with less complex hardware and increased reliability.

This requirement paved way for integrating the minimum hardware required for complete functioning of a Processor on to a single chip i.e. same chip as the processor, to be precise.

This is the rise of Microcontrollers, an Integrated Circuit, which contains all the functions and hardware in order to make a complete computer system. Here, the computational power of the device is of less importance than the integration of all the components, including memory.

Basics of Microcontrollers

Basically, a Microcontroller consists of the following components.

- Central Processing Unit (CPU)
- Program Memory (ROM – Read Only Memory)
- Data Memory (RAM – Random Access Memory)
- Timers and Counters
- I/O Ports (I/O – Input/Output)
- Serial Communication Interface
- Clock Circuit (Oscillator Circuit)
- Interrupt Mechanism

Most modern Microcontrollers might contain even more peripherals like SPI (Serial Peripheral Interface), I2C (Inter Integrated Circuit), ADC (Analog to Digital Converter), DAC (Digital to Analog Converter), CAN (Controlled Area Network), USB (Universal Serial Bus), and many more.

The CPU (Central Processing Unit) in a Microcontroller performs the arithmetic, logic, math and data-oriented function, similar to CPU in the Microprocessor. The difference between a Microprocessor and Microcontroller is that a Microprocessor need to be interface with external memory and other I/O Interfaces to work as a computer whereas, a Microcontroller has all the required peripherals on the same chip as the CPU.

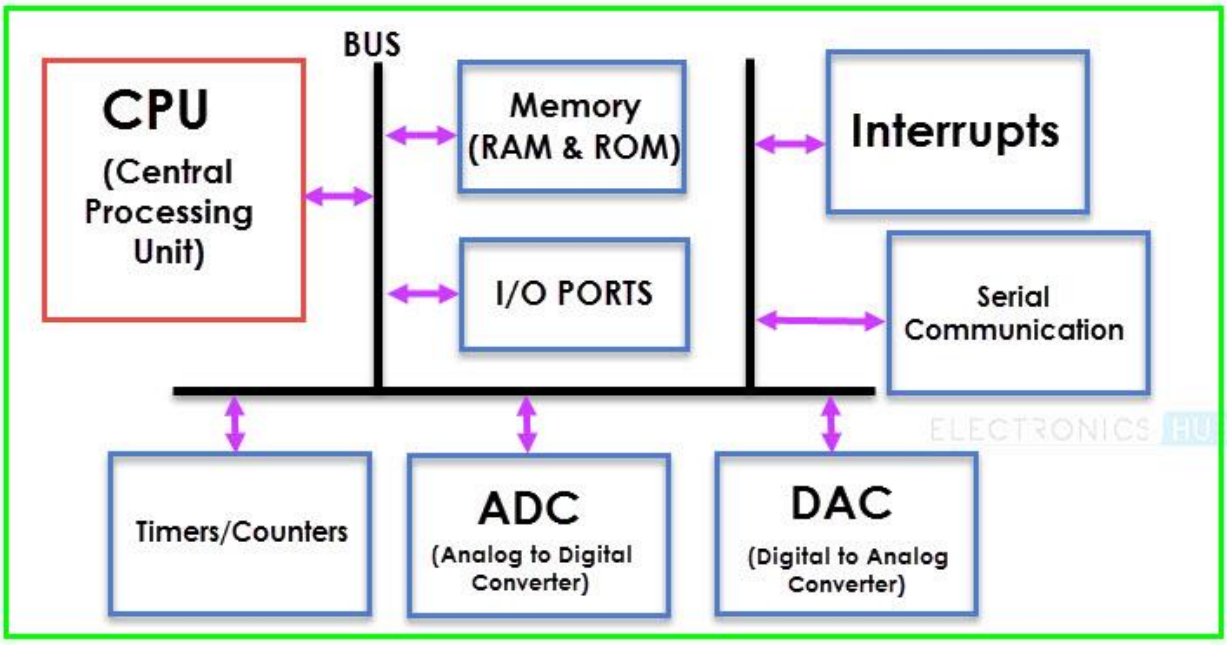
The integration of features like ADC, DAC etc. on the same chip as the CPU makes it more efficient and cheaper than to use a separate ADC Chip.

Developing a Computer Controlled System involves design of the Hardware and also writing an efficient Software Program. Since a Microcontroller has all the hardware, that are required to make a computer-controlled system on a single chip, using a Microcontroller will drastically reduce the efforts and time spent on hardware design and wiring.

Basic Structure of a Microcontroller

You might have seen the basic structure of a Microcontroller many times. If you have already seen the structure of Microcontroller and the basic components of a Microcontroller before, then consider this as a revision. If you haven't seen it, then it is very important to get an idea about the basic structure of a Microcontroller.

The following image shows the Basic Structure of a Microcontroller.



From the above image, you can understand that the three important (or major) components of a Microcontroller are:

- The CPU (Central Processing Unit)
- The Memory and
- The I/O Ports

This doesn't mean that other components are of less importance. But these can be considered as supporting devices. We will now see each of the Basic Components of a Microcontroller mentioned in the above structure.

CPU

Central Processing Unit or CPU is the brain of the Microcontroller. It consists of an Arithmetic Logic Unit (ALU) and a Control Unit (CU). A CPU reads, decodes and executes instructions to perform Arithmetic, Logic and Data Transfer operations.

Memory

Any Computational System requires two types of Memory: Program Memory and Data Memory. Program Memory, as the name suggests, contains the program i.e. the instructions to be executed by the CPU. Data Memory on the other hand, is required to store temporary data while executing the instructions.

Usually, Program Memory is a Read Only Memory or ROM and the Data Memory is a Random-Access Memory or RAM. Data Memory is sometimes called as Read Write Memory (R/W M).

I/O Ports

The interface for the Microcontroller to the external world is provided by the I/O Ports or Input/Output Ports. Inputs device like Switches, Keypads, etc. provide information from the user to the CPU in the form of Binary Data.

The CPU, upon receiving the data from the input devices, executes appropriate instructions and gives response through Output Devices like LEDs, Displays, Printers, etc.

Bus

Another important component of a Microcontroller, but rarely discussed is the System Bus. A System bus is a group of connecting wire that connect the CPU with other peripherals like Memory, I/O Ports and other supporting components.

Timers/Counters

One of the important components of a Microcontroller are the Timers and Counters. They provide the operations of Time Delays and counting external events. Additionally, Timers and Counters can provide Function Generation, Pulse Width Modulation, Clock Control, etc.

Serial Port

One of the important requirements of a Microcontroller is to communicate with other device and peripherals (external). Serial Port proves such interface through serial communication. Most common serial communication implemented in Microcontrollers is UART.

Interrupts

A very important feature of a Microcontroller is Interrupts and its Interrupt Handling Mechanism. Interrupts can be external, internal, hardware related or software related.

ADC (Analog to Digital Converter)

Analog to Digital Converter or ADC is a circuit that converts Analog signals to Digital Signals. The ADC Circuit forms the interface between the external Analog Input devices and the CPU of the Microcontroller. Almost all sensors are analog devices and the analog data from these sensors must be converted in to digital data for the CPU to understand.

DAC (Digital to Analog Converter)

Digital to Analog Converter or DAC is a circuit, that works in contrast to an ADC i.e. it converts Digital Signals to Analog Signals. DAC forms the bridge between the CPU of the Microcontroller and the external analog devices.

Advantages of Microcontrollers

- A Microcontroller is a true device that fits the computer-on-a-chip idea.
- No need for any external interfacing of basic components like Memory, I/O Ports, etc.
- Microcontrollers doesn't require complex operating systems as all the instructions must be written and stored in the memory. (RTOS is an exception).
- All the Input/Output Ports are programmable.
- Integration of all the essential components reduces the cost, design time and area of the product (or application).

Disadvantages of Microcontrollers

- Microcontrollers are not known for their computation power.
- The amount of memory limits the instructions that a microcontroller can execute.
- No Operating System and hence, all the instruction must be written.

Applications of Microcontrollers

There are huge number of applications of Microcontrollers. In fact, the entire embedded systems industry is dependent on Microcontrollers. The following are few applications of Microcontrollers.

- Front Panel Controls in devices like Oven, washing Machine etc.
- Function Generators
- Smoke and Fire Alarms
- Home Automation Systems
- Automatic Headlamp ON in Cars
- Speed Sensed Door Locking System

Internet of things

Definition and interpretation of IoT

The Internet of Things (IoT) is one of the most widely used phrases in modern computing developments. Although often referred to as a technology, it is more accurately a platform for connecting objects (via sensing) so that data gathered about them might be used to analyze, interpret, decide and act on that data and other associated information.

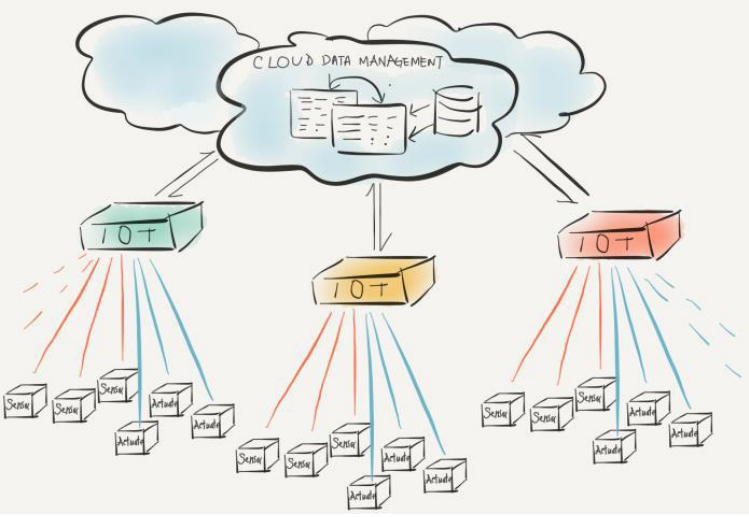
According to TechTarget's Internet of Things Network Internet it is defined as follows:
The **Internet of Things**, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Additional definitions by organizations such as IEEE and OASIS provide the same sense of IoT being a mechanism for connecting physical items to the internet:
A network of items—each embedded with sensors—which are connected to the Internet.
A system where the Internet is connected to the physical world via ubiquitous sensors.

In its simplest form, IoT is generally understood to comprise three key component levels:

- Edge functionality – sensors (and actuators) connected to physical objects and machines.
- Data Gateways – mechanisms for receiving sensed data from edge devices and transmitting to edge devices. Also capable of transmitting and receiving information from networked servers.
- Data Management & Analysis – (cloud) server-based collection, linking, analysis and use of data.

Connecting these levels are the necessary communication systems. Edge to Gateway communications are supported by a myriad of wired and wireless communication methods



Technologies have made IoT possible?

While the idea of IoT has been in existence for a long time, a collection of recent advances in a number of different technologies has made it practical.

- **Access to low-cost, low-power sensor technology.** Affordable and reliable sensors are making IoT technology possible for more manufacturers.
- **Connectivity.** A host of network protocols for the internet has made it easy to connect sensors to the cloud and to other "things" for efficient data transfer.
- **Cloud computing platforms.** The increase in the availability of cloud platforms enables both businesses and consumers to access the infrastructure they need to scale up without actually having to manage it all.
- **Machine learning and analytics.** With advances in machine learning and analytics, along with access to varied and vast amounts of data stored in the cloud, businesses can gather insights faster and more easily. The emergence of these allied technologies continues to push the boundaries of IoT and the data produced by IoT also feeds these technologies.
- **Conversational artificial intelligence (AI).** Advances in neural networks have brought natural-language processing (NLP) to IoT devices (such as digital personal assistants Alexa, Cortana, and Siri) and made them appealing, affordable, and viable for home use.

What is industrial IoT?

Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially with respect to instrumentation and control of sensors and devices that engage cloud technologies. Recently, industries have used machine-to-machine communication (M2M) to achieve wireless automation and control. But with the emergence of cloud and allied technologies (such as analytics and machine learning), industries can achieve a new automation layer and with it create new revenue and business models. IIoT is sometimes called the fourth wave of the industrial revolution, or Industry 4.0. The following are some common uses for IIoT:

- Smart manufacturing
- Connected assets and preventive and predictive maintenance
- Smart power grids
- Smart cities
- Connected logistics
- Smart digital supply chains

How does IIoT work?

IIoT is a network of intelligent devices, via networks linked to databases, which monitors, collects, exchanges and analyzes data. Typically, an IIoT system consists of:

- Intelligent equipment that can measure and store information about itself and communicate.
- A data communication structure such as public internet or individual networks.
- Intelligent applications that create useful information from raw data and utilize it to control and optimize processes.
- Interface and analysis tools that provide people with the opportunity to utilize the information for qualified decision-making.

The connected equipment sends information directly to the IIoT infrastructure, where it is transformed into useful information on the status or performance of a machine a group of machines or of the whole plant. The information can then be used to optimize production and supply processes as well as foresee maintenance.

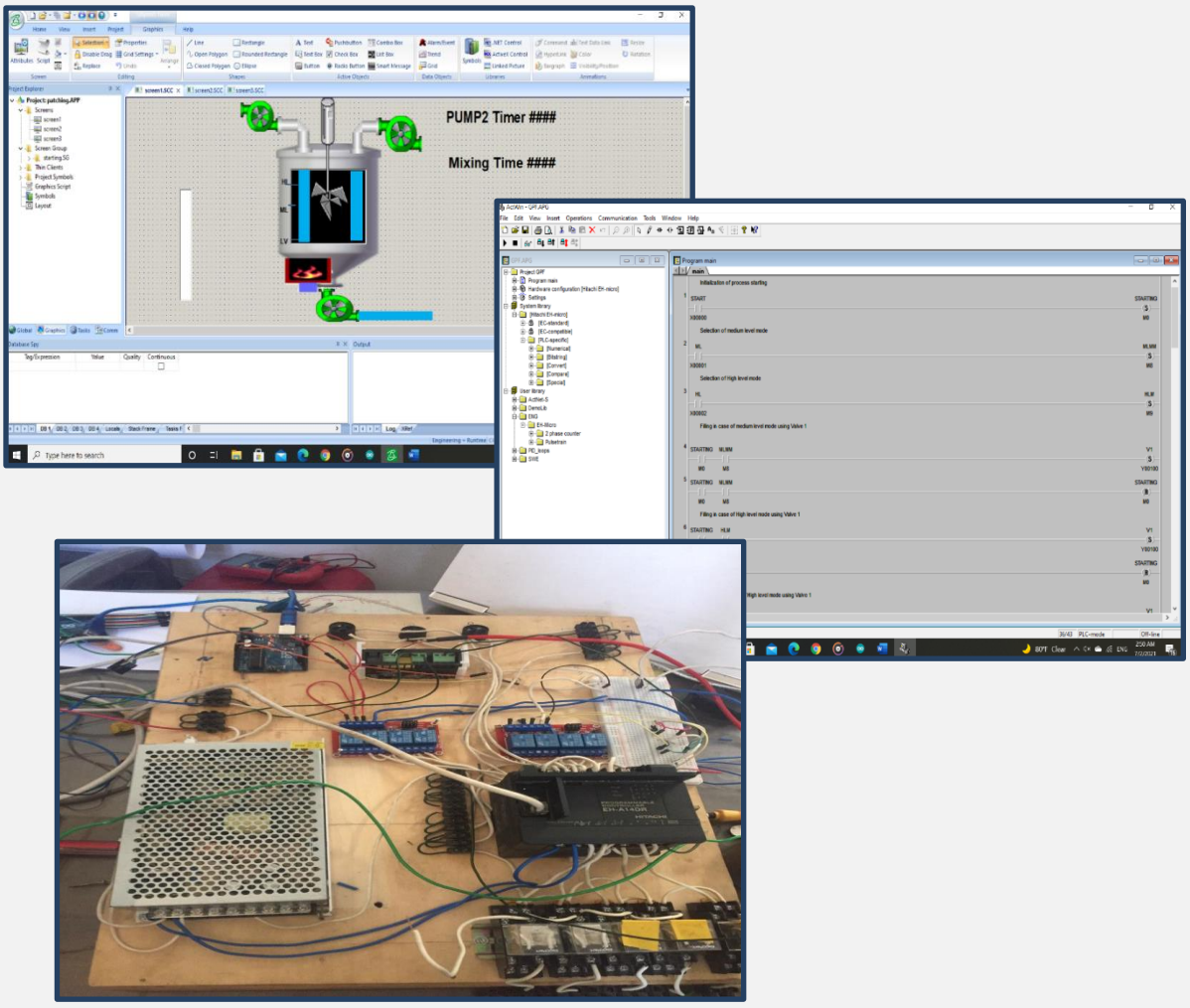
What is Arduino in IoT?

Arduino acts as the brain of the system and processes the data from the sensor. Arduino is an open source hardware platform that is readily available for hobbyists & enthusiasts across the globe to build projects. It comes with an ATMEGA microcontroller that processes the data and facilitates the proper working of the IoT system. And the beauty is that the Arduino can be programmed ‘n’ number of times making it possible for you to build various types of IoT projects just by changing a simple code.

You need to use C++ language for Arduino programming. Also, IDE software is needed for Arduino based IoT projects. And you need to use ESP-8266 WIFI module to establish the WIFI communication between the Arduino and cloud platform.

PART 2

Project Design



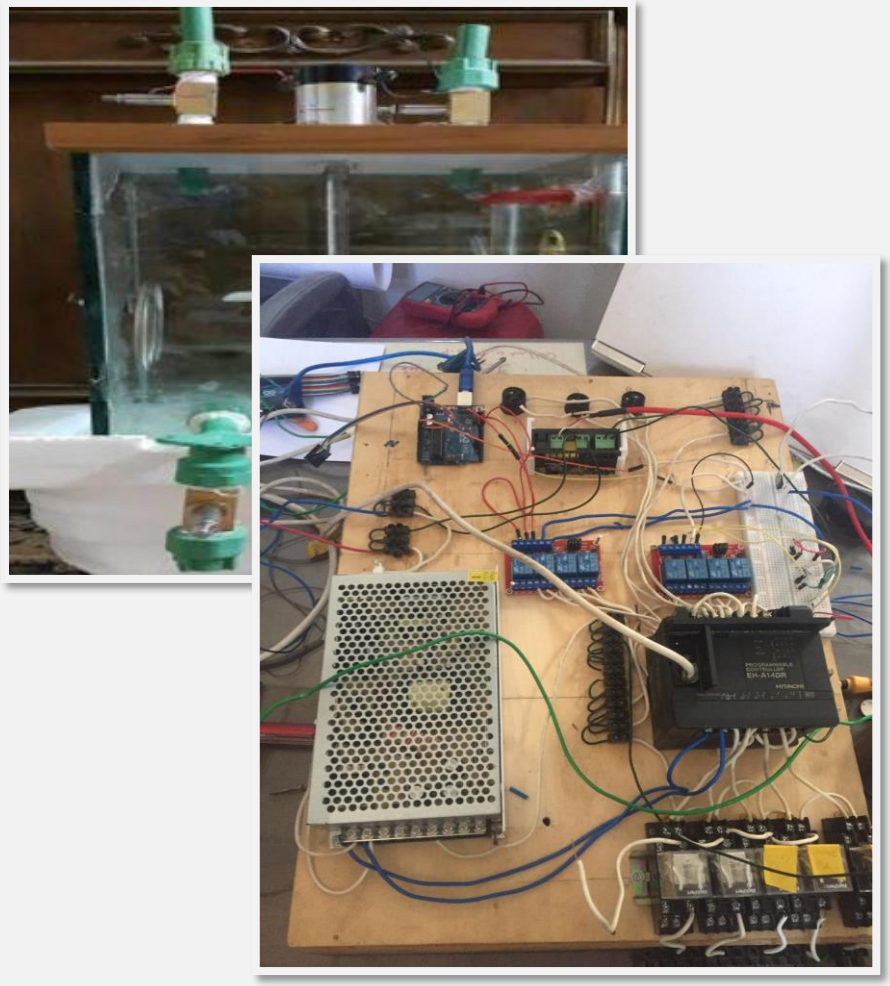
Chapter 4

Project Description and Hardware

Overview

In this chapter we will discuss:

- **The Project description**
- **PLC hardware and wiring**
- **Arduino uno and WIFI module wiring**
- **Temperature controller wiring**
- **Output devices wiring**

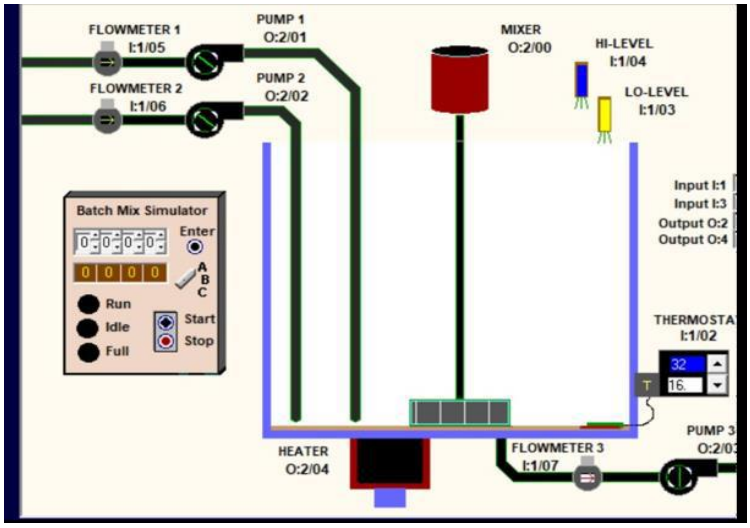


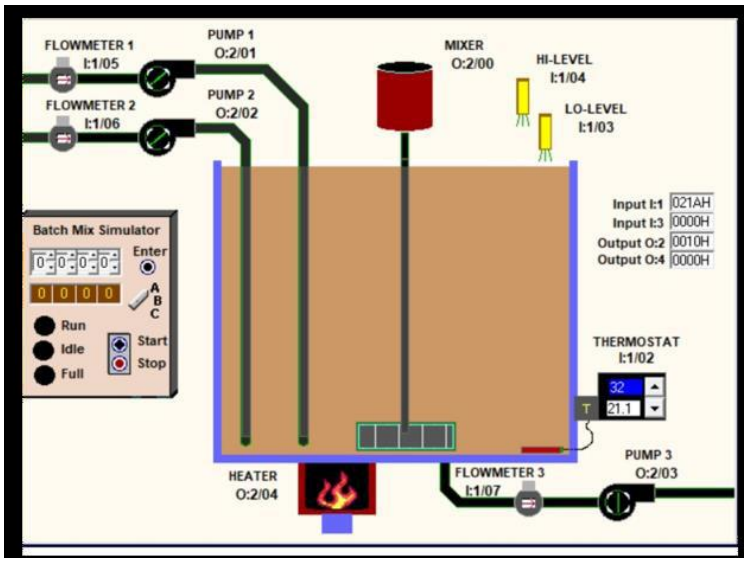
Chapter 4 : Project Description and Hardware

Project Description

Our project contains the four main processes in many industrial processes which are Tank filling, heating, mixing and emptying. The process will be controlled by digital control technique by using programmable logic controller. We defined the inputs and the outputs of the process and the sequence of the whole process.

In our First 4 weeks we developed a simulation for the process.





We have implemented different methods to control this process. The process begins with choosing the required level of the liquid either high or medium. This step can be done with three ways; PBs connected to the plc, SCADA software or IOT android application.

After we choose the required level the liquid valve will be open until the liquid reaches the level. We will know the using the level sensor. In our project we used ultrasonic sensor to sense the level of the liquid. We have tried different methods to sense the level throughout our working on the project like using two electrodes in each level to give indication but we figured out that the ultrasonic sensor is the best option due to its sensitivity and accuracy.

The ultrasonic sensor is connected to an Arduino uno board (see the connection in chapter 4) then the three levels indication outputs are connected to Arduino relays and the Arduino relays are connected to the PLC.

In the SCADA software there will be indication for the level of the liquid too through graphical illustration.

After the Arduino relay send the signal to PLC it will activate the heater for amount of time until it reaches the required temperature. This temperature is previously set to some value using temperature controller.

When the temperature sensor read the required temperature, the controller will send a signal to PLC to stop the heater and start valve 2. The valve will be open for some time. And the mixer motor will start to operate.

Finally, after mixing the two liquids together valve 3 will open to empty the tank. Valve 3 will be open until the level sensor indicate that the level reaches below low level.

In later sections in this chapter we will discuss every component we used throughout our project. We will give information about the specification of each component and how every component is connected.

I/O list

Inputs	Outputs
3 PBs (Start- M level – H level)	3 Valves
Ultrasonic Sensor	Motor (mixer)
Temperature sensor	Heater

PLC

For completion of this project we used Hitachi EH-A14EDR PLC. We will discuss in detail all the specifications and wiring of this PLC in this section.

Functional Specification

Item		10-point type	14-point type	20-point type	23-point type	28-point type	40-point type	64-point type
RS-232C port		1	1	1	1	1	1	1
RS-422/485 port		–	–	1(Optinal)	1	1	1(Optinal)	1(Optinal)
High-speed counter		10kHz 1-phase 3ch, 1-phase 2ch or 2-phase 1ch	10kHz 1-phase 4ch, 1-phase 2ch or 2-phase 1ch + 1phase 1ch	100kHz 1-phase 4ch, 2-phase 2ch or 2-phase 1ch + 1phase 2ch	10kHz 1-phase 4ch, 1-phase 2ch or 2-phase 1ch + 1phase 1ch		100kHz 1-phase 4ch, 2-phase 2ch or 2-phase 1ch + 1phase 2ch	
Interruption input		3 points	4 points					
PWM output		2kHz (in total)		65kHz (each channel)	2kHz (in total)		65kHz (each channel)	65kHz (each channel)
Pulse train		5kHz (in total)		65kHz (each channel)	5kHz (in total)		65kHz (each channel)	65kHz (each channel)
Analog input		8-bit : 1ch *1	–	–	12bit:2ch(0-10V or 0-20mA)	–	–	–
Analog output		–	–	–	12bit:1ch(0-10V or 0-20mA)	–	–	–
Potentiometer		–	10-bit : 2ch	–	10-bit : 2ch		–	–
Battery(optional)		–	–	EH-MBATL	EH-MBAT or EH-MBATLC		EH-MBATL	
Real-time clock		–	–	Yes	Yes	Yes	Yes	Yes
Digital filter		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Power source	AC100/200V	No	Yes	Yes	Yes	Yes	Yes	Yes
	DC24V	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Input	DC	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	AC	No	Yes	No	No	Yes	No	No
Output	TR DC24V	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	RY	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	SSR	No	Yes	No	No	Yes	No	No
Positioning expansion		No	No	Yes	Yes	Yes	Yes	Yes

Input/Output Specifications

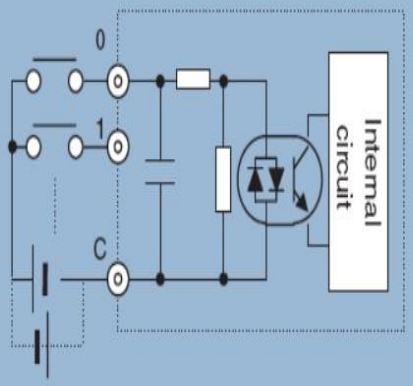
It has 8 input Points and 6 output points. The input voltage is 24V DC. The output points are relay outputs type.

Type	Model Name	Power		Input	Input Point		Output	Output Point						
		24V DC	100/200V AC		24V DC	100/200V AC		Relay Output	Transistor Output		Transistor Output (source ESCP)		Transistor Output(source)	SSR
									Low Current	High Current	Low Current	High Current		
					①	④		⑤	⑦	⑧	⑨	⑩	⑪	⑬
10 Points	EH-D10DT	○		24V DC x 6	6 (1)		Transistor x 4(sink)		4 (1)					
	EH-D10DTP	○		24V DC x 6	6 (1)		Transistor x 4(source)		4 (1)					
	EH-D10DR	○		24V DC x 6	6 (1)		Relay x 4	4 (1)						
14 Points	EH-D14DT	○		24V DC x 8	8 (2)[4,4]		Transistor x 6(sink)		4 (1)	2				
	EH-D14DTP	○		24V DC x 8	8 (2)[4,4]		Transistor x 6(source)		4 (1)	2				
	EH-D14DTPS	○		24V DC x 8	8 (2)[4,4]		Transistor (source ESCP) x 6				4 (1)	2		
	EH-D14DR	○		24V DC x 8	8 (2)[4,4]		Relay x 6	6 (3)[1,1,4]						
	EH-A14DR		○	24V DC x 8	8 (2)[4,4]		Relay x 6	6 (3)[1,1,4]						
	EH-A14AS		○	AC x 8		8 (2)[4,4]	SSR x 6							6 (2)[2,4]
23 Points	EH-D23DRP	○		24V DC x 13 Analog x 2(12bits)	13 (3) [4,4,5]		Relay x 9 Transistor x 1(source) Analog 1(12bits)	9 (5) [4,1,1,1,2]					1 (1)	
	EH-A23DRP		○	24V DC x 13 Analog x 2(12bits)	13 (3) [4,4,5]		Relay x 9 Transistor x 1(source) Analog 1(12bits)	9 (5) [5,1,1,1,2]					1 (1)	
	EH-A23DR		○	24V DC x 13 Analog x 2(12bits)	13 (3) [4,4,5]		Relay x 10 Analog 1(12bits)	10 (6) [1,4,1,1,1,2]						
28 points	EH-D28DT	○		24V DC x 16	16 (4) [4,4,4,4]		Transistor x 12(sink)		8 (2)[6,6]	4				
	EH-D28DTP	○		24V DC x 16	16 (4) [4,4,4,4]		Transistor x 12(source)		8 (2)[6,6]	4				
	EH-D28DTPS	○		24V DC x 16	16 (4) [4,4,4,4]		Transistor (source ESCP) x 12				8 (2)[6,6]	4		
	EH-D28DRP	○		24V DC x 16	16 (4) [4,4,4,4]		Relay x 11 Transistor x 1(source)	11 (6) [4,1,1,1,1,3]					1 (1)	
	EH-D28DR	○		24V DC x 16	16 (4) [4,4,4,4]		Relay x 12	12 (7) [1,4,1,1,1,1,3]						
	EH-A28DRP		○	24V DC x 16	16 (4) [4,4,4,4]		Relay x 11 Transistor x 1(source)	11 (6) [4,1,1,1,1,3]					1 (1)	
	EH-A28DR		○	24V DC x 16	16 (4) [4,4,4,4]		Relay x 12	12 (7) [1,4,1,1,1,1,3]						
	EH-A28AR		○	AC x 16		16 (4) [4,4,4,4]	Relay x 12	12 (7) [1,4,1,1,1,1,3]						
	EH-A28AS		○	AC x 16		16 (4) [4,4,4,4]	SSR x 12							12 (4) [2,4,2,4]

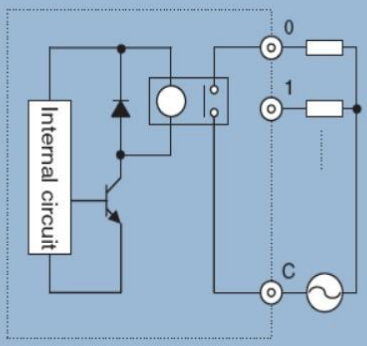
DC input

Item		Specification
Input voltage		24 V DC
Allowable input voltage range		0 to 30 V DC
Input impedance		Approx. 2.8 kΩ
Input current		Approx. 7.5 mA
Operating voltage	ON voltage	15 V DC (min) / 4.5 mA (max)
	OFF voltage	5 V DC (max) / 1.5 mA (max)
Input lag	OFF → ON	0.5 to 20 ms (configurable)
	ON → OFF	0.5 to 20 ms (configurable)
Polarity		None
Insulation system		Photocoupler insulation
Input display		LED (green)
External connection		10-point type: fixed type terminal block 14/23/28-point types: Removable type screw terminal block (M3)

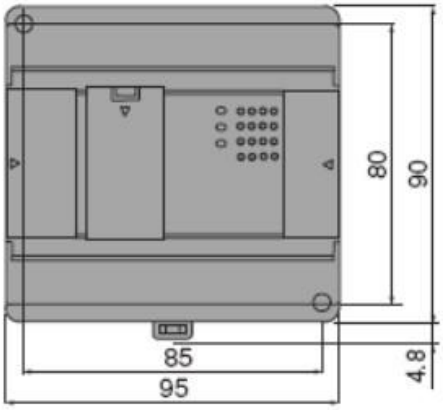
Circuit diagram

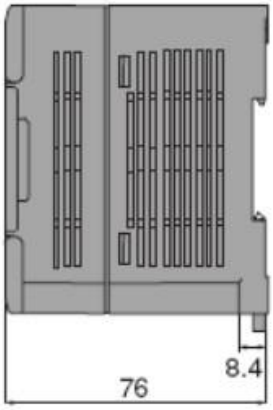


Relay output

Item		Specification	Circuit diagram
Rated load voltage		5 to 250 V AC, 5 to 30 V DC	
Minimum switching current		10 mA	
Leak current		15 mA or less	
Maximum load current	1 circuit	2 A (24 V DC, 240 V AC)	
	1 common	5 A	
Output response time	OFF→ ON	15 ms (max)	
	ON→ OFF	15 ms (max)	
Surge removing circuit		None	
Fuse		None	
Insulation system		Relay insulation	
Output display		LED (green)	
Externally supplied power (for driving the relays)		Not necessary	
Contact life		20,000,000 times (mechanical) 200,000 times (electrical: 2 A)	
Insulation		1,500 V or more (external-internal) 500 V or more (external-external)	
External connection		Removable type screw terminal block (M3)	

Dimensions

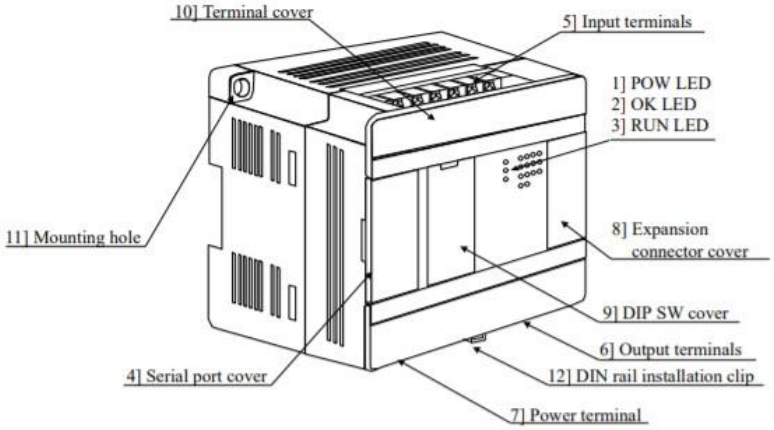


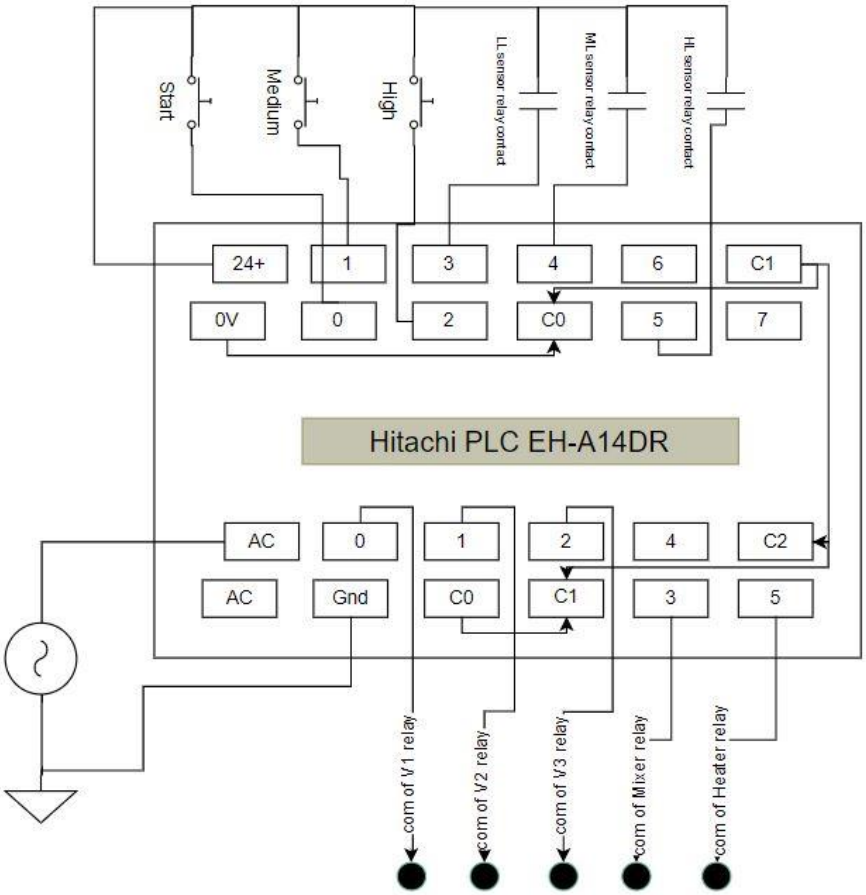


Network Compatibility

RS-232C port as standard (Port 1)

Wiring of PLC





Arduino Uno and WIFI module

Overview

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an 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 AC-to-DC adapter or battery to get started.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

There are a couple of other pins on the board:

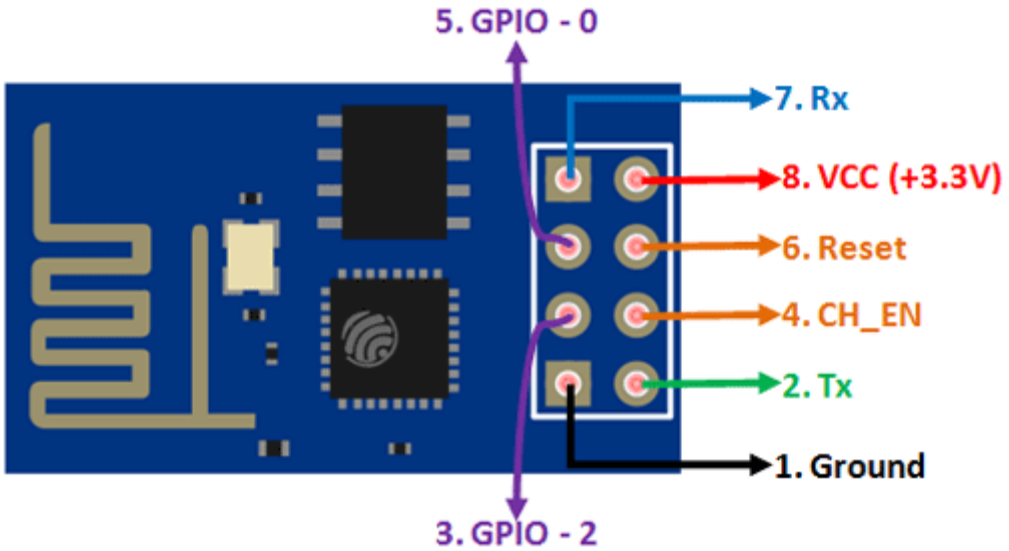
- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

ESP8266 - WIFI Module

ESP8266 Pin Configuration



Pin Number	Pin Name	Alternate Name	Normally used for	Alternate purpose
1	**Ground**	-	Connected to the ground of the circuit	-
2	**TX**	GPIO – 1	Connected to Rx pin of programmer/uC to upload program	Can act as a General-purpose Input/output pin when not used as TX
3	**GPIO-2**	-	General purpose Input/output pin	-
4	**CH_EN**	-	Chip Enable – Active high	-
5	**GPIO - 0**	Flash	General purpose Input/output pin	Takes module into serial programming when held low during start up
6	**Reset**	-	Resets the module	-
7	**RX**	GPIO - 3	General purpose Input/output pin	Can act as a General-purpose Input/output pin when not used as RX
8	**Vcc**	-	Connect to +3.3V only	

ESP8266-01 Features

- Low cost, compact and powerful Wi-Fi Module
- Power Supply: +3.3V only
- Current Consumption: 100mA
- I/O Voltage: 3.6V (max)
- I/O source current: 12mA (max)
- Built-in low power 32-bit MCU @ 80MHz
- 512kB Flash Memory
- Can be used as Station or Access Point or both combined
- Supports Deep sleep (<10uA)

- Supports serial communication hence compatible with many development platforms like Arduino
- Can be programmed using Arduino IDE or AT-commands or Lua Script

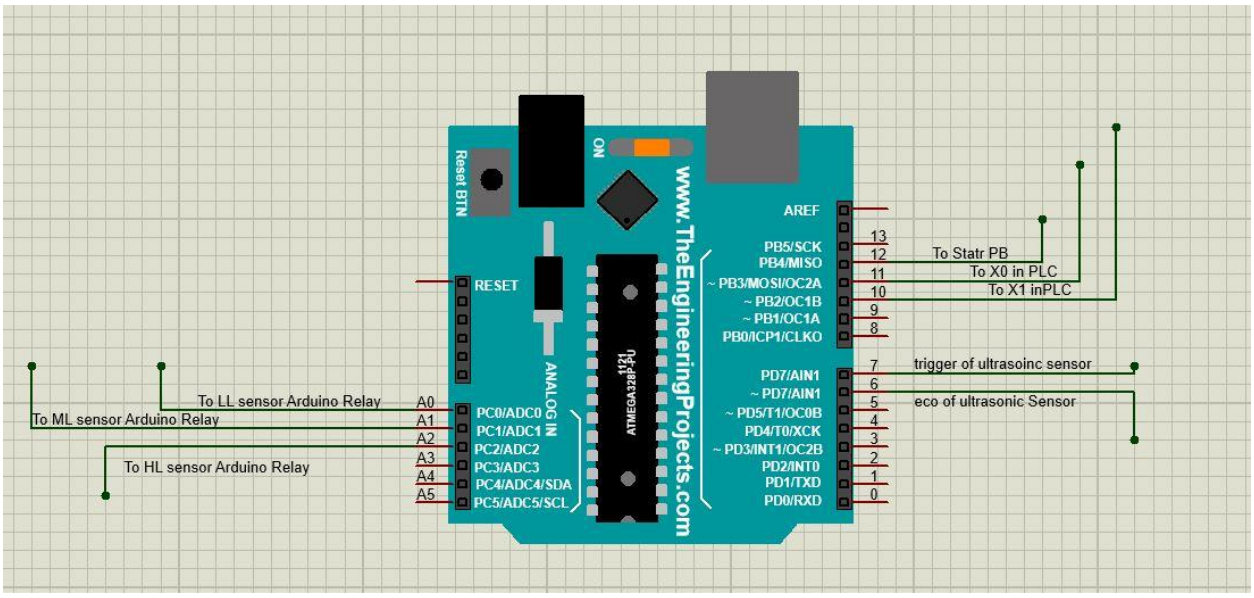
Applications

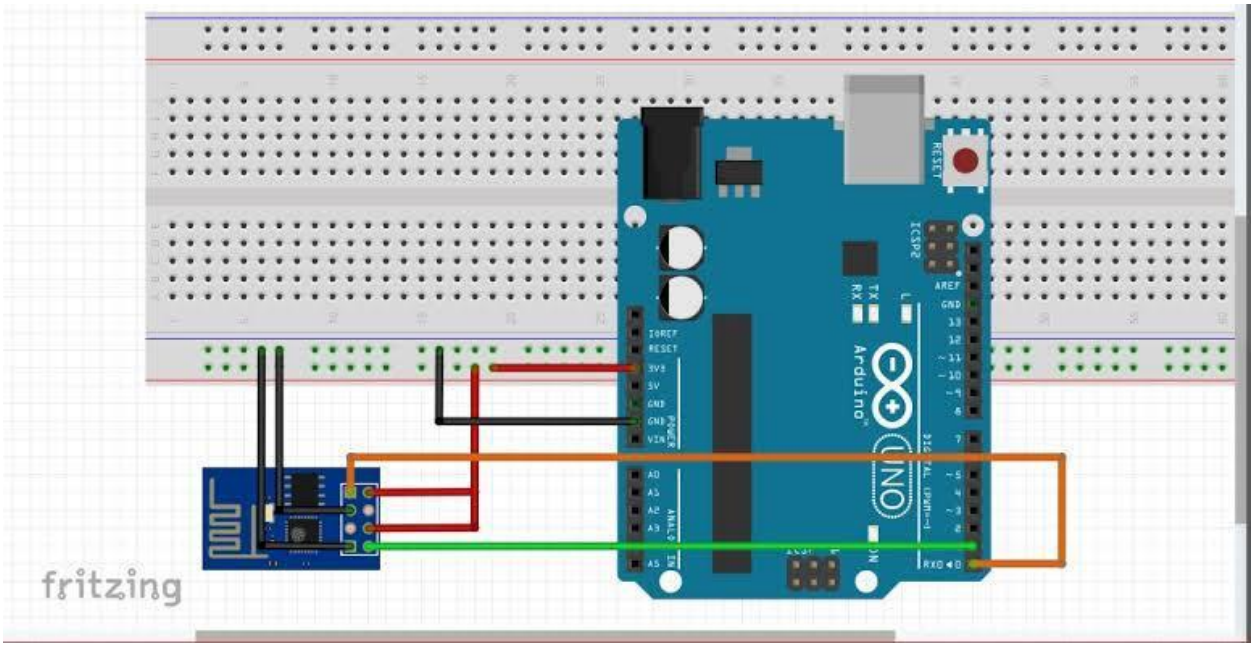
- IOT Projects
- Access Point Portals
- Wireless Data logging
- Smart Home Automation
- Learn basics of networking
- Portable Electronics
- Smart bulbs and Sockets

Where to use ESP8266-01

The **ESP8266** is a very user friendly and low-cost device to provide internet connectivity to your projects. The module can work both as an Access point (can create hotspot) and as a station (can connect to Wi-Fi), hence it can easily fetch data and upload it to the internet making **Internet of Things** as easy as possible.

Wiring of Arduino uno and WIFI module

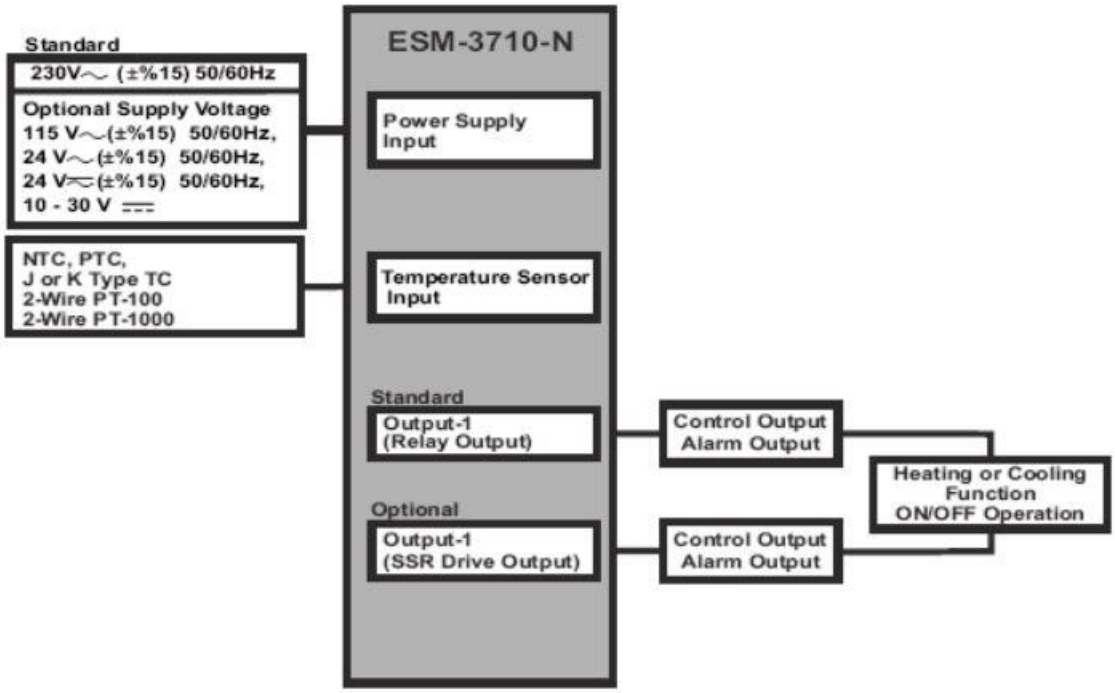


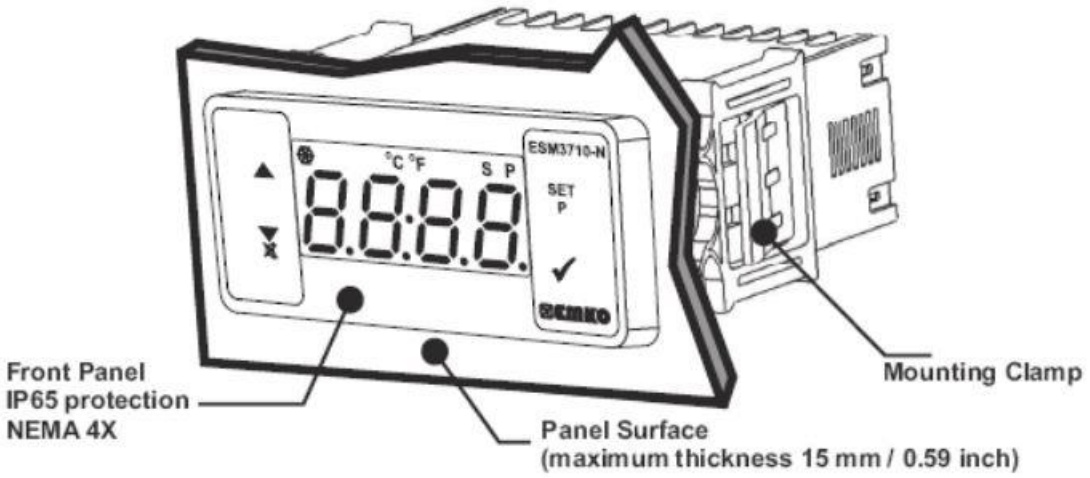


Temperature controller

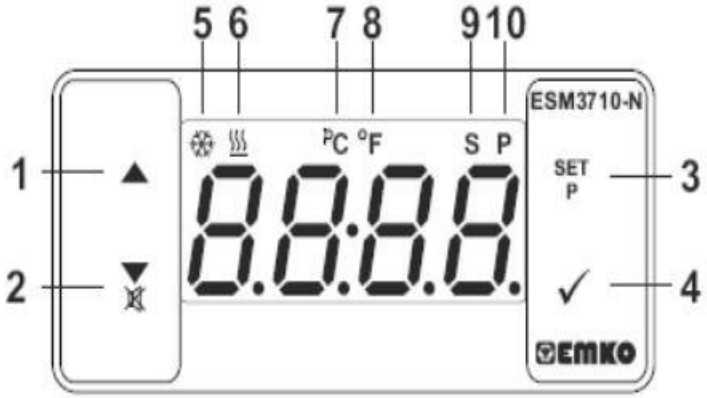
ESM-3710N series temperature controllers are designed for measuring and controlling temperature. They can be used in many applications with their ON / Off control form, heating and cooling control form and easy-use properties.

General specifications





Front Panel Definition and Accessing to the Menus



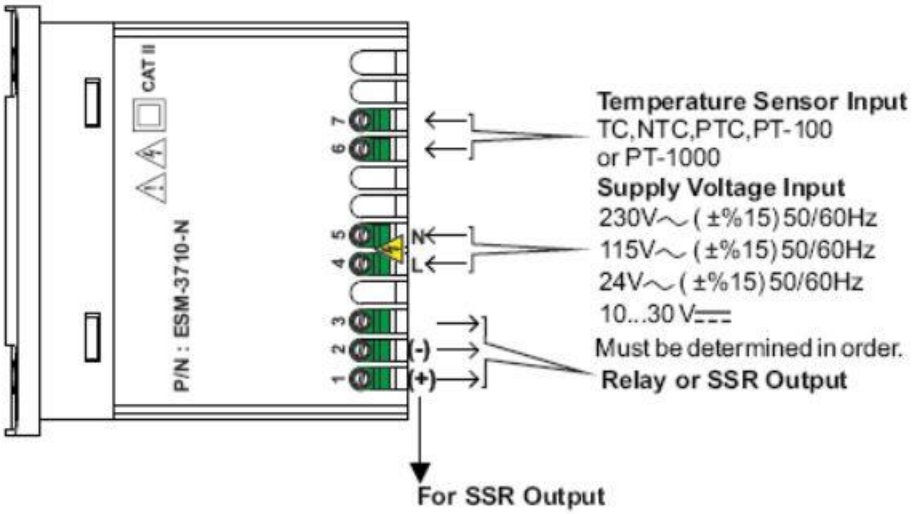
BUTTON DEFINITIONS

1. **Increment Button**
   It is used to increase the value in the Set screen and Programming mode.
2. **Decrement, Silencing Buzzer and Downloading to Prokey Button**
   It is used to decrease the value in the Set screen and Programming mode.
   It is used to silence the buzzer.
3. **Set Button**

In the main operation screen; if this button pressed, set value will be displayed. Value can be changed using increment and decrement buttons. When Enter button pressed, value is saved and returns back to main operating screen.
To access the programming screen; in the main operation screen, press this button for 5 seconds.

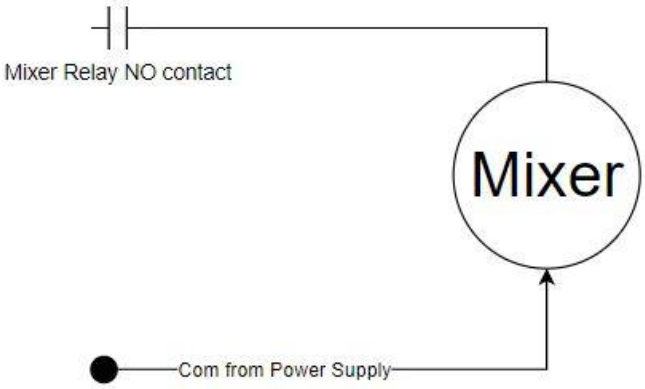
4. **Enter Button**
   It is used to saving value in the Set screen and programming screen.

Wiring of temperature controller



DC Motor

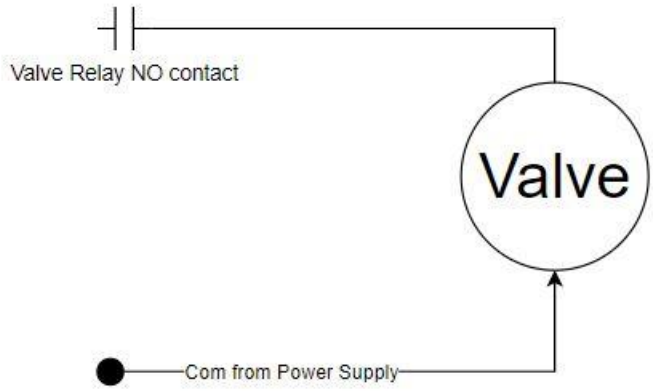
Wiring of DC motor (mixer)



Solenoid Valves

In our project we use 3 solenoid valves. Each one of them related to different stages of our project. One valve is for filling the water and other one is for filling the other liquid and the last one is for emptying the tank.

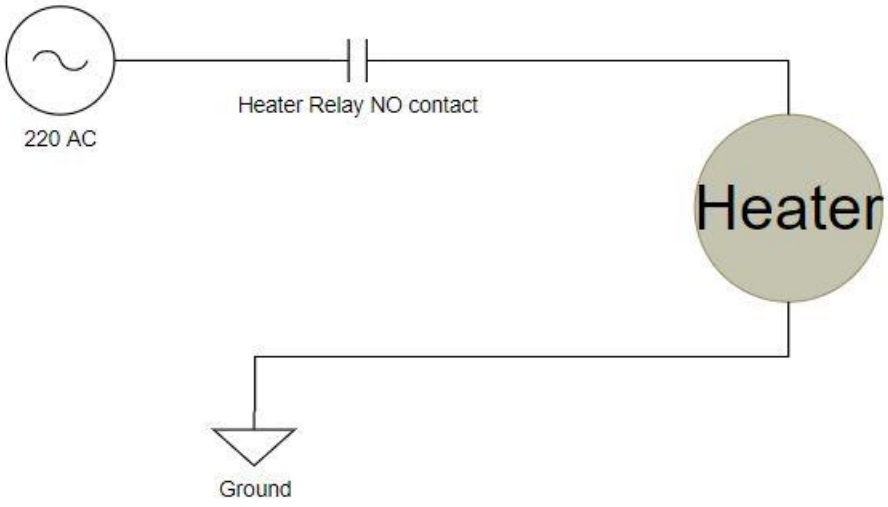
Wiring of solenoid Valves



Heater

220 VAC ,1500 Watt

Wiring of heater



Ultrasonic Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is D = ½ T x C (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).

Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (though the physical components are still affected by variables such as heat).

Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories). Most notably, ultrasonic technology has enabled the medical industry to produce images of internal organs, identify tumors, and ensure the health of babies in the womb.

Chapter 5

Project Software

Overview

In this chapter we will discuss:

- **The Project Ladder program**
- **SCADA Programming**
- **IOT Programming**
- **Arduino software and programming**

Chapter 5: Project Software

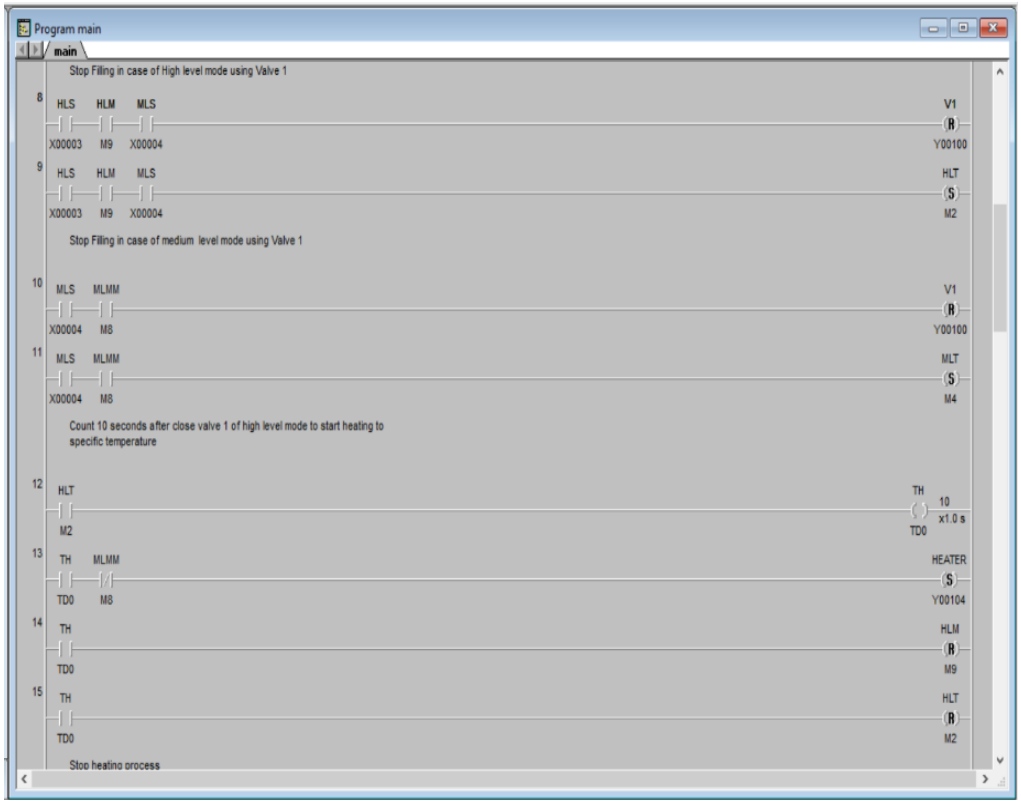
PLC programming

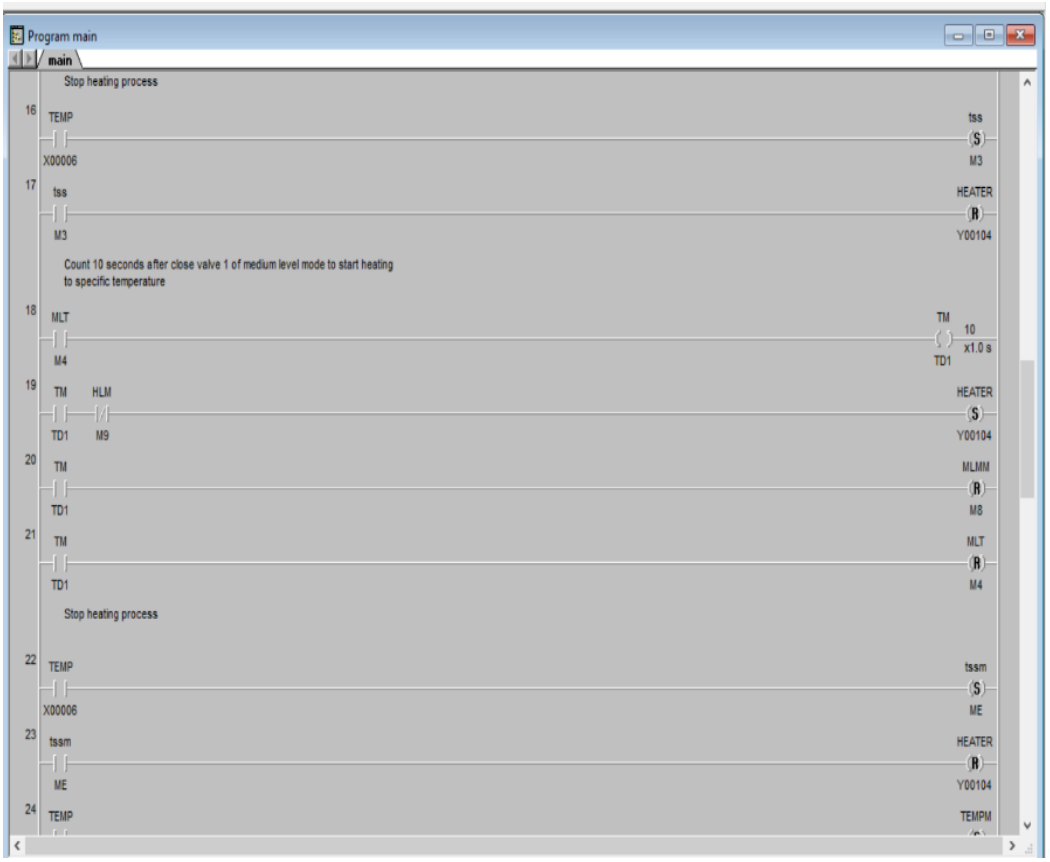
The used plc is HITACHI EH-A14DR and its software for programming is ACTWIN software

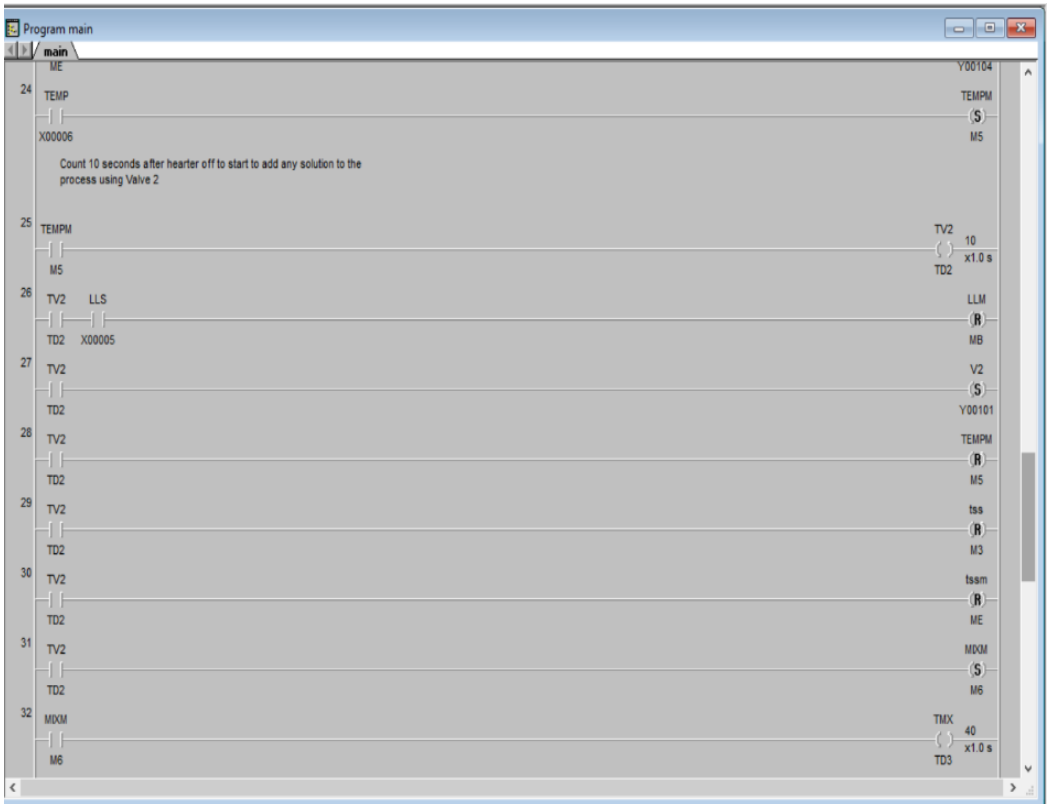
Table: Inputs and outputs with plc

Actual Inputs	Inputs Addresses	Actual Outputs	Outputs addresses
Start P. B	**X0**	**Valve 1**	**Y00**
M.L Selection P. B	**X1**	**Valve 2**	**Y1**
H.L Selection P. B	**X2**	**Valve 3**	**Y2**
H.L Sensor	**X3**	**Mixer**	**Y4**
M.L Sensor	**X4**	**Heater**	**Y3**
L.L Sensor	**X5**		
Temperature Sensor	**X6**		

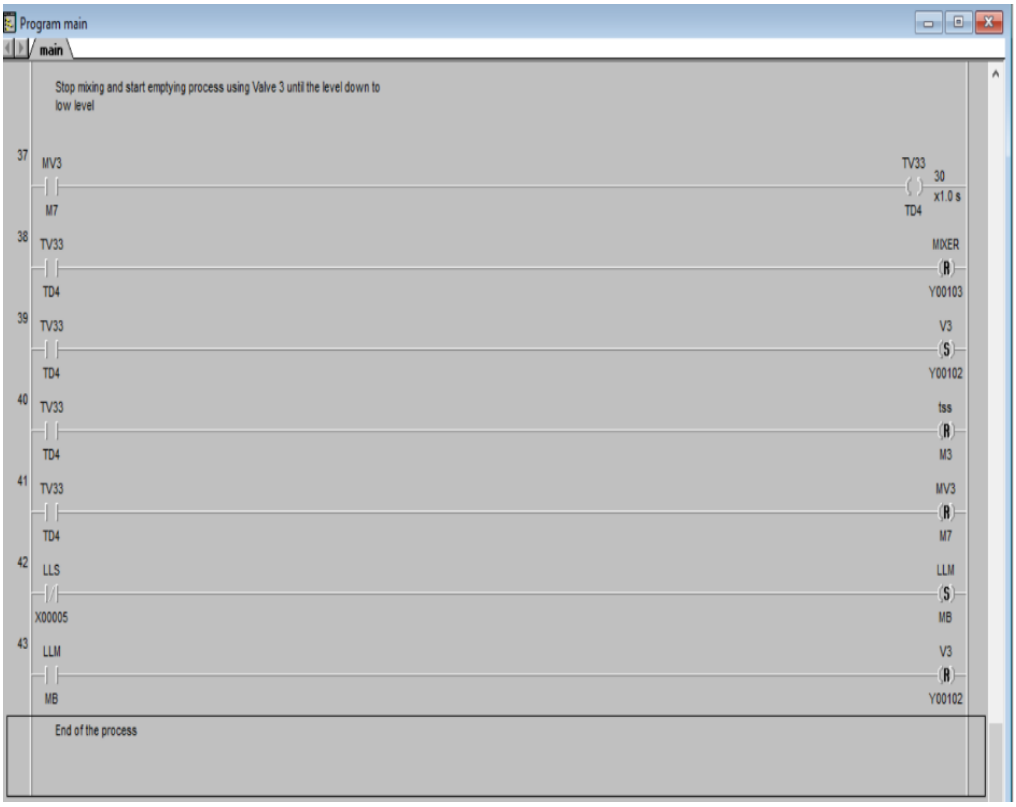
The ladder diagram which describe the process




Program main
main
Stop heating process
16
TEMP
X00006
tss
M3
17
tss
M3
HEATER
Y00104
Count 10 seconds after close valve 1 of medium level mode to start heating
to specific temperature
18
MLT
M4
TM
TD1
10
x1.0 s
19
TM
TD1
HLM
M9
HEATER
Y00104
20
TM
TD1
MLMM
M8
21
TM
TD1
MLT
M4
Stop heating process
22
TEMP
X00006
tssm
ME
23
tssm
ME
HEATER
Y00104
24
TEMP
TEMPM


Program main
main
ME
Y00104
24
TEMP
X00006
TEMPM
M5
Count 10 seconds after hearter off to start to add any solution to the
process using Valve 2
25
TEMPM
M5
TV2
TD2
10
x1.0 s
26
TV2
TD2
LLS
X00005
LLM
MB
27
TV2
TD2
V2
Y00101
28
TV2
TD2
TEMPM
M5
29
TV2
TD2
tss
M3
30
TV2
TD2
tssm
ME
31
TV2
TD2
MDXM
M6
32
MDXM
M6
TMX
TD3
40
x1.0 s


Program main
main
Start Mixing process for period of 30 seconds
33 TMX TD3 MIXER Y00103
34 TMX TD3 V2 Y00101
35 TMX TD3 MIXM M6
36 TMX TD3 MV3 M7
Stop mixing and start emptying process using Valve 3 until the level down to low level
37 MV3 M7 TV33 TD4 30 x1.0 s
38 TV33 TD4 MIXER Y00103
39 TV33 TD4 V3 Y00102
40 TV33 TD4 tss M3
41 TV33 MV3

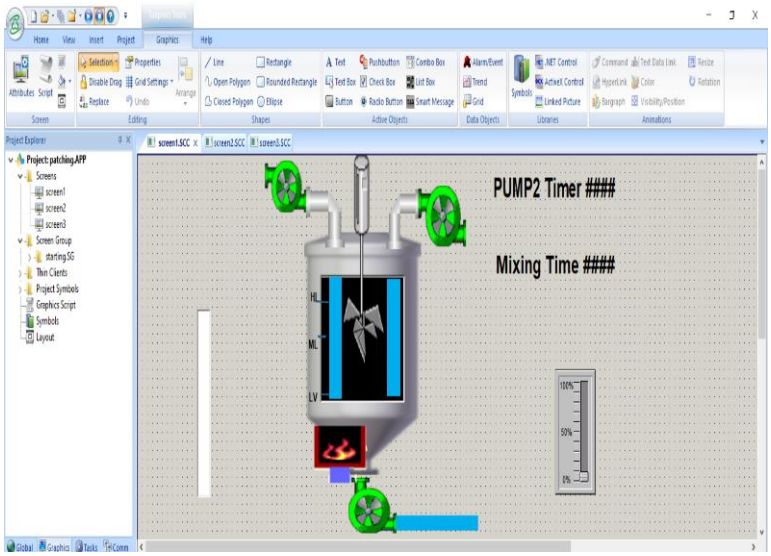

Program main
main
Stop mixing and start emptying process using Valve 3 until the level down to low level
37 MV3 M7 TV33 TD4 30 x1.0 s
38 TV33 TD4 MIXER Y00103
39 TV33 TD4 V3 Y00102
40 TV33 TD4 tss M3
41 TV33 TD4 MV3 M7
42 LLS X00005 LLM MB
43 LLM MB V3 Y00102
End of the process

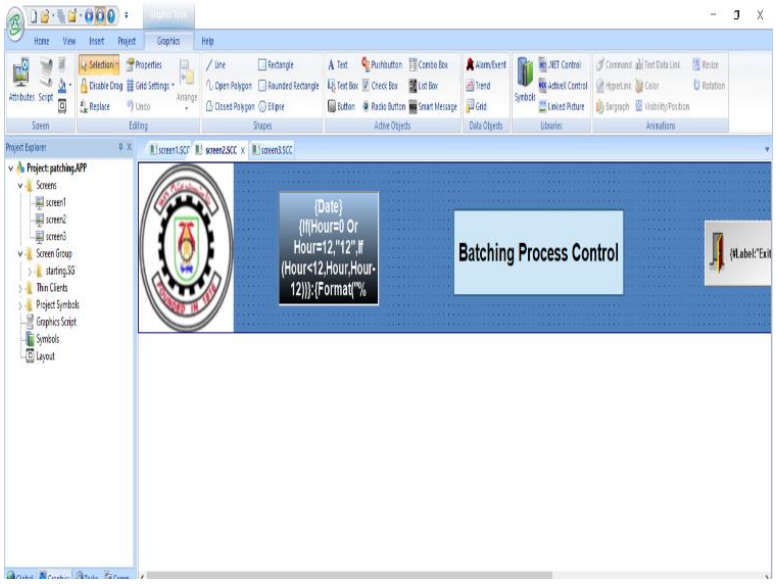
Scada

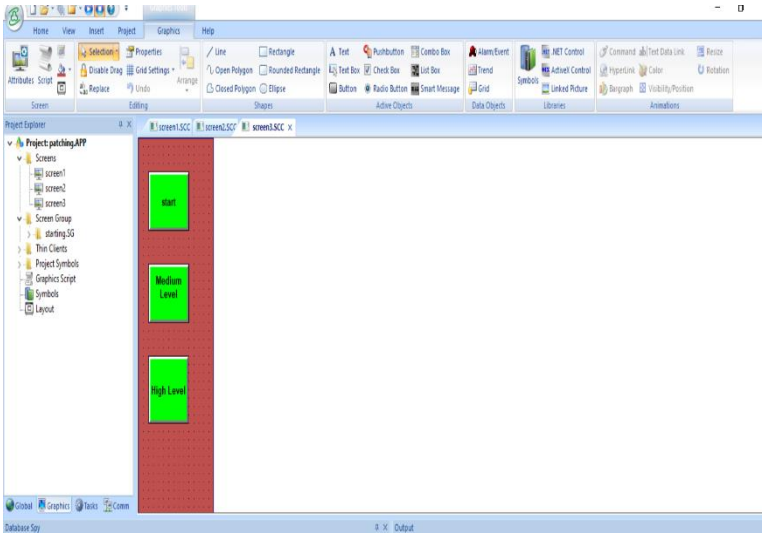
Software used is **IWS v8.0 Wonderware InduSoft Web Studio**

interface screen:

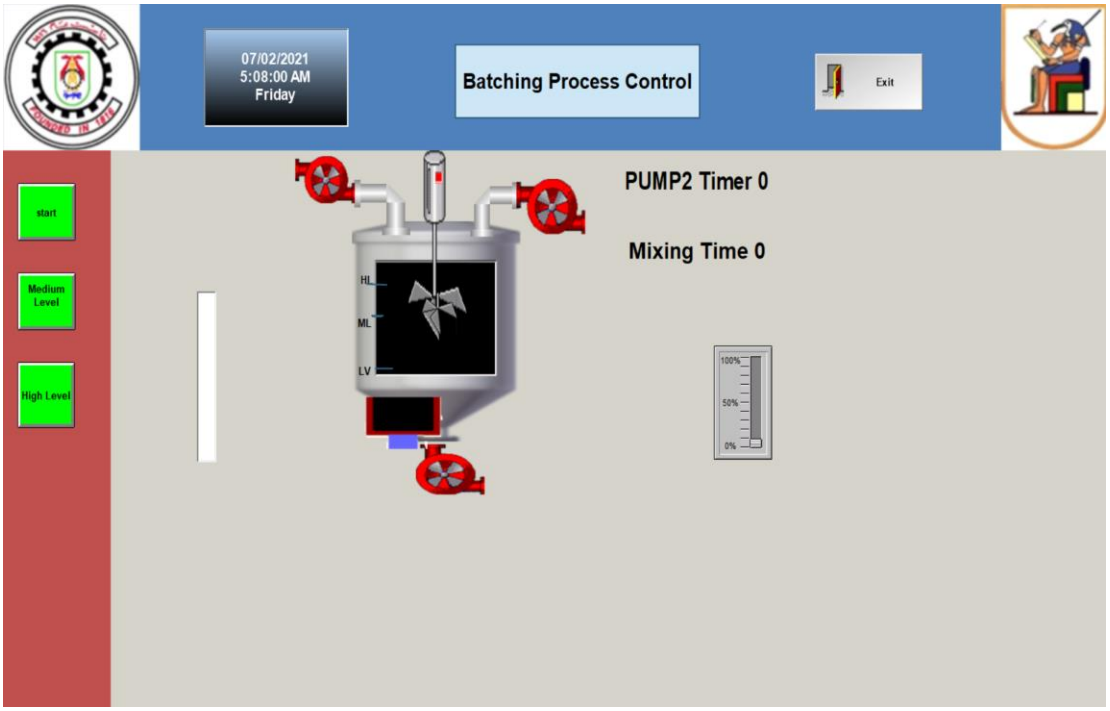
- Add three separate screens
  screen 1: for process interface
  screen 2: for time, data and ….
  screen 3: for control switches of the process







- Then make group screen of the three screens to get all process interface

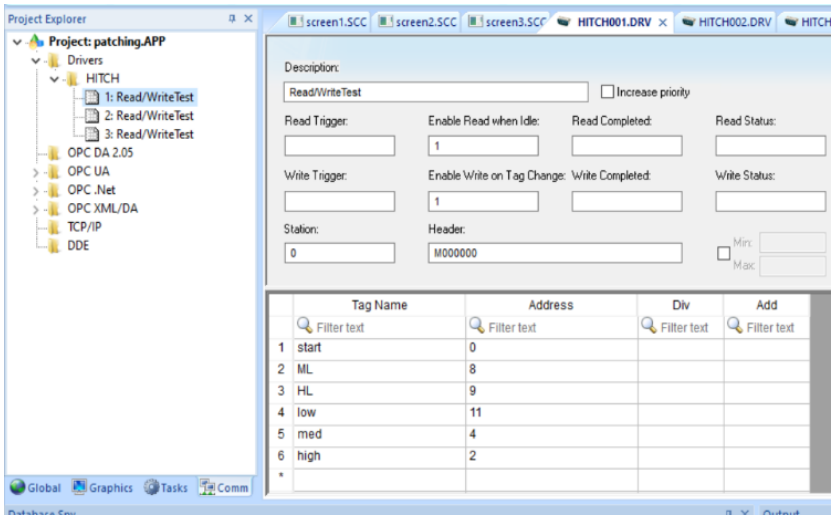


Communication: -

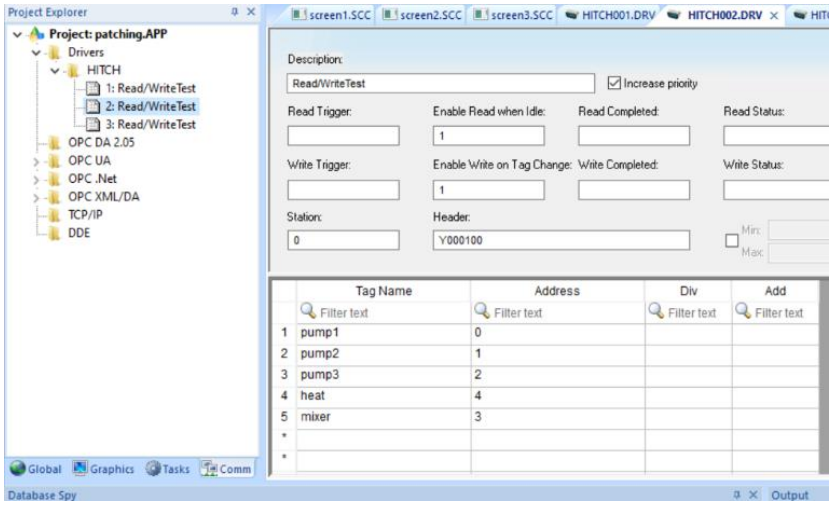
IWS v8.0 Wonderware InduSoft Web Studio many communication drivers which can communicate with different type of PLC

SO, choose HITCH driver and initialize data sheets communication between Scada and plc for inputs, outputs and markers addresses

- Inputs addresses data sheet



- Output addresses data sheet



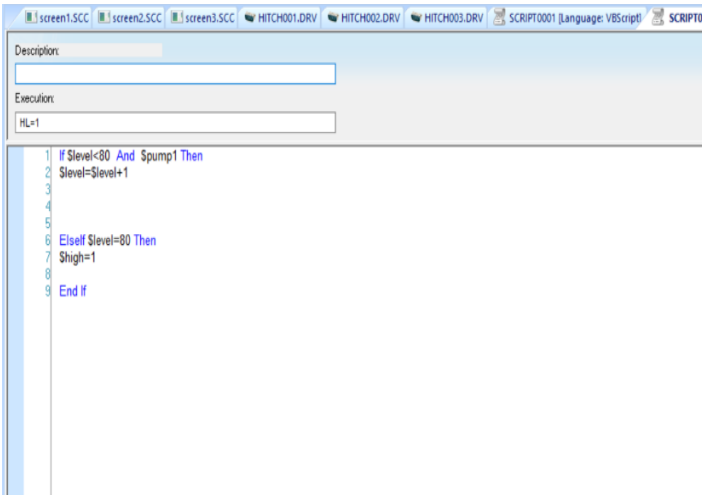
Tasks:

Using some visibility and position animation to represent process through write some tasks scripts

- Filling process at high level mode (Valve 1)

Interface

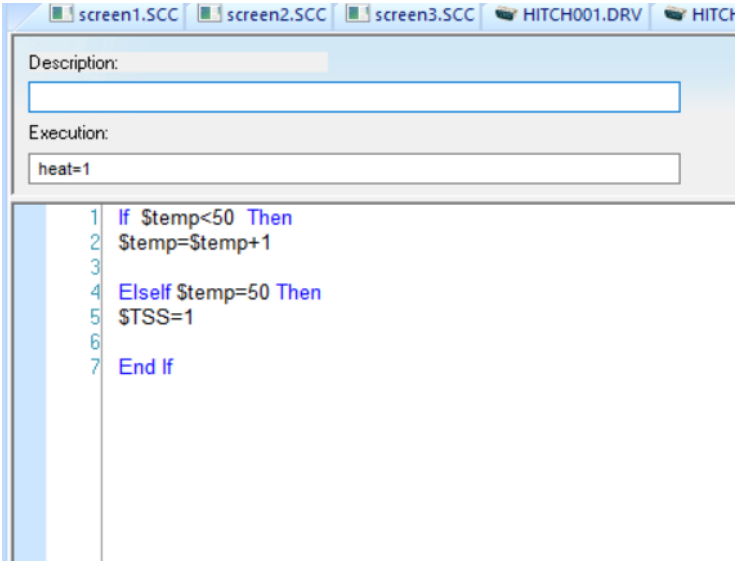
script



- Heating process

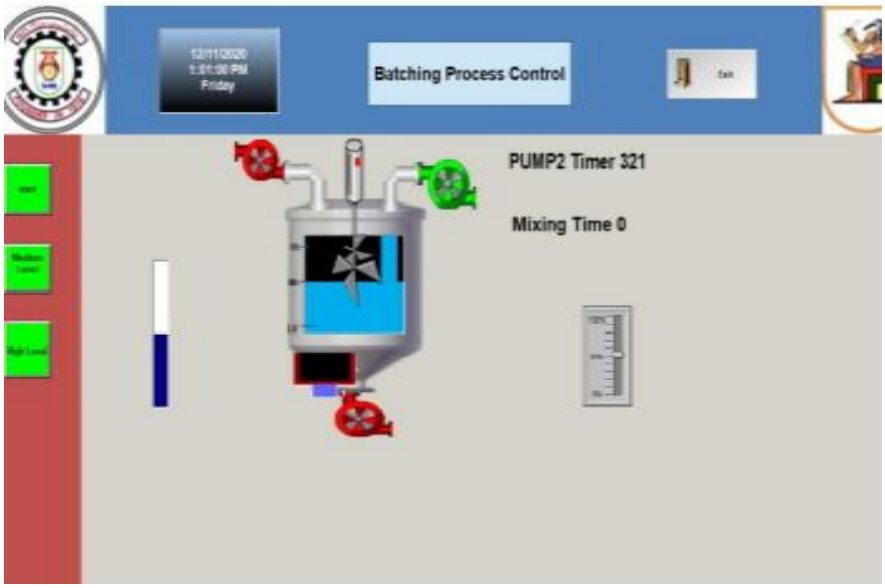
interace

Script

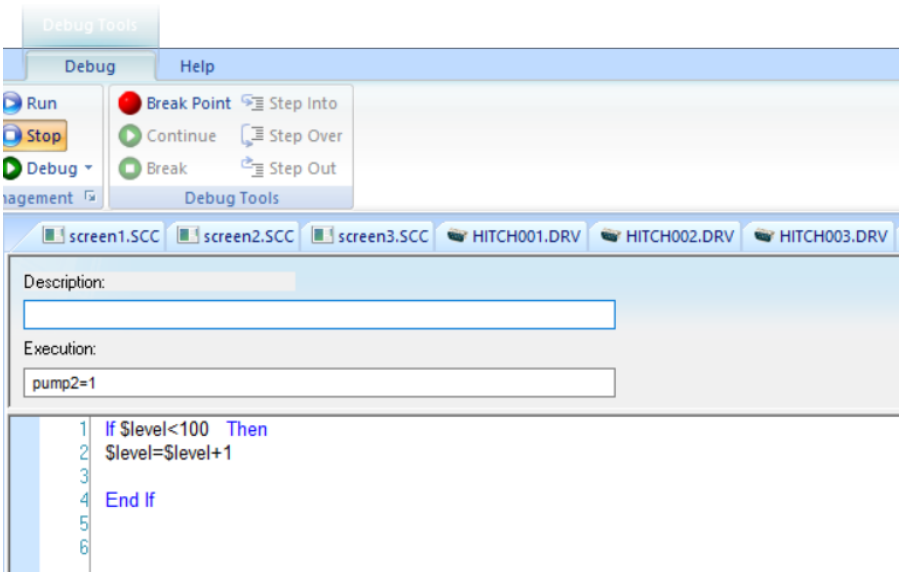


- Filling process of solution (Valve 2)

interace

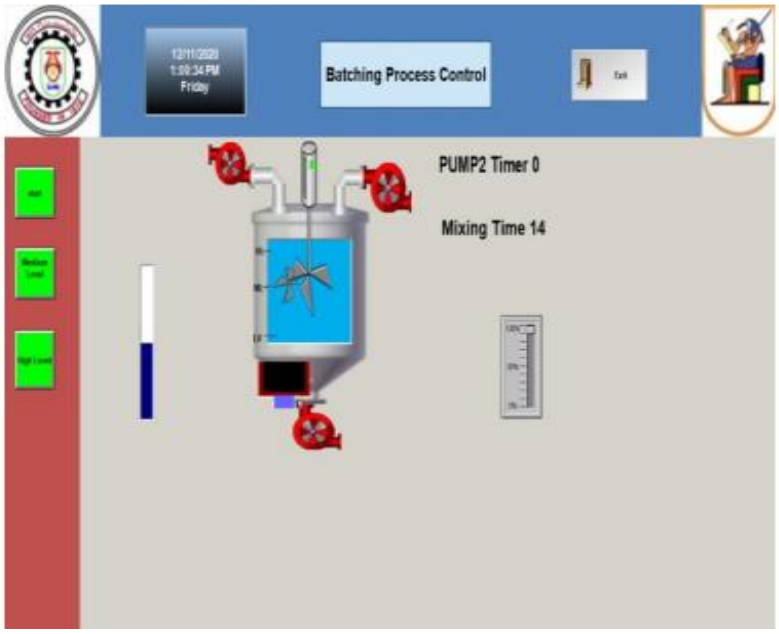


Script

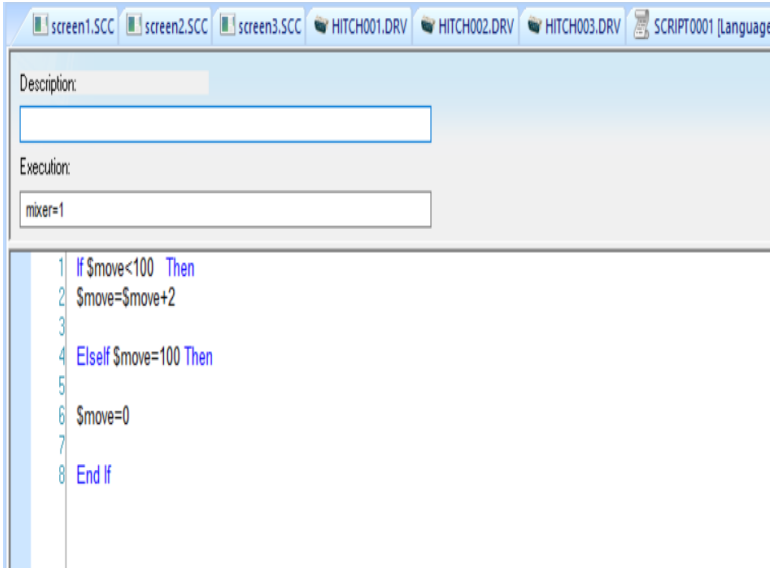


- Mixing process

interace

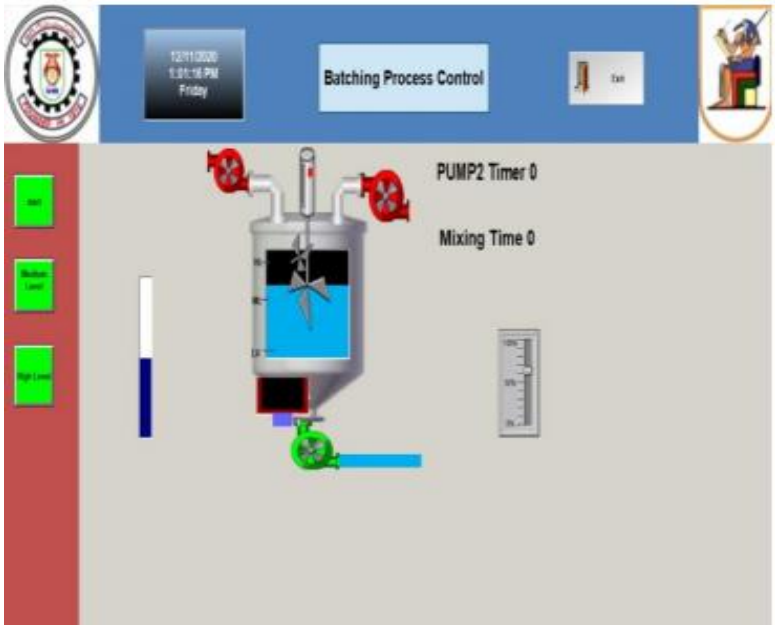


Script

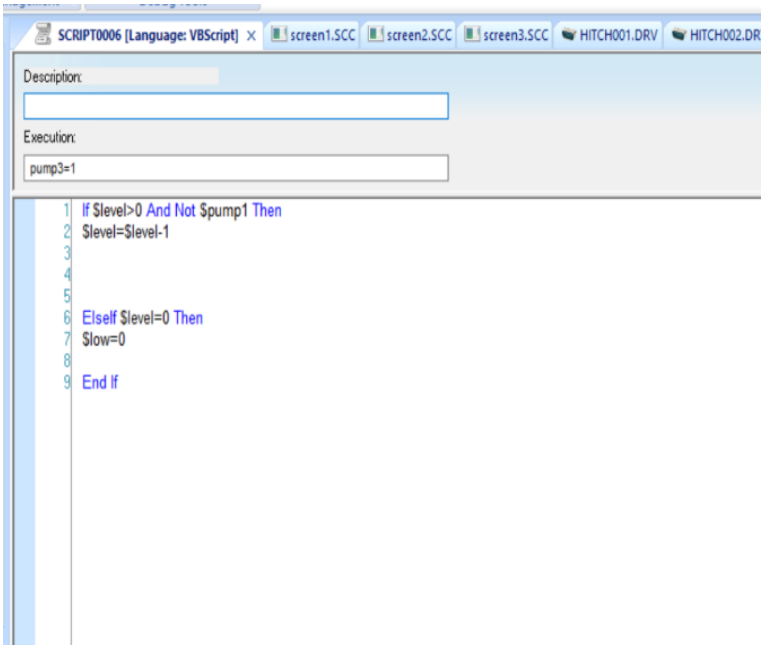


- Emptying process (Valve 3)

interace



Script



Arduino level sensor Code

```
// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP8266_HARDSERIAL_POINT

#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_SERIAL Serial
#define REMOTEXY_SERIAL_SPEED 115200
#define REMOTEXY_WIFI_SSID "RemoteXY"
#define REMOTEXY_WIFI_PASSWORD "123456789"
#define REMOTEXY_SERVER_PORT 6377


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
 { 255,3,0,0,0,64,0,11,161,0,
 1,0,2,25,24,24,8,31,83,116,
 97,114,116,0,1,0,36,27,23,23,
 8,31,77,101,100,0,1,0,70,28,
 23,23,24,31,72,105,103,104,0,129,
 0,24,2,54,8,1,80,97,116,99,
 104,32,67,111,110,116,114,111,108,32,
 0 };
```

```
// this structure defines all the variables and events of your control interface
struct {


   // input variables
  uint8_t Start; // =1 if button pressed, else =0
  uint8_t MED; // =1 if button pressed, else =0
  uint8_t high; // =1 if button pressed, else =0


   // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0


} RemoteXY;
#pragma pack(pop)


/////////////////////////////////////////////
//          END RemoteXY include          //
/////////////////////////////////////////////
#define PIN_START 12
#define PIN_MED 11
#define PIN_HIGH 10
#define trigpin 7
#define echopin
int led1 = A0;
int led2 = A1;
int led3 = A2;
int value=0;
float voltage;
```

```
void setup()
{
  RemoteXY_Init ();


  pinMode (PIN_START, OUTPUT);
  pinMode (PIN_MED, OUTPUT);
  pinMode (PIN_HIGH, OUTPUT);


Serial.begin(9600);
pinMode(trigpin, OUTPUT);
pinMode(echopin, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);


   digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  // TODO you setup code


}
void loop()
{
  RemoteXY_Handler ();
  digitalWrite(PIN_START, (RemoteXY.Start==0)?LOW:HIGH);
  digitalWrite(PIN_MED, (RemoteXY.MED==0)?LOW:HIGH);
  digitalWrite(PIN_HIGH, (RemoteXY.high==0)?LOW:HIGH);
```

```
   int duration, distance;
 digitalWrite(trigpin, HIGH);
delayMicroseconds(1000);
digitalWrite(trigpin, LOW);
duration = pulseIn(echopin,HIGH);
distance = ( duration / 2) / 29.1;
Serial.println("cm:");
Serial.println(distance);
if(   (distance <= 10) &&(distance>0)   )
{
  digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
      digitalWrite(led3, HIGH);

 Serial.println("highlevel");
}

else
if(   (distance <= 16) && (distance>10)&& (distance>0)  )
{

digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);

 digitalWrite(led3, HIGH);
    Serial.println("midlevel");
```

```
} else
if(  (distance > 16) && (distance<=20) && (distance>0)  )
{
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, HIGH);
   Serial.println("lowlevel");
} else
if(distance>23  && (distance>0))
{
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
   Serial.println("nolevel");
  }
  // TODO you loop code
  // use the RemoteXY structure for data transfer
  // do not call delay()

delay(2000);
}
```

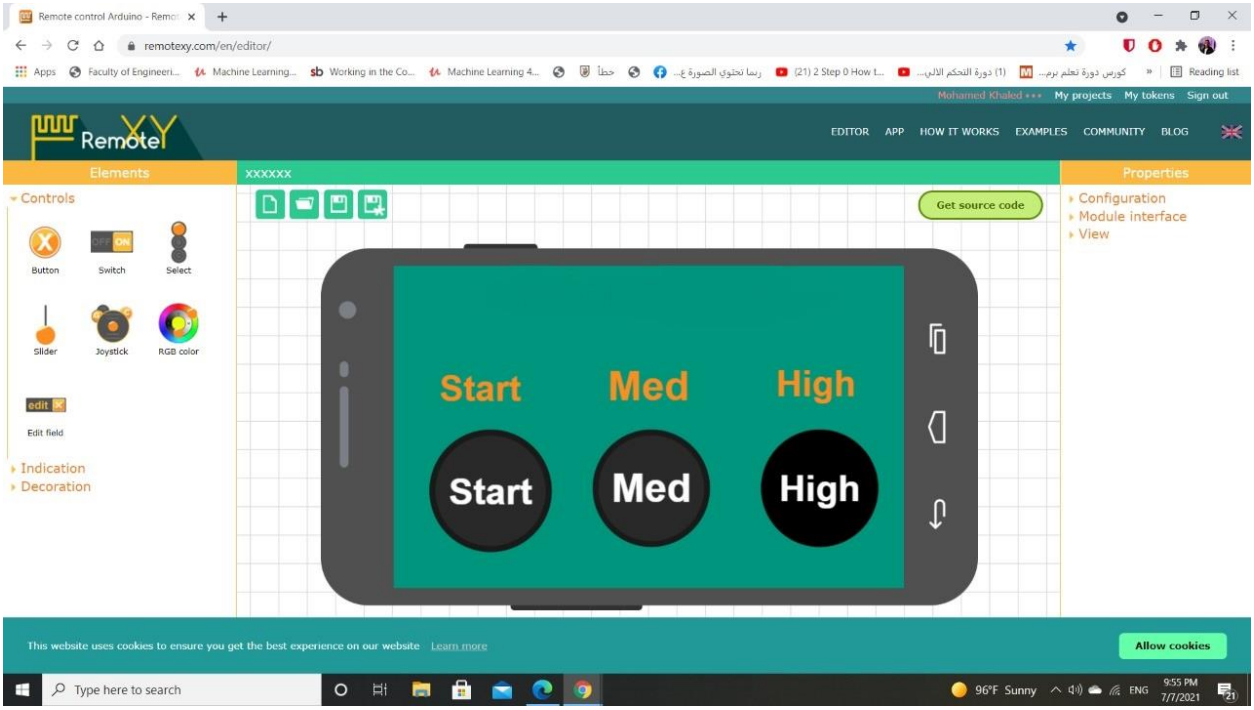
Arduino IOT Code

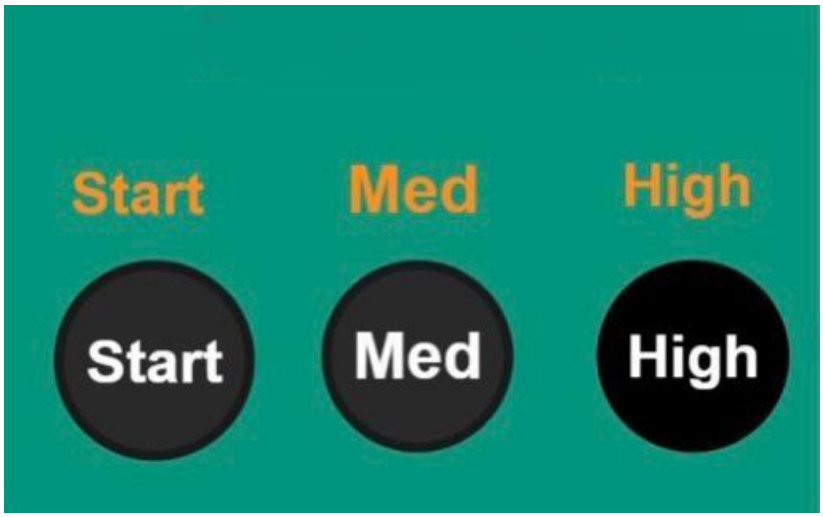
```
//////////////////////////////////////////////
//        RemoteXY include library          //
//////////////////////////////////////////////
// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP8266_HARDSERIAL_POINT
#include <RemoteXY.h>
// RemoteXY connection settings
#define REMOTEXY_SERIAL Serial
#define REMOTEXY_SERIAL_SPEED 115200
#define REMOTEXY_WIFI_SSID "RemoteXY"
#define REMOTEXY_WIFI_PASSWORD "123456789"
#define REMOTEXY_SERVER_PORT 6377
// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
 { 255,3,0,0,0,64,0,11,161,0,
 1,0,2,27,24,24,8,31,83,116,
 97,114,116,0,1,0,36,27,23,23,
 8,31,77,101,100,0,1,0,70,28,
 23,23,24,31,72,105,103,104,0,129,
 0,24,2,54,8,1,80,97,116,99,
 104,32,67,111,110,116,114,111,108,32,
 0 };
// this structure defines all the variables and events of your control interface
```

```
struct {
   // input variables
  uint8_t Start; // =1 if button pressed, else =0
  uint8_t MED; // =1 if button pressed, else =0
  uint8_t high; // =1 if button pressed, else =0
   // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0
} RemoteXY;
#pragma pack(pop)
/////////////////////////////////////////////
//          END RemoteXY include          //
/////////////////////////////////////////////
#define PIN_START 12
#define PIN_MED 11
#define PIN_HIGH 10
void setup()
{
  RemoteXY_Init ();
  pinMode (PIN_START, OUTPUT);
  pinMode (PIN_MED, OUTPUT);
  pinMode (PIN_HIGH, OUTPUT);
  // TODO you setup code
}
void loop()
{
```

```
RemoteXY_Handler ();

digitalWrite(PIN_START, (RemoteXY.Start==0)?LOW:HIGH);

digitalWrite(PIN_MED, (RemoteXY.MED==0)?LOW:HIGH);

digitalWrite(PIN_HIGH, (RemoteXY.high==0)?LOW:HIGH);


// TODO you loop code

// use the RemoteXY structure for data transfer

// do not call delay()
```

Android application interface





Conclusion

The main objects of the project are to design an appropriate model for automatic filling , mixing and heating liquid in a tank, design a SCADA system for the process , design the IOT system, and design a prototype system for the process using the hardware component which are one tank 36 liter which the process take place, 3 solenoid valves 0.5inch for filling and emptying the tank , one dc motor act as mixer, Dc supply 24 volt, ultrasonic sensor to detect the level, temperature controller, 3 PBs, Arduino Uno and plc Hitachi.

As seen the project started with selecting appropriate PLC and studying its configurations like processor, mounting track, input and output modules, power supply and programming unit.

As we saw throughout all the project we faced some problems like: sourcing some equipment like the solenoid valves and the dc motor which acts as the mixer.

We also had a mechanical problem in the fan connected to the motor which we couldn’t at the beginning of the project to attach the pole to the shaft of the motor, then we managed to solve this problem by going to the mechanical workshop of the project.

We also faced a problem in sensing for the levels of the tank at first, we used the electrode method, it doesn’t work because it needs a transducer “very high cost”, then we got an idea to sense for levels using the ultrasonic sensor which is a low cost method compared to that of the electrode and more efficient.

At the end of the project we accomplished a lot of things and we benefited from it to be familiar with PLC, inputs and outputs, sensors actuators and relays.

Also, we implemented simulation process on SCADA, and we made complete system integration.

Also, we studied the PLC input and output devices. And we designed the IOT system and implemented it using Arduino and android application.

References

- Muthukrishnan, V., Shakir, Usama, Saiful, Raul, Yerunkar, P., Ramakrishnan, H., Cc, Sachin, & Ugo, E. (2021, April 11). *Programmable Logic Controllers (PLCs): Basics, Types & Applications*. Electrical4U. https://www.electrical4u.com/programmable-logic-controllers/.
- Muthukrishnan, V. (2021, April 4). *SCADA System: What is it? (Supervisory Control and Data Acquisition)*. Electrical4U. https://www.electrical4u.com/scada-system/.
- https://www.theengineeringknowledge.com/plc-discrete-input-and-output-devices/
- https://www.hitachi.com.au/documents/product/micro_eh_tech_specs.pdf
- Vignesh, & Mike. (2019, August 25). *What is PLC scan and types of Scan Cycles?* Instrumentation Forum. https://instrumentationforum.com/t/what-is-plc-scan-and-types-of-scan-cycles/8592.
- Jost, D. (2019, October 7). *What is an Ultrasonic Sensor?* FierceElectronics. https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor.
- Phillips, R. (2020, April 29). *PLC Actuators and Output Devices (Alarms and Indicators, too)*. PLC Basics. https://basicplc.com/plc-actuators-and-output-devices/
- Posey, B., Rosencrance, L., & Shea, S. (2021, March 24). *industrial internet of things (IIoT)*. IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT
- https://components101.com/wireless/esp8266-pinout-configuration-features-datasheet
- https://www.hitachi-da.com/files/pdfs/service-support/Produktarchiv/SPS/NJI-350EX_MICRO-EH_Manual_EN.pdf
- https://docplayer.net/8282655-Iot-basics-getting-started-with-the-internet-of-things.html
- Mortenson, T. (2020, August 24). *PLC Hardware Explained | PLC Hardware Components | RealPars*. PLC Programming Courses for Beginners | RealPars. https://realpars.com/plc-hardware/
- Phillips, R. (2020b, April 29). *PLC Basic Sensors and Input Devices – A Beginner's Guide*. PLC Basics. https://basicplc.com/plc-basic-sensors-and-input-devices/
- https://www.skyfilabs.com/resources/what-is-arduino-in-iot

- K. (2020, January 21). *What is Industrial Internet of Things (IIoT) and why is it important in manufacturing?* Aptean. https://optiware.com/blog/what-is-industrial-internet-of-things-iiot-and-why-is-it-important-in-manufacturing/
- https://www.farnell.com/datasheets/1682209.pdf
- https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf