Chapter 11 – Industrial control systems

Goals

- To fill in some gaps between classical Control Systems study and the real world as Control Theory is implemented in industry today
- To make the student aware of how the theoretical implementation of PID Control fits into the bigger universe of industrial controls

Introduction

In classical Control Theory, one learns primarily the modelling of linear systems and their control using the PID family of controllers. Thus the study involves Laplace Transforms, system modelling, root locus, frequency response, etc. The study of all these things is essential for understanding the dynamics of industrial systems. The problem is, however, that even after this body of knowledge is mastered, the student is still a long way from being familiar with control and automation systems as they are used in industry today.

Modern industrial plants, vehicles, buildings, etc. are outfitted with smart devices that together form a system that is designed to function as an integrated unit. These devices “talk” to each other and “talk” to a central, coordinating computer too. As these devices become “smarter”, i.e. more capable, they are able to do more, such as recognize when something is wearing out, when a sensor is going bad, etc. As automation technology has matured, these smart devices have been hooked up to a local network in the plant, vehicle, or building or even to the Internet. The connectiveness of all these devices and their increasing “intelligence”, this milieu, has come to be known as the Internet of Things (IoT). In the IoT world, much of the intervention performed by humans can be eliminated, because the smart devices know how to take care of themselves. For instance, performance monitoring software in an airliner flying over Greenland detects a change in the vibration signature of a bearing in one of the airliner’s turbine engines. It knows that within a certain number of flight hours the bearing will need to be serviced, knows the airliner’s schedule, so the monitoring software automatically intervenes, orders parts shipped to a service center where the airplane will be serviced, and then even schedules the repair. All of this preparation for the repair is done automatically, though probably approved by a human. But the smart devices have dealt with the coming fault automatically, with much less human interaction than had occurred in the past.

This chapter attempts to fill the gap between your “book knowledge” of classical Control Theory and what you will find in the real world, by introducing the real world of automation and control systems that one finds on the factory floor, in an industrial facility like a refinery or electric generation station, or in a vehicle like an automobile or an airliner. Linear control theory is, of course, a big part of that. But there are also other parts and pieces of this milieu that people involved in it must know about to be fluent in this industrial environment. And, of course, this is rapidly changing technology, especially now with the full-scale exploitation of artificial intelligence to reduce the necessity of human intervention.

Several technologies are presented here:

1. Sequential control with Programmable Logic Controllers (PLCs)
2. SCADA system (SCADA stands for Supervisory Control and Data Acquisition)
We do not have here the space to discuss any of these in great detail. This chapter is meant simply as a brief introduction and explanation of these technologies. Further reading and research will be needed to truly master these technologies.

PLCs are a sort of alternative to PIDs and their continuous control. Continuous control means that the dynamics of variables is tracked and control actions are taken to hold these values constant or to change them to some desired values. Discrete control is the domain of PLCs. The control here is two-state, i.e. on/off. Not all processes need continuous control. For instance, in a yoghurt packaging plant, the yoghurt cup moves from one point to another, where the yoghurt is put in the cup, then, after it is full, the cup moves to a station where a top is put on it. After that, the yoghurt moves to a packaging stage, where 6 or 12 or 24 yoghurt cups are aligned and put into a box. The boxes may then be assembled in turn, palletized, and put into refrigerated storage. All those steps, all that moving, needs to happen in a coordinated fashion, sequenced, timed. Sensors are placed to confirm the completion of each step before the next one starts. But all of that can happen without the involvement of a single PID controller. “Can” is emphasized, because it could also be that a belt for moving the yogurt has a PID speed controller on it or that some type of flow-rate controller is used to squirt the yoghurt into the cups.

The four technologies mentioned above are not mutually exclusive but rather are tied together rather intimately. The last three involve networked systems, in that they usually include many different PLCs, PID controllers, sensors, and actuators. Networked systems organize the controllers in the network and provide the user interface for a plant operator, building supervisor, or driver or pilot to monitor the system and keep track of the product flow through a plant or the operating state of a vehicle. They also manage fault-detection and safety logic to prevent unscheduled shutdowns, breakdowns, or even more dire consequences when something goes wrong. In vehicles, driver-assistant systems stand ready to jump in and prevent an accident when it is determined that the driver or pilot is in a dangerous situation that he or she may not be able to get out of. Vehicle technology is advancing very rapidly, so that we can see on a pretty close time horizon driverless cars and pilotless aircraft. After all, the cause of most vehicular accidents is human error. So this technology promises to bring a dramatic decrease in accidents as it matures and is deployed.

Sequential control and PLCs

See the Wikipedia article on programmable logic controllers.

Sequential control is on/off control. It is so named because it orchestrates sequences of events. That is, it makes sure that a number of steps occur in the proper sequence and with the proper timing. This type of sequencing is common in many manufacturing environments. PLCs were developed in the 1960s and 70s to replace hard-wired relays and sequencers. This was still in the early days of computer development. PLCs were just specialized industrial computers. One feature they had in the early days is that there were ruggedized for the industrial environment. One other feature is that, like other computer processors, the tendency is for them to operate ever faster.

The native language of PLCs is ladder logic. This is a graphical method of laying out switching logic that runs in the PLC. Ladder logic evolved from hardwired relay logic diagrams that were used for sequential
control prior to the advent of PLCs. Figure 11.1 shows a typical ladder logic program running in a PLC. It’s obvious why this programming technique has its name. The ladder rail on the left represents an electrical power supply, while the right rail represents ground. The PLC governs the flow of electricity from the hot rail to the ground rail. The rungs of the ladder show the switches and relays through which the power flows.

![Ladder logic program](image)

The program in a PLC runs in a loop. We start with rung 0000 and run down to rung 0004 in this case. The program operates by considering each rung, performing its operation, then proceeding down to the next rung. Once the bottom rung is reached, the program starts over at the top rung once again. Once cycle through the rungs is called a scan. The time it takes to perform a single pass through a program is called the scan rate. This is typically on the order of milliseconds or microseconds, so that the PLC can manage fast-moving actuators in a plant.

Now consider each rung. On the left, the symbols that look like brackets are the switches for each rung. On the right, the symbols that look more like parentheses are relays for machinery in a plant. The switches control whether or not the relay receives power and thus whether a particular machine turns on (or off, as the case may be). For each rung, then, the left-hand, bracket symbols are designated “I” for input. The right-hand, parenthesis symbols are designated “O” for output. The open brackets (“–]”) are normally open switches. That is, without being pushed, this switch is open. The closed brackets (“[/”) are normally closed switches. When these switches are actuated, they open. The top two rungs (0000 and 0001) are very simple rungs. The relays on the right of these two rungs close only when the switches on the left are turned on.

**Example 11.1 – Motor-start circuit**

We want to program a PLC to handle the starting of a motor in a plant by actuating its relay. The motor is normally off, and we want to turn it on. A simple ladder rung shows the basic logic for doing this.
The “start” shown here refers to a normally open switch. One pushes it, this actuates the relay for the motor, the motor starts. The problem with this, however, is that since the start switch is normally open, one would have to stay there on this switch, holding it shut. To get out of doing this, we need to latch the switch. That means that once the motor starts, we want it to stay on. This is indicated in ladder logic as shown below.

There are now two parallel paths for actuating the motor relay. If the start button is pushed, the motor relay is actuated. But the motor being on actually closes this second switch, labelled “motor” above. So if the start switch is then released, it opens, but the motor keeps running because of the latch indicated by the bottom parallel path.

Now what is missing is a way to stop the motor. We add a normally closed stop switch as shown in the ladder diagram below.

The stop switch, normally closed, is in series with the start switch and the latch. If the motor is up and running in the latch, one can push the stop switch, which interrupts the flow of electricity to the relay, so the motor shuts off. Note that this also opens the latch. When the stop switch is no longer actuated, it closes. But the motor does not start again. Both paths (start and motor) on the rung have been opened. One must actuate the start switch again to put the motor into operation.

Example 11.2 – Discrete control of tank level

Consider the following situation:

A tank is placed in a process plant to dispose of liquid accumulated in a process. The tank is emptied by a pump. The in-flow into the tank is controlled elsewhere. Unlike the steady tank level controller
discussed in Chapter 3, here the tank level fluctuates between two extreme levels. You do not want the tank to overflow nor do you want the pump to run when the tank is dry. The tank is equipped with two level sensors—high and low. The control logic is:

- When the tank level drops below the low-level sensor the discharge pump is turned off and the red, low-level indicator light is turned on. The pump is latched off.
- When the tank level rises above the low-level sensor, the red, low-level indicator light is turned off.
- When the tank level rises above the high-level sensor, it is turned on. The pump turns on and the yellow, high-level indicator light is turned on. The pump is latched on.
- When the tank level falls below the high-level sensor, the yellow, high-level indicator light turns off.

Let’s start the ladder-logic diagram with the three indicator lights. There is no latching here. When the level is above the high-level switch, the yellow light is illuminated. When the level is below the low-level switch, it the red light is illuminated. When the motor is on, the green light is illuminated. Thus the ladder logic for the indicator lights is

The sensors are “on” when the water level is above them. This switches the states shown in this ladder-logic diagram. With the tank full and the sensors on, the yellow, high-level indicator is illuminated. The red, low-level indicator is off. With the tank empty and the sensors off (in the state shown in the diagram), the yellow, high-level indicator is off, and the red, low-level indicator is on.

Now let’s add the latching logic and the on/off states of the pump. Above high-level, the pump turns on and latches on. Below low-level, the pump turns off and latches off.

In the top rung, when the high-level sensor is reached, the switch closes. The pump stop is a normally closed switch, so the pump relay closes, the pump starts, and this latches it on. At this point, with the high-level exceeded, the low-level switch is on, and therefore open. So the second rung is not actuated. The pump continues to operate, draining the liquid from the tank. When the tank level falls below the low-level switch, it closes, and this actuates the pump-stop switch. It is normally closed, so the actuation opens it. The pump relay deactivates and the pump turns off. The water level rises, and when it exceeds the low-level, the low-level switch opens, deactivating the pump-stop switch. This closes this
switch, since it is normally closed. But the pump does not turn on because 1) the high-level switch is still open and 2) the pump (its latch) is still off. When the tank level exceeds the high-level, the cycle starts over again.

The entire ladder diagram for this system and its indicator lights is just the combination of these two ladders.

From this example it can be seen that the logic, especially the latching logic, can be quite tricky. The various scenarios of operation are called the system states. Modern PLCs come with software that allows a user to program the ladder diagram on a personal computer and then simulate its operation, prior to loading it into the PLC for operation in the plant.

The above diagram does enable the control logic laid out for the system. But it does not include some things that are desirable. The system needs an emergency stop button. This button should not operate directly through the PLC. It should be wired directly to the pump relay. That opens that relay directly without depending on the proper operation of the PLC. But the pressing of that E-stop (as they are called) should also be detected by the PLC, so that it can take action for the system based on the fact that the E-stop was pushed. All operations should be unlatched, so that the system can be put back into operation in a safe manner.

Also missing is the start button for the system and the logic that accompanies it. The system should start up in some logical manner. Also missing is that most automated systems also have a manual mode. There is no provision for MAN/AUTO logic. Often with automatic systems, they are first started in manual mode, brought up to some state of operation, then popped into automatic mode. Here we’ve considered the operation of the system in automatic mode but not in manual mode.

Besides the basic switches and relays described up to this point, ladder logic also contains other blocks that perform tasks useful in manufacturing settings. One is a counter. Figure 11.2 shows a count-up/count-down counter.
Chapter 11 – Industrial control systems

Figure 11.2 - Up/Down counter block

This block counts the number of times a switch closes. Actually there are two switches, one for counting up and one for counting down. The switches may be proximity detectors. If the count-up switch is closed by an item on a conveyor belt, for example, the accumulator will be incremented by one. If, instead, the count-down detector detects the presence of some passing object, the accumulator is decremented by one.

As shown in the figure, the counter block has set in it some limit that is to be reached. In the figure shown, a pre-programmed limit of 24 is shown. The Accumulator is where the count is incremented or decremented. In the example, 16 things have been counted up to the state shown. When the limit is reached, the (done) relay is turned on and stays on so long as the Accumulator is equal to or greater than the Limit. The (up) and (down) outputs merely blink on when the up or down switches are closed. These could be connected to LEDs lighting up or some indication on a monitoring screen lighting up. In this example, perhaps it is necessary for a certain number of pieces to accumulate before another operation is performed. In this case, the (done) output can become an input switch for a rung below this in the ladder diagram. Until (done) is on, that next step will not start. The reset switch is used to set the counter, the Accumulator, back to 0.

A possible scenario for such a block would be in a palletizing operation for, say, bottles of water. If cases of 24 bottles of water are palletized, 24 bottles have to pass a certain point before the case is complete. If downstream of the up-counter signal there is an inspection that can reject bottles with flaws, then each time a bottle is rejected, it passes by a detector that triggers the count-down switch. Once the 24 acceptable bottles have been gathered, the palletizer lift can move down to accept another case on top of the completed one.

Another useful block is a timer. This is shown in Figure 11.3.

Figure 11.3 – Timer block

This timer can be set to work in three different modes. The Limit is the number of seconds to count up to. The Accumulator is the counter for the seconds as they go by; i.e., it is the clock.

The three modes of the timer are:

11-7
1. **Timer-on-delay** – The timer in this mode starts counting when the \[\text{timer}\] switch is turned on. It continues timing, accumulating seconds, so long as this switch is closed. When this switch is opened, the Accumulator is reset to 0. When the Accumulator reaches the Limit, the \{\text{done}\} relay is turned on. While the timer is timing, the \{\text{timing}\} relay on. This could just lead to an indicator that shows whether the timer is counting or not.

2. **Timer off relay** – The timer in this mode starts counting when the \[\text{timer}\] switch goes from on to off. It works exactly opposite the timer-on-delay.

3. **Retentive-on-delay** – The timer in this mode counts like the timer-on-delay. The only difference is that when the \[\text{timer}\] switch is turned off, the Accumulator retains its value. When the \[\text{timer}\] switch is turned back on again, the counting of seconds continues where it left off. To reset the Accumulator to 0, the \[\text{reset}\] switch must be turned on.

There are many other useful function blocks supplied with a PLC. To work effectively with a PLC, you need to be familiar with this function set. The documentation for the PLC will provide the particulars for each function block.

PLCs were originally just sequential-control devices. But as the technology has matured, they are able to take analog inputs and deliver analog results. Thus PLCs may need to perform calculations, and for this, it is necessary to have functions such as arithmetic and trigonometric functions. Again, you need to be choose the PLC with capabilities that are adequate for the application for which it is being used.

**Other, besides ladder logic – C, Python, etc.**

The PLC manufacturer with the largest market share in the United States historically is Allen Bradley which is part of Rockwell Automation. Other big players in the U.S. and abroad are General Electric, with its Fanuc line of PLCs; Siemens; ABB; Modicon, who developed the first PLC, now a part of Schneider Electric; etc. The cost of PLCs has dropped dramatically since their introduction 50 years ago. Also like other computers, their capabilities have increased. PLC architecture has become more modular too. Separate units can be purchased for the PLC itself, the power supply, the I/O (input/output) modules, etc. These are then ganged together to suit the automation job at hand.

A good way to get your feet wet with PLCs is to purchase a low-cost PLC and simply begin tinkering with it. A good line of inexpensive PLCs are those available from a company called AutomationDirect ([www.automationdirect.com](http://www.automationdirect.com)).

**Machine State Diagram**

**SCADA systems**

Please see the Wikipedia article on SCADA systems. Unlike the narrow perspective of a PLC, where an individual machine is controlled, SCADA has a more macro-viewpoint, where an entire plant is under the control of the SCADA system. This is indicative by the name; SCADA stands for Supervisory Control and Data Acquisition. So a SCADA system stands above the nitty-gritty control of individual machines and looks at the plant as a whole.

This viewpoint is not strictly fixed. There are some “machines” that are so big and complicated that they are almost an entire plant in themselves. A good example of this is a paper machine. Raw pulp comes in at one end of the machine, finished paper out of the other end. These machines can be very long and
the intermediate processes numerous and quite complex. Thus you might find a SCADA system for this type of complicated machine.

A SCADA system has four levels of control; that is, it is hierarchical. Figure 11.4 shows this SCADA layout.

![SCADA layout](image)

**Figure 11.4 – SCADA layout**

The layers of automation and control proceed from the lowest level, the plant itself, upward, each layer more supervisory than the one below it. The actual detailed control is performed by PLCs and PID controllers on level 1. The function intimately with the hardware processes that they control without any intervention. Level 1 also includes all the sensors, detecting units, and actuators that are needed to detect the plant condition and change it as needed. The PLCs and PIDs at this level are supervised by computers on level 2. This is where the SCADA software resides. It monitors the functioning of the automation hardware on the level below it. In Figure 11.4 two supervisory computers are shown, one to supervise the PLCs in the plant, another to supervise the PIDs. But there may be more computers on this level, and they do not need to be organized like this. For instance, in a steam power plant, the level-2 computers might be organized by the equipment they control. There might be a boiler-control computer, a turbine-control computer, etc.

From the SCADA level (level 2), an operator can change set points for the PIDs, make decisions to override programmed operation of the PLCs and PIDs, switch the controllers from AUTO to MAN. From level 2 the human/machine interface (HMI) is run. Faults are detected here. Also alarm and emergency plant conditions are detected and presented to the plant operator for response. The dynamic operating conditions of the plant are saved for further analysis to assess plant performance and to improve it for future operation.
Level 3 is where a plant-wide, overseeing computer resides. It is not directly involved in plant control but monitors the plant operation as a whole.

From this diagram it is also clear that a SCADA system may be used to monitor more than one plant. A power company may have more than one generating station. It may have different types of generating units—steam, gas, hydro units, for example. Level 4 monitors the whole system from above. Power production levels for each unit can be organized and dispatched from this level.

Often SCADA systems are used for remote monitoring of control systems. I remember first hearing about SCADA systems when I was a professor at a small university in Maine. Someone told me that all the hydro dams on the Penobscot River were monitored remotely by a SCADA system. An operator could monitor the output of the dams from his house on a computer screen. Also, if anything went wrong in a plant, the SCADA system would send a message to his pager, notifying him of the alarm state. That seemed like magic in the early 1990s.

There are many manufacturers of SCADA systems. Historic leaders in the field are Siemens, Honeywell, WonderWare, Iconics, Intellution, Rockwell Software, GE/Fanuc.

**SCADA system components**

The transmission of operational data between level 1 and level 2 involves the collection of sensor data and then its transmission over a network. Sensor data is collected by a remote telemetry unit (RTU) and organized into a protocol for transmission to the supervisory computer or SCADA station. Likewise command data is transmitted over this network in this protocol, which is kind of language for data communication between levels 1 and 2. In early SCADA systems the data protocol was proprietary, particular to the manufacturer of the SCADA system. But as SCADA systems have evolved, standards have been developed for the data protocol, and SCADA systems are built to these open standards.

The protocol must include tags to identify what data comes from what sensor. Data transmitted as part of a SCADA system can be either digital or analog data. Digital data is bipolar; that is, it is either off or on. This tells whether a particular machine is on or off, a particular valve is open or shut. Some equipment runs at just one level, so it is appropriate to characterize its operation as an on/off state, an open/closed state. Analog data, on the other hand, varies continuously over a range. For example, a throttling valve, as opposed to an open/closed valve, can be open at any setting between 0% and 100%. It is used to modulate flow, that is to maintain it at a desired level. Thus the valve-opening setting is a piece of analog data that would be transmitted over a SCADA network to remotely monitor a plant. Other such analog data would include pressures, flow rates, temperatures, tank levels, etc. Likewise, data transmitted from the supervisory computer to the controllers and PLCs in the plant can be either digital or analog. Digital data—on/off data—could be used to start or stop a motor, for example, from a remote SCADA station. Analog data sent from a remote station could include, for example, a temperature set-point for a PID controller. It might also include the gain settings for a controller. Thus an operator can monitor a plant remotely and even operate it remotely using a SCADA station and transmitting data back and forth over a network running a particular SCADA data protocol.

Another important part of a SCADA system is notification of alarms. If part of the process exceeds specified limits, an alarm will be sent from level 1 to level 2. Generally a process will run safely within pre-specified limits. For example a tank will operate between a pre-specified low and high pressure.
Chapter 11 – Industrial control systems

the tank exceeds the high value or drops below the low value, an alarm is triggered. Generally there are two stages of alarms: low and low-low at the lower end of the range and high and high-high at the top end of the range. And in general the first-level alarm (low or high) triggers a yellow indicator light. The second level alarms (low-low and high-high) trigger a red light and also an audible alarm. Second-level alarms indicate that an emergency situation exists and that impending damage or an explosion will occur if immediate action is not taken. The SCADA system records and transmits alarm states. It also records actions taken in response to alarms. In the event of an accident, the SCADA systems serves as a record of what went wrong, the order in which things happened, and the actions taken in response to an alarm state in the plant.

Besides the collection and transmission of data and commands and the alarm transmission, a SCADA system also includes a back-up power supply in case electric power is lost to the system.

Human Machine Interface (HMI)

The HMI is an integral part of a SCADA system. It has replaced the old, hard-wired, steam-gauge interface with computer monitoring and control. Figure 11.5 shows a comparison of these two technologies, separated in time by about a half century.

![Figure 11.5 - Hard-wired, vintage control room vs. modern SCADA control room](image)

The modern SCADA system allows for much more effective monitoring and control. Screens can be switched to show an overview of the plant or to focus on a particular area of operation. If some part of the plant is in an alarm state, that part of the plant can be automatically displayed on a screen and demand action from the operator. Also, most modern SCADA systems have an editor that can be used to modify the display on a screen or make new screens.

What appears on the screens is usually a schematic of a process in the plant, showing the logical layout of that part of the process. This is linked to the data acquisition part of the SCADA system, so that the state variables of the plant are shown in the diagram alongside the equipment to which they apply. Figure 11.6 shows a generator control screen that is part of a SCADA system.
Distributed Control Systems

Please see the Wikipedia article on Distributed Control Systems. DCSs are very similar to SCADA systems nowadays. But where SCADA systems in the past were used often for remote monitoring and control, DCSs were always located in a control room at the process facility they are used to control. Also, historically, SCADA systems were used more for discrete control, i.e. control primarily with PLCs, while DCS systems ran control loops with PID controllers. And where SCADA systems were used primarily for monitoring and the discrete operation of equipment, such as turning pumps or other equipment on or off, DCSs actually had the control loops centralized in the main operating room of the plant. As control equipment evolved, however, the field devices became smarter, and it was possible and more secure to have a valve or a tank with its own local processor to run the loop locally. Thus the DCS came to resemble a SCADA system. Likewise, from the other direction, PLCs got to be more powerful so that they too could run PID logic. Thus now the differences between DCS and SCADA systems is not so great. The Wikipedia article on DCSs charts their history and evolution since their inception in the 1960s.

The big DCS manufacturers over the years have been Bailey Controls, Fisher Controls, ABB, Yokogawa, Honeywell, and others.
Automation data protocols

**CAN bus**

Besides control and monitoring systems for manufacturing and process facilities, there are also systems for vehicles. Automobiles have evolved from mostly mechanical devices with electronics playing a minor role into what many characterize today as “a computer on wheels”. Indeed, today’s automobile contains 30 to 100 microprocessors, all linked together to coordinate the operation of the vehicle. This computerization of the automobile has given us anti-lock braking systems, adaptive cruise control, anti-skid stability systems, etc. And we are poised now to see the introduction of the driverless vehicle.

CAN stands for Controller Area Network. It was introduced by Bosch in 1986 as a protocol to allow Electronic Control Units (ECUs) in vehicles to communicate of a bus architecture. ECUs are self-contained control units in a vehicle dedicated to a particular task. Typical ECUs in a vehicle include the Engine Control Module, Transmission Control Module, Brake Control Module, Suspension Control Module, etc. For example a light-control unit turns on the headlights and taillights, the turn signals, the fog lights, the brake lights when the brakes are applied, etc.

![Engine Control Unit](image)

**Figure 11.7 - Engine Control Unit, one of the principal Electronic Control Units in a vehicle**

Figure 11.7 shows an engine control ECU. Pre-CAN ECUs had communicated via point-to-point wiring. This was becoming increasingly complex as the number of ECUs in a vehicle increased. Thus Bosch developed the CAN bus to simplify the wiring in a vehicle and to allow the different ECUs to communicate with each other. You are probably unwittingly familiar with a CAN bus. If something goes wrong with your vehicle, the “check engine” light comes on. You take your car to a shop, and there they plug in a diagnostic tool that reads the fault codes to identify the problem. The mechanic is then accessing the CAN bus and the principal vehicle ECU to read off the faults.
Figure 11.8 - CAN-bus architecture in a vehicle

Figure 11.8 shows a CAN bus and the ECUs connected to it. It also shows the D-pin connector used to access the CAN for monitoring and diagnostic purposes. Notice that the bus consists of two wires, the CAN Hi wire and the CAN Lo wire. The CAN bus passes data from one ECU to another using a stream of signals. Thus it is a serial bus in that data is passed as a sequence of bits. The bits are identified as ones or zeros depending on the voltages of the sequence. Figure 11.9 shows this.

Figure 11.9 - Data frame on a CAN bus

The CAN Hi and Lo wires are biased at 2.5 volts; that is, in a dormant state, each wire carries 2.5 volts. If no voltage change is induced in the wires, the two 2.5-volt signals are interpreted as a 1. If the CAN Hi wire is driven toward 5 volts and the CAN Lo wire simultaneously toward 0, this is interpreted as a 0. What the 1s and 0s means constitutes a data frame of the CAN protocol. Figure 11.9 also shows the different fields and their meanings in different colors.

Note that only a small part of the frame actually contains data. The rest of the frame is for administration and data integrity checking. The arbitration field contains a identifier that tells what the data represents. This could be rpm, coolant temperature, oil pressure, etc. This field also contains a priority, that is how important the message is. An alarm has more priority than just an informative message that gives the value of a certain physical quantity. The control field tells how many bits the data that follows uses. Reporting whether or not the brake pedal is pressed takes only one bit, whereas reporting a physical quantity’s value may require all 8 bits. The CRC field is used for checking the integrity of the data in the frame, which can be corrupted in an electronically noisy environment like that of a vehicle. And seven bits with 1s in them are used to mark the end of the frame.

The CAN data frame in Figure 11.9 is Bosch’s CAN 2.0A data frame. This is the 11-bit data frame because there are 11 bits in the Arbitration Field. When this was released in 1991, Bosch also released CAN 2.0B,
which has 29 bits in the Arbitration Field. In 2012 Bosch released CAN FD, where the FD stands for “flexible data rate”. Though the CAN bus was originally developed for automotive uses, it has proven itself to be useful also in applications ranging from aircraft and ships to robotics and use in automated medical devices. Also, though Bosch has been the principal developer of CAN technology, this has been formalized in the ISO 11898 series of standards.

Where does PID control fit into a CAN-bus system? The ECUs that are linked in a CAN bus contain internal algorithms that implement PID control for various loops in the vehicle. For example the cruise-control loop will be embedded in the Body Control Module, which contains the control for the vehicle as a whole. Though it may seem on the surface that CAN-bus systems have nothing to do with PID loops, they are there, at a level closer to the automotive hardware, keeping the engine running right and helping the driver drive the car down the road.

Conclusions

As can be seen from this chapter, PID control and continuous control, as covered in chapters 1-10 is a very important part of control, but it is only a part of the industrial automation and control environment. In going into Automation and Control as an industrial career, one needs to know the technical context of the industry into which he or she is going. Is it primarily continuous and PID control? Is it instead primarily sequential control with PLCs? Is a SCADA system or a DCS involved? These are really the first questions that need to be answered.

Then one needs to be aware that different automation and control environments have their own specialized language. You need to know the vocabulary of that language to be effective as a controls practitioner in a specific automation and control position.

The continuous, classical Control Theory covered in Chapters 1-10 is perhaps the most challenging of the technologies involved in automation and control. You may find that you know more about this than other engineers and technicians with whom you are working. This has been my personal experience. Many instrumentation and controls engineers and technicians in plants know very little of the theory behind the everyday practice that they are involved in. Often they learn how a plant behaves by working with it, tuning it by making small changes and then observing the effect. But the big picture, the control laws and the dynamics of a plant remains outside their area of expertise. That can severely handicap their ability to understand the plant, recognize why it behaves as it does, and make bigger tuning changes that improve plant operation.

Links

Wikipedia article on programmable logic controllers: https://en.wikipedia.org/wiki/Programmable_logic_controller

YouTube introduction to ladder logic: https://www.youtube.com/watch?v=Ei4_HqzUFBs

Wikipedia article on SCADA systems: https://en.wikipedia.org/wiki/SCADA

Introduction to SCADA systems: https://www.youtube.com/watch?v=5ZiIA-kMV8M

Chapter 11 – Industrial control systems

Wikipedia article on automation protocols:

Problems

11.1 Using the PLC Ladder Simulator for Android, set up a simple ladder diagram to turn on and turn off four outputs, using four input switches to do this. Get rid of the other output coils so that only four outputs are displayed. Label the outputs One, Two, Three, and Four.

11.2 Use again the PLC Ladder Simulator: Take one of the rungs above and make a ladder that turns an output coil on, indicated by the yellow lights in the Simulator mode. Set this up with the input as a Push-button instead of a switch. Add the latching necessary to have the light stay on until a stop button (not a switch) is pushed to turn it off. Make sure this works by turning it on and off several times.

11.3 Use again the PLC Ladder Simulator: Use two NO inputs, I0.0 and I0.1 as switches, not push-buttons, to turn on four different output coils: O0.0-O0.3. O0.0 should turn on when I0.0 is on, O0.1 should turn on when I0.1 is on. O0.2 should turn on when either of the inputs is on. O0.3 should turn on only when both of the inputs are on. Put your ladder diagram into the simulator and test that it works as specified.

11.4 Use again the PLC Ladder Simulator to simulate the functioning of a stop light. Name O0.0-O0.2 GRN, YLLW, and RED, respectively. Have the sequence of illuminating the lights so that the green and red lights stay on 5 seconds each, the yellow 2 seconds. This sequence should start when you turn on a STRT switch (I0.1) and continue until you turn this switch off.

11.5 View the simulation video of the Siemens PLC S7-1200 on YouTube (https://www.youtube.com/watch?v=1AKlU1Zw768) and answer the following questions.

a. Why is this video named “Siemens PLC...”?

b. During the simulation you will notice that N1 reads a certain rotational speed in rpm as does N3. Why are these values different? What does the SCADA graphic tell you about the configuration of the machine?

c. What is the gear ratio of this machine?

d. How is the rotational speed of the machine governed?

e. Before the turbine is running, there is no flow through it. How is it started?

f. What is roughly the air/fuel ratio of this turbine by weight?

g. There is a funny thing about this system, regarding its power output, during the course of this video. What is that?