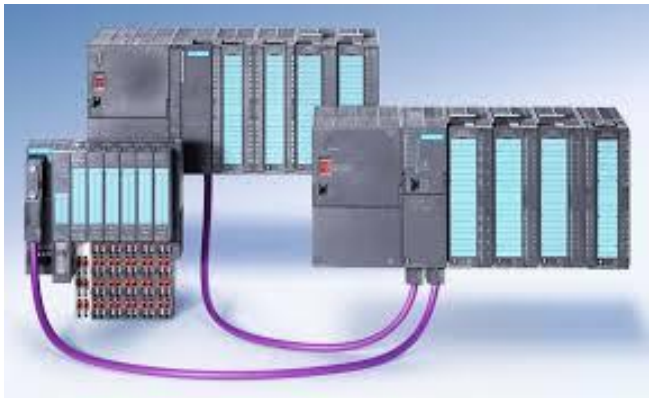
King Saud University

College of Engineering, Electrical Engineering Department

EE 457

Applied Control Laboratory



Student Name:

ID Number :

[Updated: Feb, 2015]

Course Description:

EE 457 – Applied Control Laboratory *1(0,0,2)*
This laboratory is equipped with basic instruments and real time experiments that are necessary to familiarize the students with the advanced concepts and updated technology in the control field. The undergraduate experiments are designed to reinforce and expand many concepts covered in the advanced control course EE 454 and digital control course EE483. Experiments are organized in several groups of real time applications, such as:
- Data Acquisition and system modeling
- Computer control system using MATLAB
- Digital Control using PLC.

Textbooks: LAB Notes are prepared including a complete set of experiments.
Co-requisite: *EE 456*.

[*Course description is taken from EE KSU new plan, 2008*]

Contents of this manual:

Note: The PLC part in this manual is compiled from previous manual (EE352), while sensor part is compiled from QNET MECHKIT Lab Workbook. Some improvements are still needed to enhance this manual. [Sutrisno Ibrahim]

Experiment # P1

PLC (S7) Hardware and Software

1. Objectives

Upon the completion of this experiment, the student should be able to:

- Identify the major components of a PLC (for example SIMATIC S7) and describe their functions
- Read a basic ladder logic diagram and statement list
- Implement basic logic operation, motor start and stop circuit, timers and counters.
- Solve basic PLC problems in general.

2. Theory and Background

2.1. Programmable Logic Controllers (PLCs)

Many manufacturing operations are ON/OFF in nature, that is a conveyor or heaters is either on or off, a valve is either open or closed, and so on. In the past, these types of discrete control functions were often provided by a system of electrical relays wired according to a complex diagram into what was called a relay logic system.

In recent years, computers have also taken over the operation of such relay logic controllers, known as programmable logic controllers (PLCs). Even though originally designed to control discrete state (ON/OFF) systems, they are capable also to control analog control loops.

A PLC monitors inputs, makes decisions based on its program, and controls outputs to automate a process or machine as shown in figure 1.This section is meant to supply the reader with basic information on the functions and configurations of PLCs.

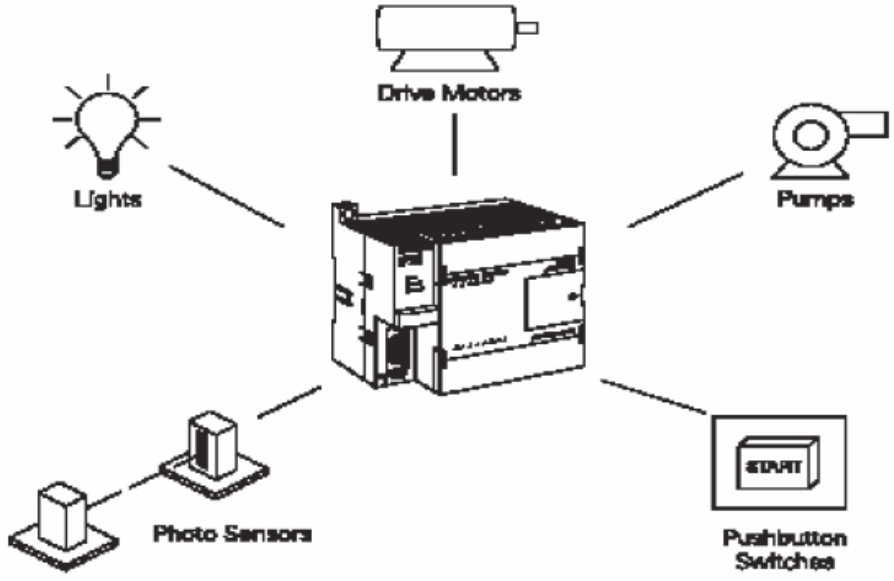


Figure P1.1: PLC operation environment

2.1. Fundamental elements in a PLC

Typically a PLC system has five basic elements:

- Input modules: 5 V, 24 V, 110 V, 240 V DC
- Output modules: relay type (ac and dc switching), transistor type (dc switching), or triac type (loads with ac power supply)
- Central processing module (CPU) + memory
- Power supply module
- Programming device
- Operator interface (optional)

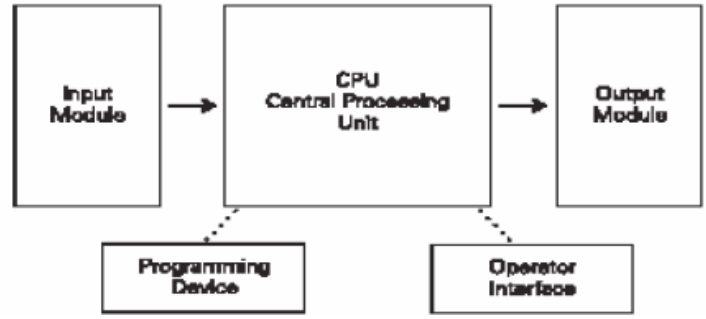


Figure P1.2: PLC elements

The *input module* accepts a variety of digital or analog signals from various field devices (sensors) and converts them into a logic signal that can be used by the CPU. The *CPU module* makes decisions and executes control instructions based on program instructions in memory. *Output modules* convert control instructions from the CPU into a digital or analog signal that can be used to control various field devices (actuators). A *power supply module* is needed to convert the main a.c. voltage to necessary voltage level for the processor and the circuits in the input and output modules. A *programming device* is used to input the desired instructions in the memory of the processor. These instructions determine what the PLC will do for a specific input. An *operator interface* device allows process information to be displayed and new control parameters to be entered. This module is optional and may be not found in some applications.

3. Required equipment for the experiment

The aim of this first experiment is to understand the functionality of the PLC SIMATIC S7. For that experiment you need the following equipment:

- SIMATIC S7 device
- Suitable software for programming the PLC
- Programming device (PC)

4. Experiment Steps

1. SIMATIC S7-Software (Step7 lite) installation, which is already done.
2. SIMATIC S7-Hardware configuration
3. Control program implementation using the Step7 lite and then downloading the program into the CPU of the SIMATIC S7
4. Testing the loaded program in PLC

4.1. PLC-Software

The corresponding software to the used PLC must be installed on the programming device like as PC. Then the hardware of the PLC must be defined for the software, this step is called hardware configuration. Control programs can be implemented in a Project of the software and then downloaded to the already configured CPU of the PLC. After providing the CPU with the program, the PLC will be ready to control a process.

Note that the software Step7 Lite is already installed on the PCs.

4.2. Hardware configuration

4.2.1. Basic Procedure for Configuring Hardware

Starting the Hardware Configuration

When you have implemented a new project, open the workspace for configuring and assigning parameters to modules by double-clicking the "**Hardware**" symbol.

Workspace for Configuring

The workspace for configuring a programmable controller consists of the following areas:

- The graphical overview, in which the racks are realistically represented with modules.
- Table(s) which represent individual racks, but which contain additional information about the modules (order number and addresses, etc.), in contrast to the graphical overview.
- The "Hardware Catalog" from which you select the required hardware components, for example, modules and interface submodules.

4.2.2. Basic Steps for Configuring a Station

Independent of which structure a station has – you always configure using the following steps:

1. Select a hardware component in the “Hardware Catalog”.
2. Copy the selected component using drag & drop
   - To a slot on the rack in the graphical view or
   - To a row in the configuration table that represents the structure of the rack.

Tips – Inserting modules

1. Copy the selected component on the rack in the graphical view beginning from the left to right, insert the power supply module, CPU, input module, output module.
2. Modules must be inserted without leaving empty slots between them. Otherwise, they cannot be supplied with power via backplane bus.

Exception in Step7 lite: Slot 3 is reserved for the interface module (IM) you can use to connect racks stacked on top. If modules are inserted only in the lowest rack, you can leave a space in the configuration.

4.3. Implementing a control program

A program consists of one or more instructions that accomplish a task. Programming a PLC is simply constructing a set of instructions. There are several ways to look at a program such as ladder logic, statement lists, or function block diagrams.

4.3.1. Ladder Logic Diagram (LAD)

Ladder logic (LAD) is one programming language used with PLCs. Ladder logic uses components that resemble elements used in a line diagram format to describe hardwired control. The left vertical line of a ladder logic diagram represents the power or energized conductor. The output element or instruction represents the neutral or return path of the circuit. The right vertical line, which represents the return path on a hard-wired control line diagram, is omitted. Ladder logic diagrams are read from left to-right, top-to-bottom. Rungs are sometimes referred to as networks. A network may have several control elements, but only one output coil.

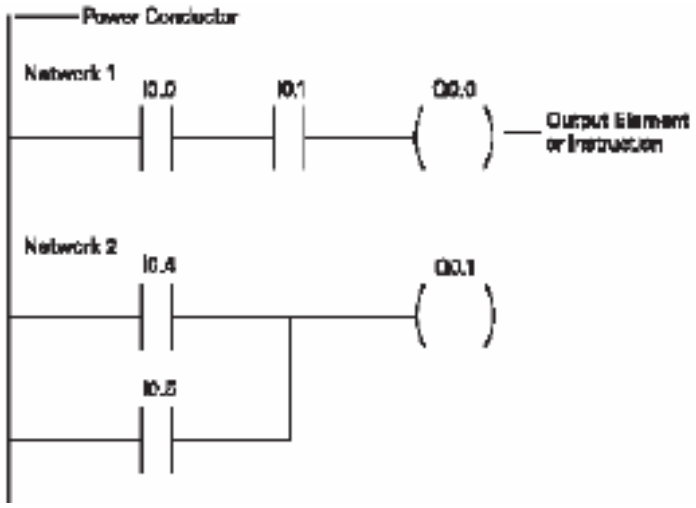


Figure 3: Ladder logic diagram example

In the example shown above I0.0, I0.1 and Q0.0 represent the first instruction combination. If inputs I0.0 and I0.1 are energized, output relay Q0.0 energizes. The inputs could be switches, pushbuttons, or contact closures. I0.4, I0.5, and Q1.1 represent the second instruction combination. If either input I0.4 or I0.5 is energized, output relay Q0.1 energizes.

How to program in LAD

1. Under View, select symbolic representation.
2. At OB1, enter your program. At Network 1, enter connecting in series. For connecting in parallel left click on the left Line of the Network 1 and then right click to select Branch Open. After connecting the desired elements in the second Branch you have to connect the right side of this Branch with the suitable Point in the first Branch.
3. For inserting program elements there are three methods:
   a. Click on the desired icon on the toolbar
   b. Right-click on the circuit to open the pop-up menu. Select what do you need.
   c. Drag the coil to your circuit using the drag-and-drop operation.
4. you have to name and address the contacts and coils. Click on the ??.?. Enter the symbolic name “any name” (with quotation marks). Then click outside the circuit to get the edit symbol, where you can address the symbolic name.

Your program is already implemented. You can download it to the CPU to control your process.

4.3.2. Statement list (STL)

A statement list (STL) provides another view of a set of instructions. The operation, what is to be done, is shown on the left. The operand, the item to be operated on by the operation, is shown on the right. A comparison between the statement list shown below, and the ladder logic shown on the previous page, reveals a similar structure. The set of instructions in this statement list perform the same task as the ladder diagram.

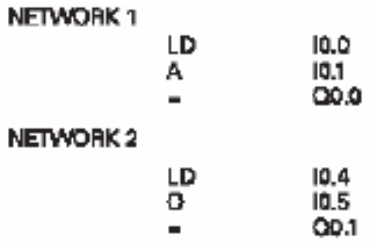


Figure 4: Statement list

4.3.3. Function Block Diagram (FBD)

Function Block Diagram (FBD) provides another view of a set of instructions. Each function has a name to designate its specific task. Functions are indicated by a rectangle. Inputs are shown on the left-hand side of the rectangle and outputs are shown on the right-hand side. The function block diagram shown below performs the same function as shown by the ladder diagram and statement list.

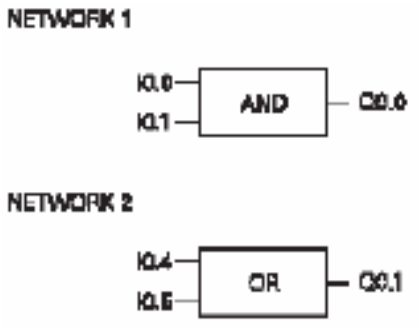


Figure 5: Functional block diagram

Note that, you can convert your control program from any programming language (LAD, STL or FBD) into another language by clicking on the view on the toolbar and selecting the desired language.

4.4. Implementation of the experiment

After you have configured the hardware (see 4.2), you have to develop the following basic software modules:

1. Basic logic operation
2. Motor start and stop circuit
3. Timers
4. Counters

4.4.1 Basic logic operation

The following programming example demonstrates an OR operation for the results of two AND operations. Two normally closed contact closures are placed in the first branch and two normally open contact closures are placed in the second branch. They were assigned addresses I124.0, I124.1, I124.2 and I124.3 respectively. If the signal state of both inputs I124.0 and I124.1 is "0" or the signal state of I124.2 and I124.3 is "1", the output Q124.0 will be "1". Except that the output Q124.0 will be "0".

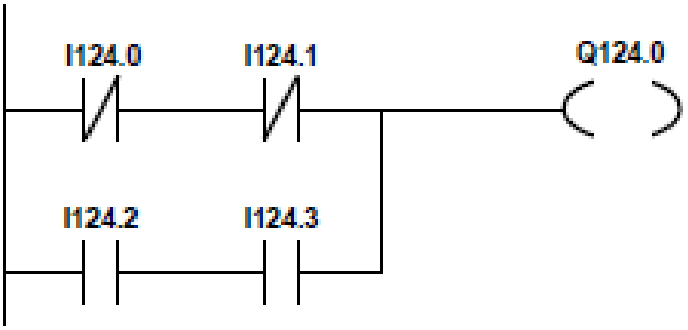


Figure 6: An example for basic logic operation

4.4.2. Motor start and stop circuit

The following example involves a motor start and stop circuit. The line diagram illustrates how a normally open and a normally closed pushbutton might be used in a control circuit. In this example a motor started (M) is wired in series with a normally open momentary pushbutton (Start), a normally closed momentary pushbutton (Stop), and the normally closed contacts of an overload relay (OL).

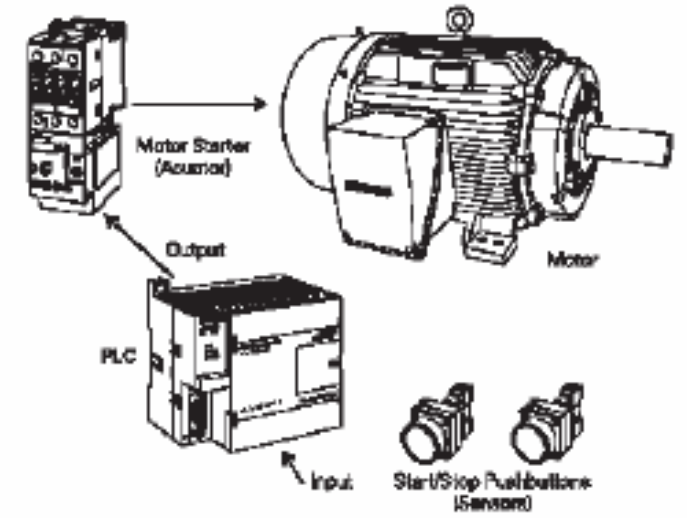


Figure 7: Motor start/stop components

Motor start and stop experiment:

A normally open Start pushbutton is wired to the first input (I124.0), a normally closed Stop pushbutton is wired to the second input (I124.1), and normally closed overload relay contacts (part of the motor starter) are connected to the third input (I124.2). The first input (I124.0), second input (I124.1), and third input (I124.2) form an AND circuit and are used to control normally open programming function contacts on the network. I124.1 status bit is logic 1 because the normally closed (NC) Stop Pushbutton is

closed. I124.2 status bit is logic 1 because the normally closed (NC) overload relay (OL) contacts are closed. Output Q124.0 is also programmed on the network. In addition, a normally open set of contacts associated with Q124.0 is programmed on the network to form an OR circuit. A motor starter is connected to output Q124.0.

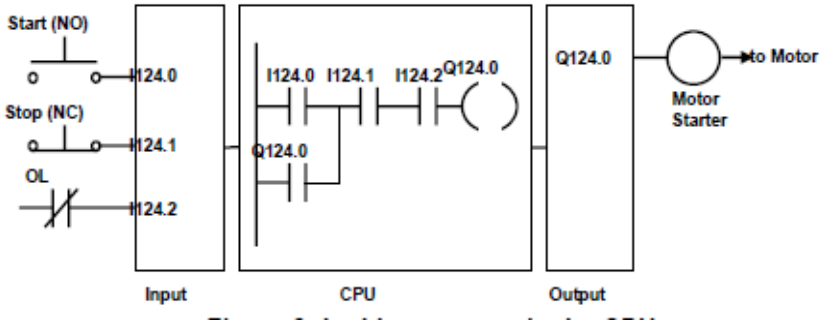


Figure 8: Ladder program in the CPU

Note that, momentarily depressing the Start pushbutton completes the path of current flow and energizes the motor starter (M).This closes the associated (auxiliary contact located As marker or memory variable in the PLC) contact Q124.0. When the Start button, the auxiliary contacts Q124.0 remains closed as the output is energized. The motor will run until the normally closed Stop button is depressed, or the overload relay opens the OL contacts, breaking the path of current flow to the motor starter and opening also its associated contact.

4.4.3. Timers

Timers are devices that count increments of time. Traffic lights are one example where timers are used. In this example timers are used to control the length of time between signal changes.

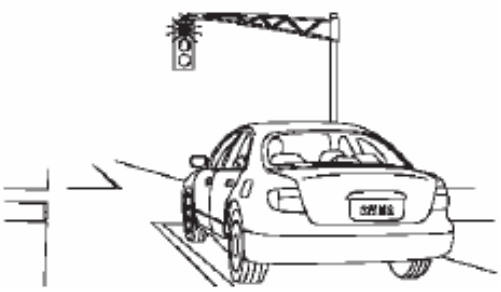


Figure 9: Timers application in traffic lights

Timers are represented by boxes in ladder logic. When a timer receives logic “1” at the start input (S), the timer starts to time. The timer compares its current time with the preset time value (TV). The output (Q) of the timer is logic 0 as long as the current time is less than the preset time. When the current time is greater than the preset time the timer output is logic 1. There are three types of timers: On-Delay (ODT), Retentive On-Delay (ODTS), and Off-Delay (OFFDT).

On-Delay (ODT)

On-Delay timer starts the specified timer if there is a positive edge at the start input (S). A signal change is always necessary in order to enable a timer. The timer runs for the time interval specified at input (TV) as long as the signal state at input (S) is positive. The signal state at output (Q) is “1” when the timer has elapsed without error and the

signal state at the (S) input is still “1”. When the signal state at input (S) changes from “1” to “0” while the timer is running, the timer is stopped. In this case the signal state of output (Q) is “0”.

The timer is reset if the reset input (R) changes from “0” to “1” while the timer is running. The current time and the time base are set to zero. The signal state at output (Q) is then “0”. The timer is also reset if there is a logic “1” at the (R) input while the timer is not running and the RLO (Result of logic operation) at input (S) is “1”. The current time value can be scanned at the outputs (BI) and (BCD). The time value at (BI) is binary coded, at (BCD) is BCD coded. The current time value is initial (TV) value minus the time elapsed since the timer was started.

Timer experiment:

Consider a switch is connected to input I124.0 and a light is connected to output Q124.0 as shown in the ladder program below.

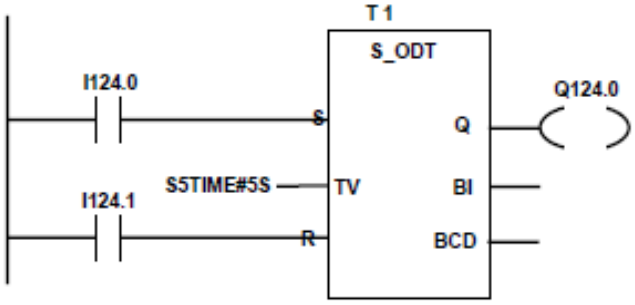


Figure 10: On-Delay timer

In this example, When the switch I124.0 is closed (changing its signal state from “0” to “1”) the timer T1 will be started. The preset time value (TV) has been set to 5. The associated contact T1 will be closed after 5 seconds and the light will turn on (Q124.0 will be “1”) as long as the switch I124.0 is still closed. If the switch I124.0 were opened before 15 seconds had passed, the timer is stopped and the light is turned off (if the signal state of the reset input I124.1 changes from “0” to “1”, the time is reset regardless of whether the timer is running or not).

4.4.4 Counters

Counters used in PLCs serve the same function as mechanical counters. Counters compare an accumulated value to a preset value to control circuit functions. Control applications that commonly use counters include the following:

- Count to a preset value and cause an event to occur
- Cause an event to occur until the count reaches a preset value

A bottling machine, for example, may use a counter to count bottles into groups of six for packaging.

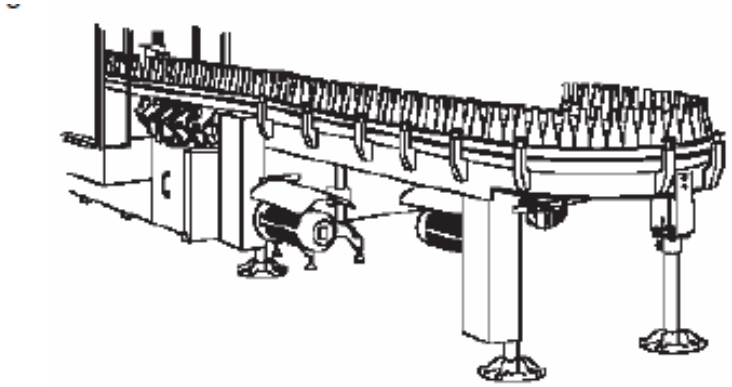


Figure 11: Counters in packaging application

Counters are represented by boxes in ladder logic. Counters increment/decrement one count each time the input transitions from off (logic 0) to on (logic 1). The counters are reset when a RESET instruction is executed. There are three types of counters: up counter (CU), down counter (CD), and up/down counter (CUD) as shown in the following figure.

Up/Down Counter (CUD)

If the set input (S) changes from logic 0 to logic 1, the counter is preset with the value at the input (PV). If the signal state of (CU) changes from logic 0 to logic 1, the value of counter (C no.) will be incremented by one – except when the value of (C no.) equal to "999". If (CD) changes from logic 0 to logic 1, the value of counter (C no.) will be decremented by one - except when the value of (C no.) equal to "0". The output Q is "1" if the current value (C no.) is not equal to zero.

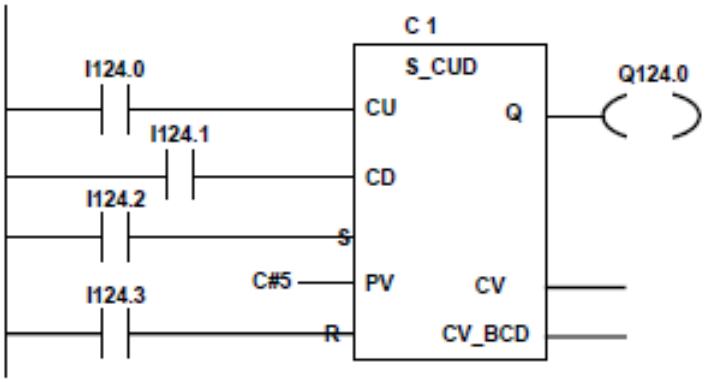


Figure 12: Up-Down counter

Counter experiment:

A counter might be used to keep track of the number of vehicles in a parking lot. As vehicles enter the lot through an entrance gate, the counter counts up. As vehicles exit the lot through an exit gate, the counter counts down. When the lot is full a sign at the entrance gate turns on indicating the lot is full.

Up/down counter C1 is used in this example. A switch, connected to the entrance gate, has been wired to input I124.0. A switch, connected to the exit gate, has been wired to input I124.1. A reset switch, located at the collection booth, has been wired to input I124.3. The parking lot has 5 parking spaces. This value has been stored in the preset value (PV) and the counter will be loaded with this value, when the signal state at the

set input (S) changes from logic 0 to logic 1. The counter output has been directed to output Q124.0. As cars enter the lot the entrance gate opens. Input I124.0 transitions from logic 0 to logic 1, incrementing the count by one. As cars leave the lot the exit gate opens. Input I124.1 transitions from a logic 0 to a logic 1, decrementing the count by 1. When the count has reached zero, the output Q124.0 turns off. When a car enters, incrementing the count to 1, the sign at the output turns on.

5. Review questions

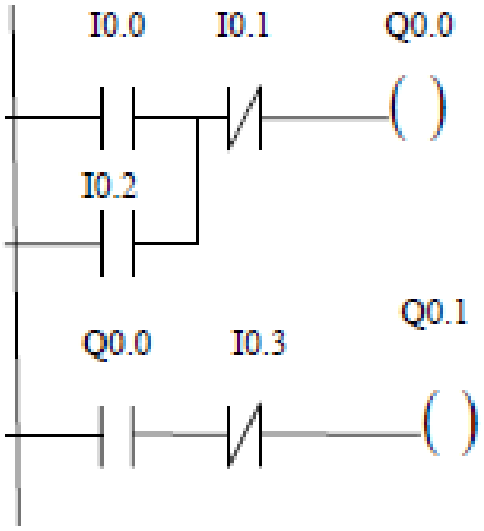
Select the correct answer for question 1 and 2.

1] The _____ makes decisions and executes control instructions based on the input signals.
A) CPU
B) Input module
C) Power supply

2] _____ is a PLC programming language that uses components resembling elements used in a line diagram.
A) Ladder logic diagram
B) Statement list
C) Function block diagram

3] Draw a block diagram for the PLC showing the main functional items, and enumerate the main steps to control a process with the PLC.

4] For the PLC ladder program shown in the following figure, state what input conditions have to be met in each case for there to be an output from Q0.0 and Q0.1.



Experiment # P2

S7: Advanced Programming Techniques

1. Objectives

Upon the completion of this experiment, the student should be able to:

- Introduction to Siemens PLCs programming and Step-7 light software tools (Symbol table, Set, reset operations, and memory flags).
- Familiarization with Siemens PLCs timers and counters.
- Study of some actual example in programming of the PLC.

Procedure:

• Create new step 7 light software project.
• Open the organization block OB1.

[A]: Programming with Symbols

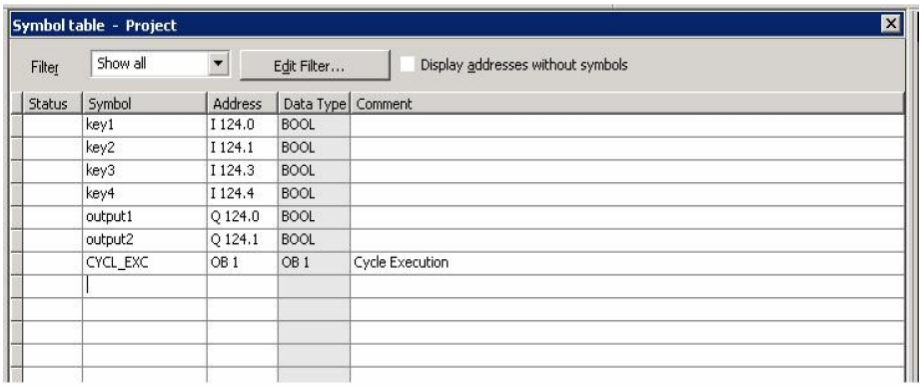
Absolute Addresses

Every input and output has an absolute address predefined by the hardware configuration. This address is specified directly; that is, absolutely for example I 124.0 is an absolute address. The absolute address can be replaced by any symbolic name you choose.

K9

In the symbol table, you assign a symbolic name and the data type to all the absolute addresses which you will address later on in your program; for example, for input I 124.0 the symbolic name Key 1. These names apply to all parts of the program and are known as global variables.

- Open the symbol table by clicking on its symbol in the left control menu or selecting (View >> Symbols) from the menu bar, or by pressing (Ctrl+Alt+Y) on the keyboard.
- Fill the symbol table as follows:



• Save the symbol table and Exit from the symbol editor.
• To change from absolute addressing to symbolic addressing, select the menu item (View>>Display>>Symbolic Representation) or press on the symbol in the bar menu.

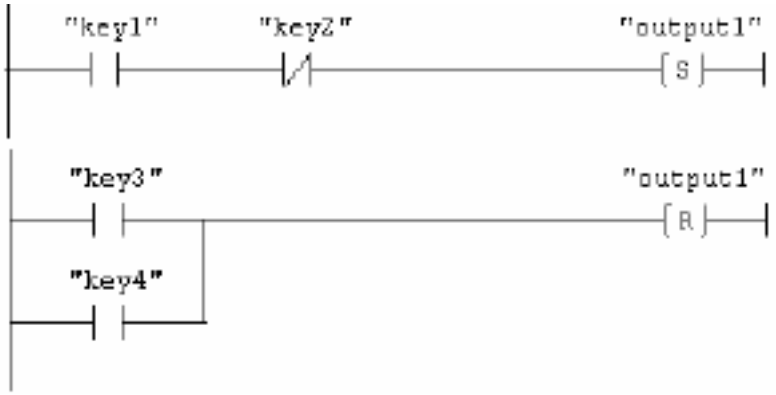
Note: We refer to the “result of logic operation” By RLO.

[B]: Set & Reset Coil

The Set Coil instruction is executed only if the RLO = 1. If the RLO = 1, this instruction sets its specified address to 1. If the RLO = 0, the instruction has no effect on the specified address. The address remains unchanged. The Reset Coil instruction is executed only if the RLO = 1. If the RLO = 1, this instruction resets its specified address to 0. If the RLO = 0, the instruction has no effect on its specified address. The address remains unchanged.

Q1) What is the difference between output coil and set coil instructions?

• Create the following ladder diagram in your project, and download it to the PLC S7, then test its operation.



Q2) What is the logic function for both set and reset operations in the above ladder diagram?

[C]: Other PLC Timers

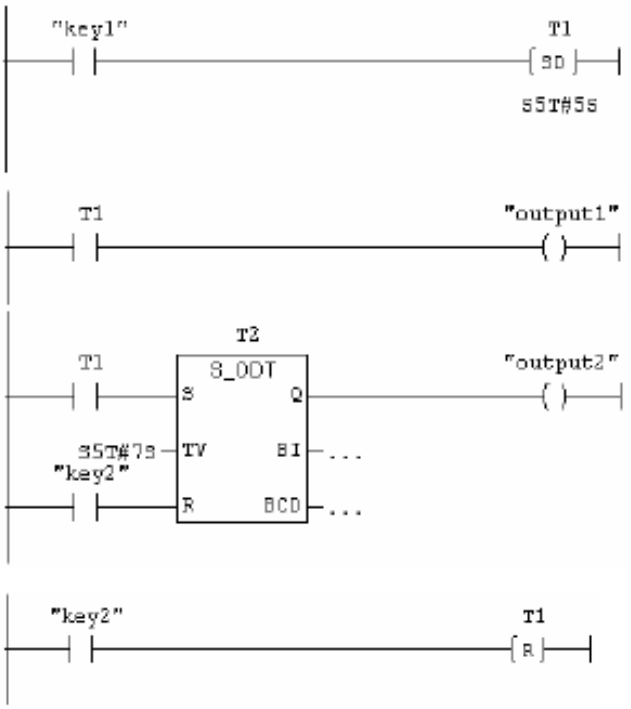
Timers have an area reserved for them in the memory of your CPU. This memory area reserves one 16-bit word for each timer address. The ladder logic instruction set supports 256 timers.

(SD) (On Delay Timer Coil) starts the specified timer with the <time value> if there is a positive edge on the RLO state. The signal state of the timer is "1" when the <time value> has elapsed without error and the RLO is still "1". When the RLO changes from "1" to "0" while the timer is running, the timer is reset. In this case, a scan for "1" always produces the result "0."

- Open the timer's folder in the program elements, and check out the different types of timers. Mark each timer and press F1 to display the help to that timer.

Q3) From the help, explain the operation of the SP timer.

- Create the following ladder diagram, and then test its operation.



Note: in the above ladder diagram the timer T2 have the following parameters:

- TV: this is the on delay time we will use to delay the input signal at the timer input (S).
- BI: is the current time value in binary form.
- BCD: is the current time value in BCD form.

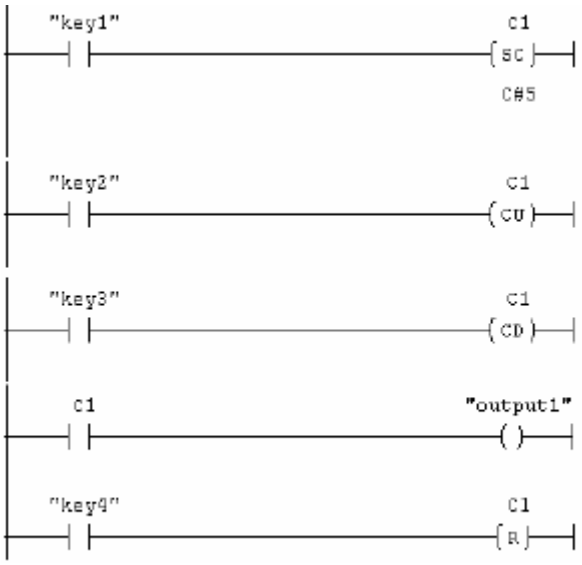
We note that when we apply a signal (key1) to the T1(5s delay), output 1 will be operated after 5 seconds, also at this moment T2 will be initialized, this means that output 2 will be operated after 12 seconds (T1+T2).

[D]: Other PLC Counters

Counters have an area reserved for them in the memory of your CPU. This memory area reserves one 16-bit word for each counter address. Bits 0 through 11 of the counter word contain the count value in BCD format.

Q4) what is the range of the count value of each counter?

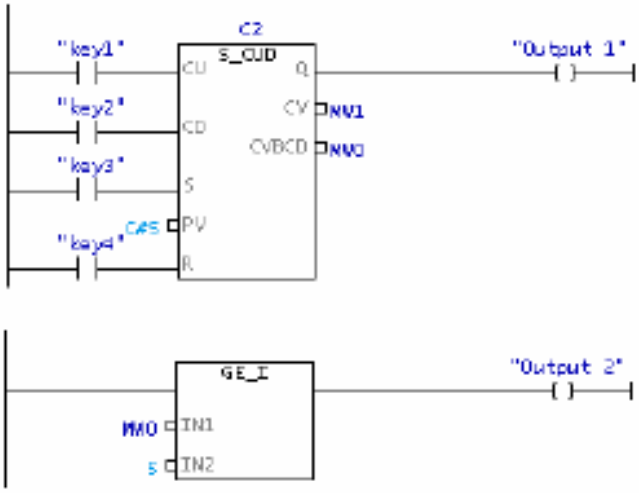
• Create the following ladder diagram, and then test its operation using PLC.



[E]: Comparison Instructions

• Create the following ladder diagram, and then test its operation using the PLC .

- Try to test other types of comparator.



[F] Designing Traffic Light system Using PLC.

We are going to control traffic light system using plc with the following sequence.

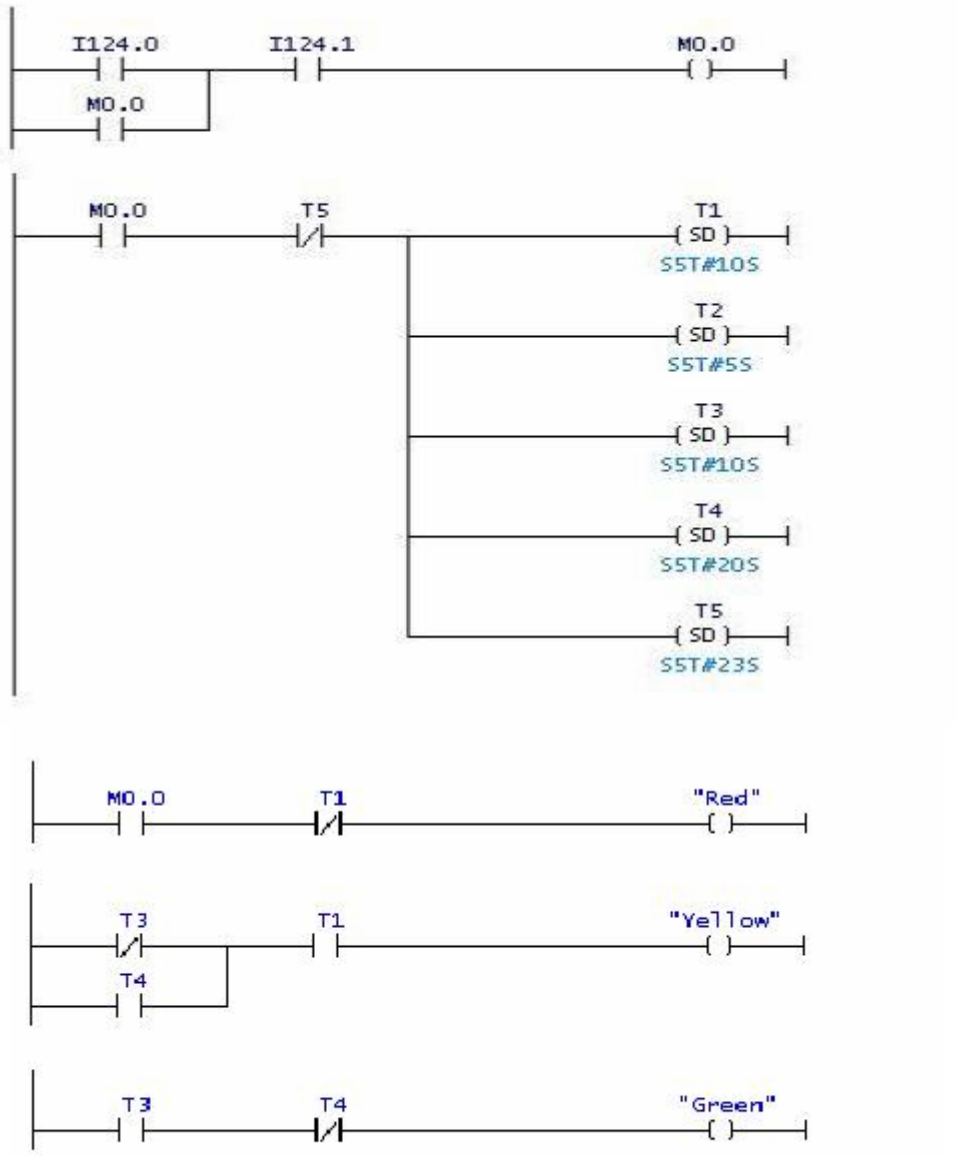
1- The red light is on for specific time.
2- Both red and yellow lights are on for specific time.
3- Both red and lights turn off and the green light turns on.
4- The green light is on for specific time.
5- The green light turns off and the yellow light is turned on for a specific time.
6- The yellow light turns off and the red light turns on again.

Let us program this on the PLC and test its operation. Insert the following in the symbol table

Q124.0= Red light.

Q124.1 = Yellow light.
Q124.2= Green light.

Then write the following program, and test its operation. Try to explain what happens in the timers each time.



Q5) what is the function of the external input I 124.0?

Experiment # P3

Control of Conveyor System Using PLCs and Sensors

Objectives

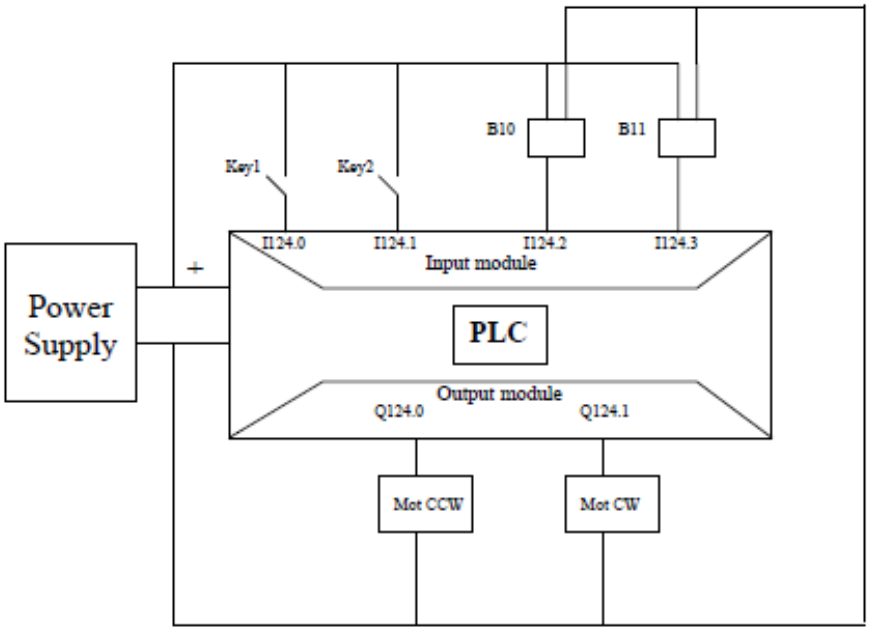
Upon the completion of this experiment, the student should be able to Objectives:
• To learn how PLCs are used in automation systems.
• To learn how to use sensors in automation systems.

Equipments Required:
Siemens S7 PLC unit, conveyor belt, proximity sensor (B11), photo sensor (B10), Metallic cylinder, and plastic cylinder.

Theoretical background:

The single track conveyer belt is for connecting two operating stations or units in order to assemble flexible manufacturing facilities .using automatic handling equipment it can be combined with any functional unit or operating module. The conveyor belt consists of a single-track belt that is equipped with an electrical drive and includes an optical sensor for detecting when work pieces reach the end of the conveyor and an inductive sensor in the middle of the belt. The mechanical design of the belt employs metal profiles that allow for additional installation options (sensors, cylinders) to easily be retrofitted.

Electrical wiring diagram:



Procedure:
• Create new step 7 light software project.

Fill the symbol table bellow.

Symbol table - Project

Filter: Show all | Edit Filter... | Display addresses withou

Status	Symbol	Address	Data Typ	Comment
	CYCL_EXC	OB 1	OB 1	Cycle Execution
	key1	I 124.0	BOOL	
	key2	I 124.1	BOOL	
	B11	I 124.2	BOOL	
	B10	I 124.3	BOOL	
	Motor CCW	Q 124.0	BOOL	
	Motor CW	Q 124.1	BOOL	

• Save the symbol table and Exit from the symbol editor.

[A]: Simple operation of the conveyor belt.

The following figure shows a conveyor belt that can be activated electrically. Two start buttons (key1 and key2) is used to move the conveyor belt in both directions, and one button is used to stop it. The proximity sensor is used to detect metallic cylinder only and photo sensor is used to detect both of the cylinders, such that when the cylinder reaches the sensor, the conveyor belt stop.

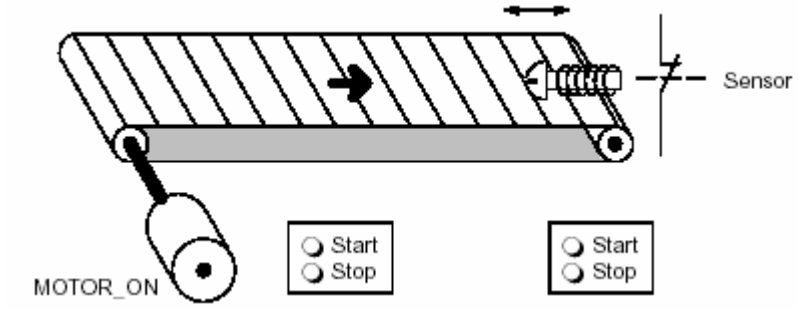
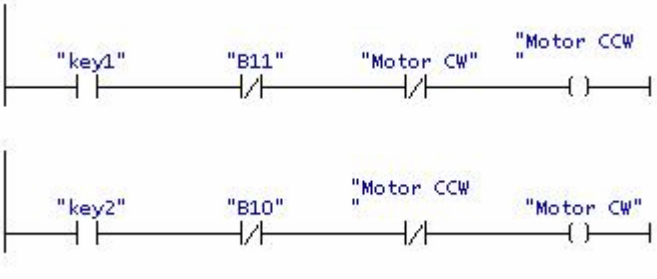


Fig.1 The conveyor belt.

Write the following program and connect the sensors and motor terminals to the needed PLC inputs according to the symbol table above. Put the metallic item between the two sensors and then write the following program in the PLC and test its operation as follows.

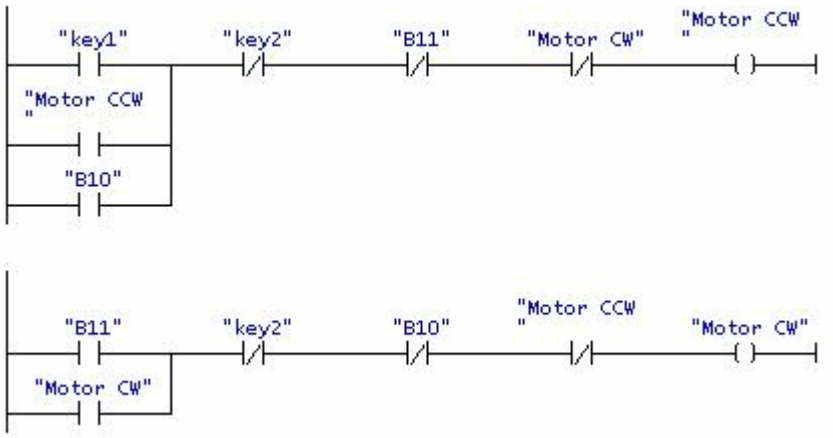


-Actuate Key1 and see what happens; comment.
-Actuate key 2 and see what happens; comment.
-Actuate the two keys at the same time and note the operation of the conveyer belt, comment.

Q1) What happens if you actuate the two operating keys at same time? Why? Now remove the metallic object, put the Plastic object, and operate the system; comment.

B) Alternating movement of the conveyor.

• Create the following ladder diagram in your project, and download it to the PLC S7, then test its operation.



This program will operate the conveyer in the alternating movement.

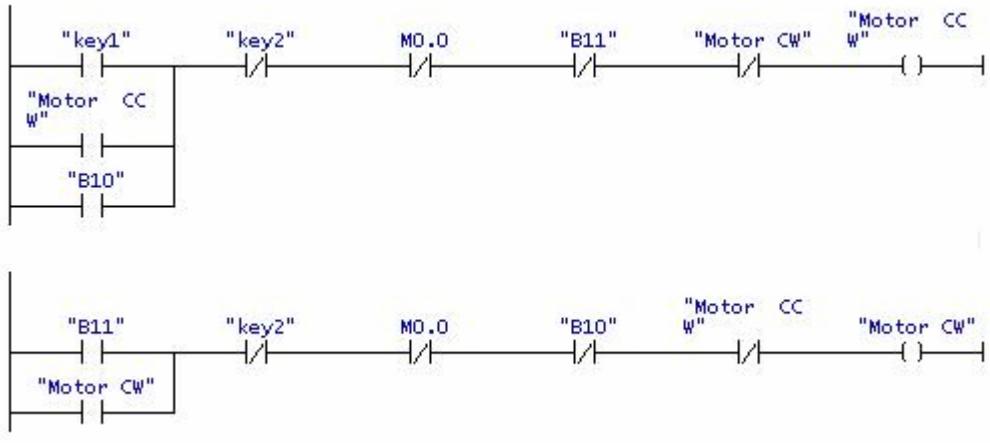
- Put the metallic cylinder between the two sensors.
- Actuate key1 and note what happens, write that in your notes and try to explain the logic of operation.
- Now actuate key2 and note that the conveyer will stop.

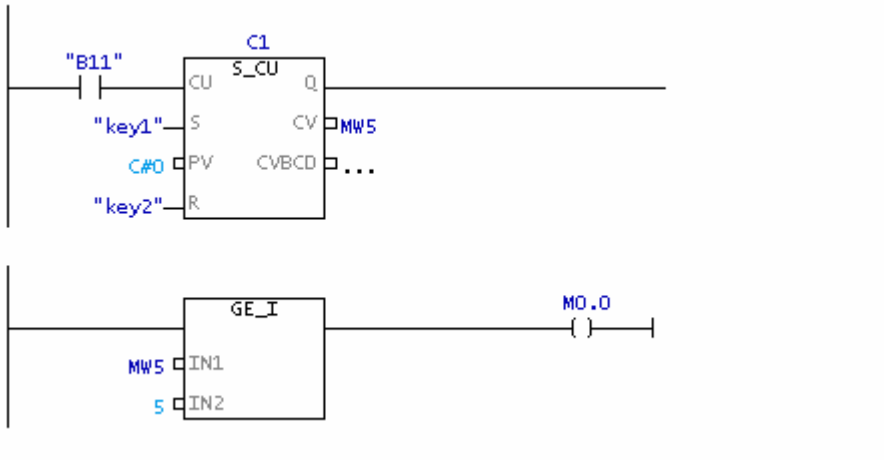
Q2) What is the function of B11, B10, and key1?

[C]: Controlling Storage Area with Counter and Comparators.

In this part, the conveyor belt is used to deliver items to a temporary storage area and to transport these items from that storage area. Using a proximity sensor, we must keep track of the number of items in the storage area (5 times). When the conveyor belt is on in the right direction, it delivers items to the storage area, while when it is on in the left direction, it transport items from the storage area.

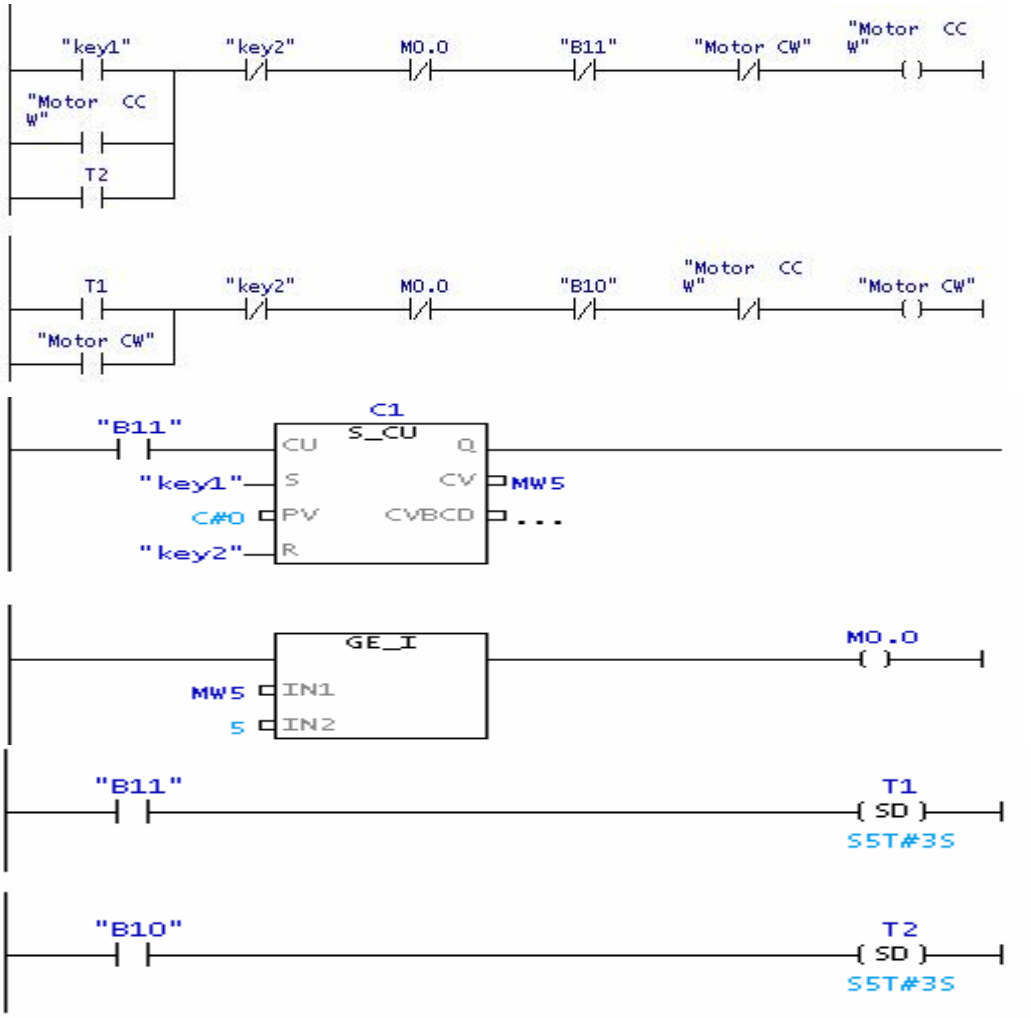
- Put the metallic item between the two sensors.
- Write the following program in the PLC.
- Test the operation of this circuit and write your notes.





[D]: Controlling the conveyor belt periodically with stops at the sensor locations.

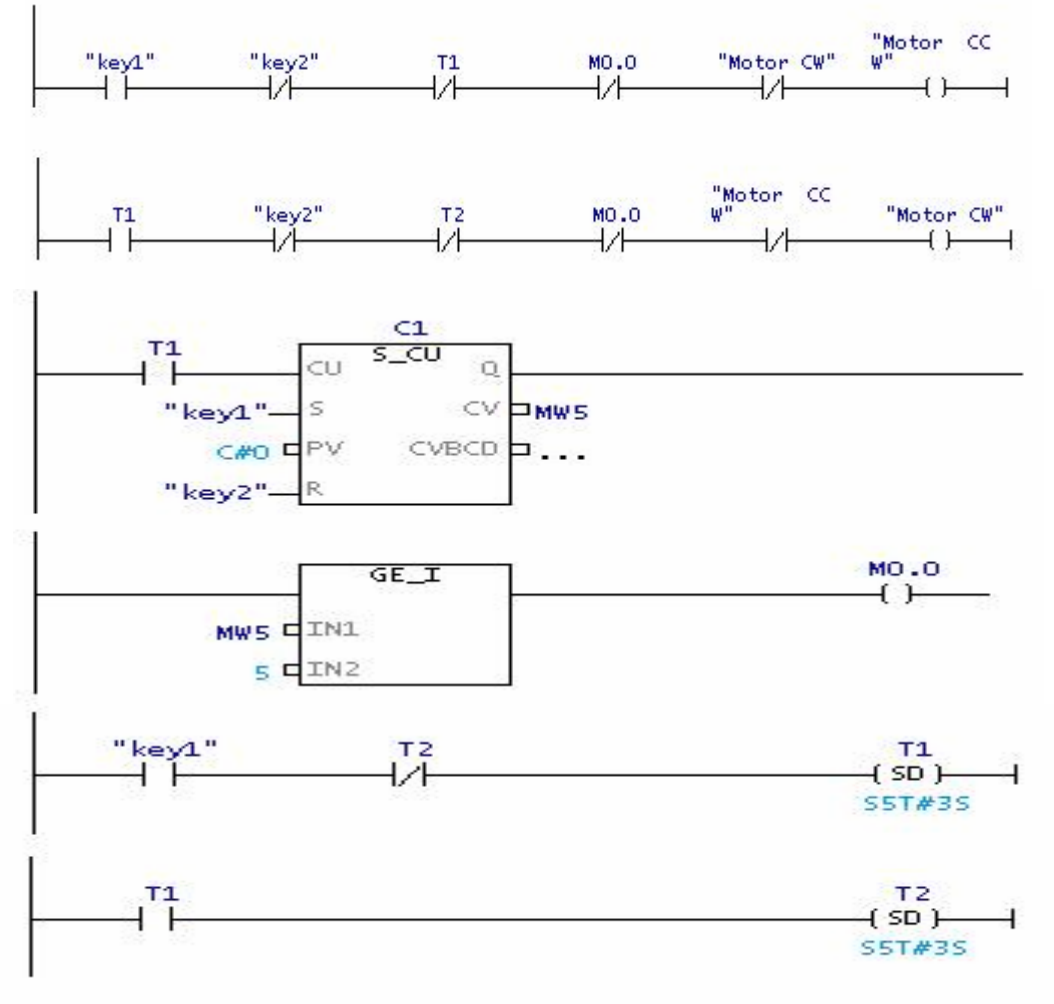
In this practice we want to transfer the metallic cylinder periodically between position proximity sensor and photo sensor for 5 times, and after that the belt will stop in front of proximity sensor. In all positions the metallic piece will stop 3 seconds in front of these sensors.



[E]: Design problem.

Using timers and counters, design a program that operates the conveyer exactly as part C.

Note: Don't use the sensors B11, and B 10 in your programming.



Experiment # S1

Introduction and Strain Gage Sensor

1. Introduction

Mechatronics engineering is a cross-disciplinary field that combines mechanical and electronic design in control systems architecture though the application of computer programming. One of the most useful topics that can be covered in an introductory mechatronics course is the understanding and application of sensors. Various sensors are used in all types of industries. For example, in the automotive industry magnetic field transducers are used for throttle, pedal, suspension, and valve position sensing. In assembly line and machine automation, optical sensors are used for non-contact position sensing and safety. Piezo film sensors are installed in packages to log vibration history of a shipment.

The QNET mechatronics sensors (MECHKIT) trainer is shown in Figure 1.1. It has ten types of sensors, two types of switches, a push button, and two LEDS. This QNET module can be used to teach the physical properties of most sensors used today, and the techniques and limitations of their application.

Figure 1.1: QNET Mechatronic Sensors Trainer (MECHKIT)

There are 12 experiments that can be done with the kit: strain gage with flexible link, pressure sensor, piezo sensor, potentiometer, infrared, sonar, optical position, magnetic field, encoder, temperature sensor, switches and LEDs, and switch debounce analysis. The experiments can be performed independently. In order to successfully carry out this laboratory, the user should be familiar with the **LabVIEW** to run Visual Instruments (Vis).

2. SENSOR PROPERTIES

This section discusses various sensor properties that are often found in technical specifications.

1. Resolution
   The resolution of a sensor is the minimum change that can be detected in the quantity that is being measured. For instance a sensor that measures angular position of a motor shaft may only be able to detect a 1 degree change. Thus if the motor moves 0.5 degrees, it will not be detected by the sensor. Depending on the precision needed for the application, this may be adequate.

2. Range
   Range sensors can only take measurements of a target within a certain operating range. The operating range specifies a maximum, and sometimes also a minimum, distance where the target can be from the sensor in order to obtain an accurate measurement. Sensors with a small range are the magnetic field and optical position sensors. Sensor with a relatively larger range are infrared and sonar.

3. Absolute and Incremental
   Absolute sensors detect a unique position. Incremental sensors measure a relative position that depends on a prior position or last power on/off. For example, if an incremental rotary encoder is used to measure the position of wheel, the encoder will measure zero every time its power is reset. If an absolute sensor such as a rotary potentiometer is used, then it will detect the same angle regardless if it has just been powered.

4. Analog Sensor Measurement
   Analog sensors output a signal that correlates to the quantity it is measuring. The relationship between the output signal of the sensor and the actual measurement varies depending on the type of sensor. For example, the voltage measured by a potentiometer is directly proportional to the angle it is measuring. However, the resistance of a thermistor decreases exponentially as the temperature increases.

Linear sensors can be modeled using the equation

$$y = ax + b \qquad (2.1)$$

where a is the rate of change and b is the offset. Variable x is the sensor output signal and y is the measurement, e.g. for the potentiometer x would be the voltage measured by the sensor and y would be the angular measurement (in either degrees or radians). Other types of sensors need to be characterized by more complex relationship such as polynomial

$$y = ax^2 + bx + c \qquad (2.2)$$

or exponential

$$y = ae^{bx} \tag{2.3}$$

Some of the different ways to characterize analog sensors is illustrated in Figure 2.1.

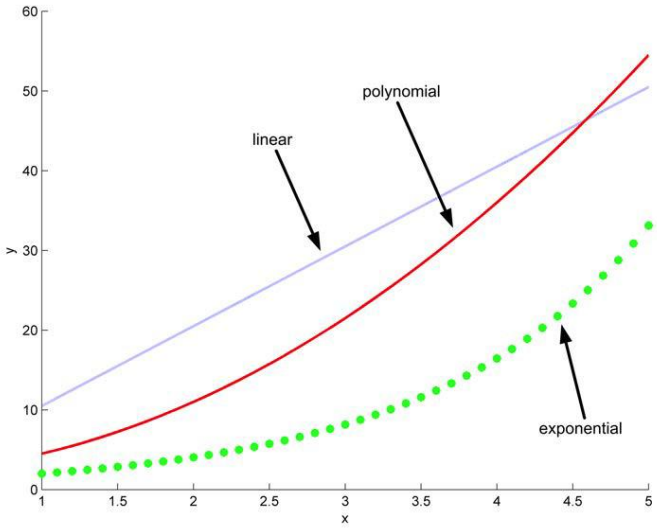


Figure 2.1: Different sensor responses

3. Strain Gage with Flexible Link

3.1 Background

A strain gage measures strain, or deflection, of an object. As shown in Figure 3.1, in the QNET mechatronic sensors trainer a strain gage is used to measure the deflection of a flexible link. As the link bends, the resistance of the strain gage changes.

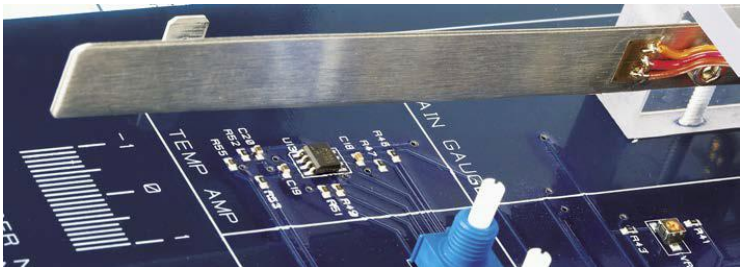


Figure 3.1: Strain gage measuring deflection of flexible link on QNET mechatronic sensors trainer deflection of flexible link on QNET mechatronic sensors trainer.

3.2 Strain Virtual Instrument

The virtual instrument used to collect data using the strain gage is shown in Figure 3.2. The virtual instrument used to calibrate strain data is shown in Figure 3.3.

The virtual instrument used to determine the natural frequency of the flexible link is shown in Figure 3.4.

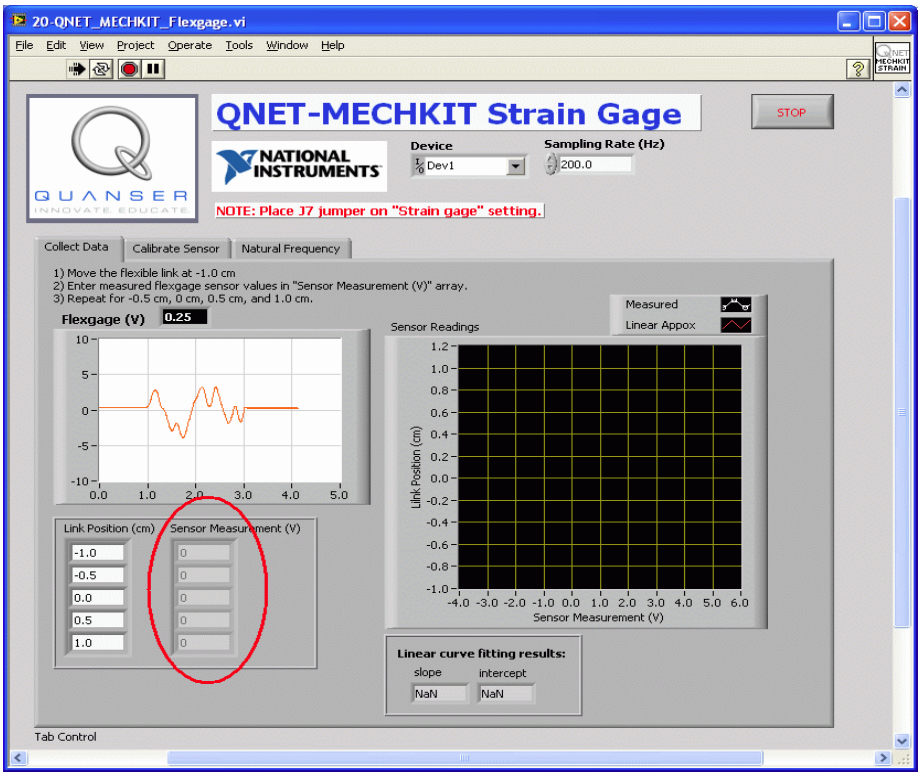


Figure 3.2: Collecting flexgage data

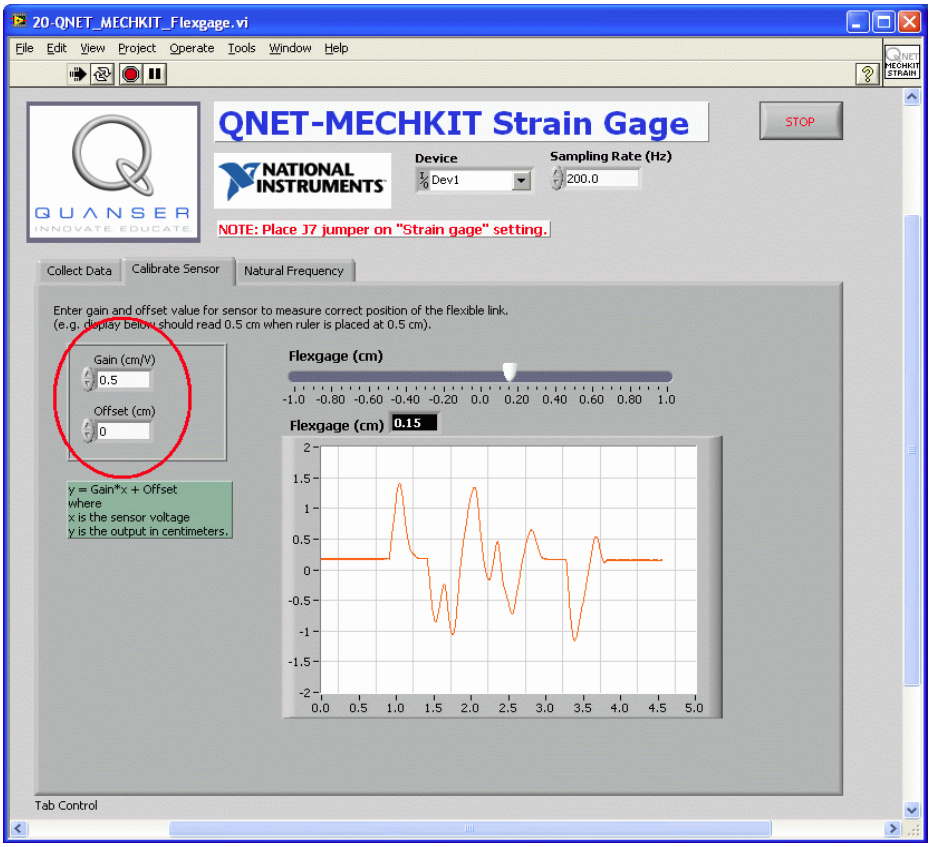


Figure 3.3: Calibrating the strain gage sensor

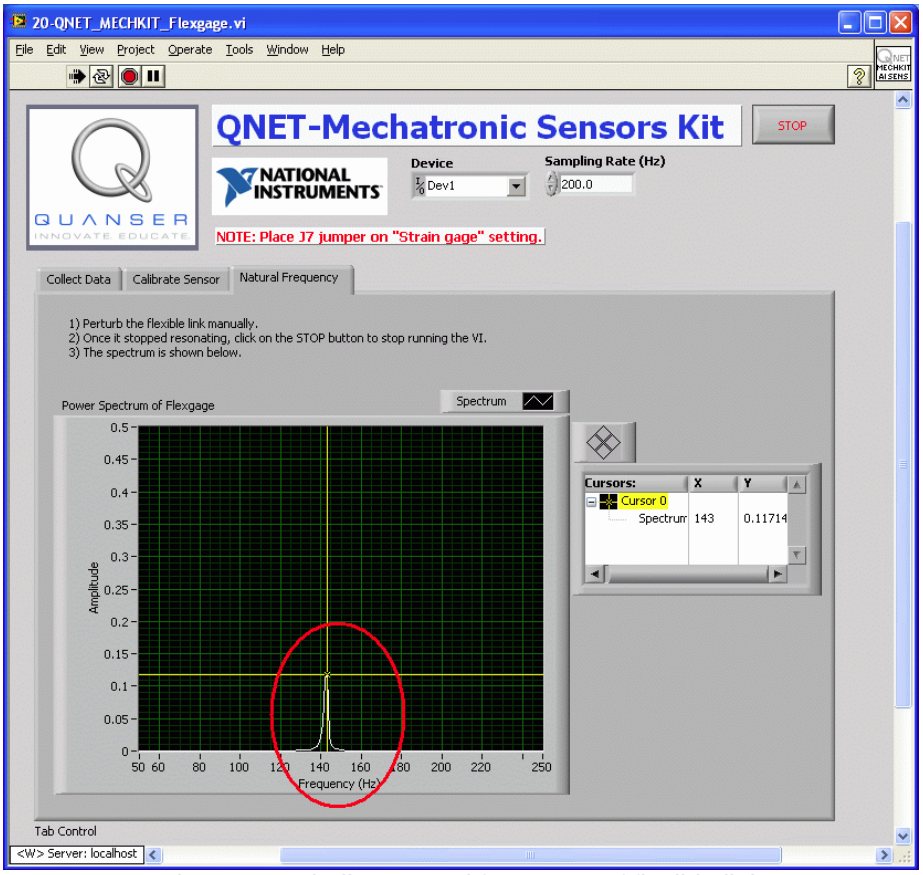


Figure 3.4: Finding natural frequency of flexible link

3.3 Lab 1: Collect Data

1. Ensure J7 is set to Strain Gage.
2. Open and configure the QNET MECHKIT Flexgage VI as described in section 15.2. Make sure the correct Device is chosen.
3. Run QNET MECHKIT Flexgage.vi
4. Move the flexible link to -1 cm.
5. Enter the strain gage voltage reading in the Sensor Measurement (V) array indicated in Figure 3.2).
6. Repeat for -0.5 cm, 0 cm, 0.5 cm, and 1.0 cm. A linear curve is automatically fitted to the data being entered and its slope and intercept are generated.
7. Enter the measured voltages and capture the Sensor Readings scope.
8. Click on Stop button to stop the VI.

Table 3.1 Strain gage results

Parameter	Value	Units	Notes
Sensor Measurement: at -1.0 cm		V	
Sensor Measurement: at -0.5 cm		V	
Sensor Measurement: at 0 cm		V	
Sensor Measurement: at 0.5 cm		V	
Sensor Measurement: at 1.0 cm		V	
Gain		cm/V	
Offset		cm	

3.4 Lab 2: Calibrate Sensor

1. Run the QNET MECHKIT Flexgage.vi
2. Select the Calibrate Sensor tab and enter the slope and intercept obtained in Section 3.3 into the Calibration Gain and Offset controls shown in Figure 3.3, below. When the link is moved, the slider indicator in the VI should match up with the actual location of the flexible link on the QNET module.
3. Enter the gain and offset obtained.
4. Click on Stop button to stop the VI.

3.5 Lab 3: Natural Frequency

1. Run the QNET MECHKIT Flexgage.vi
2. Select the Natural Frequency tab.
3. Manually perturb the flexible link and stop the VI when it stops resonating (after about 5 seconds). The spectrum should then load in the chart, as shown in Figure 3.4 (the value shown is incorrect).
4. Enter natural frequency found and capture the resulting power spectrum response. Hint: You can use the cursor to take measurements off the graph.
5. Click on Stop button to stop the VI.

3.6 Results

Table 3.2 Strain gage results summary

Parameter	Value	Units	Notes
Gain		cm/V	
Offset		cm	
Natural Frequency			

Experiment # S2

Pressure and Piezo Sensors

4. Pressure Sensor

4.1 Background

A pressure sensor is attached to the plunger on the QNET mechatronic board shown in Figure 4.1. This is a gage pressure sensor and its measurements are relative to the atmospheric pressure. The voltage signal generated is proportional to the amount of pressure in the vessel of the plunger. So as the plunger is pushed further, the air inside the vessel becomes more compressed and the reading increases. Pressure sensors can also be used to indirectly measure other values. For example, in the QNET mechatronics board the position of the plunger head is measured. It can also be used to measure the amount of volume in a reservoir or the altitude of an aerial vehicle.

Figure 4.1: Pressure sensor on QNET mechatronic sensors trainer

4.2 Pressure Virtual Instrument

The virtual instrument used to collect data using the strain gage is shown in Figure 3.2. The virtual instrument used to calibrate strain data is shown in Figure 3.3. The virtual instrument used to determine the natural frequency of the flexible link is shown in Figure 3.4.

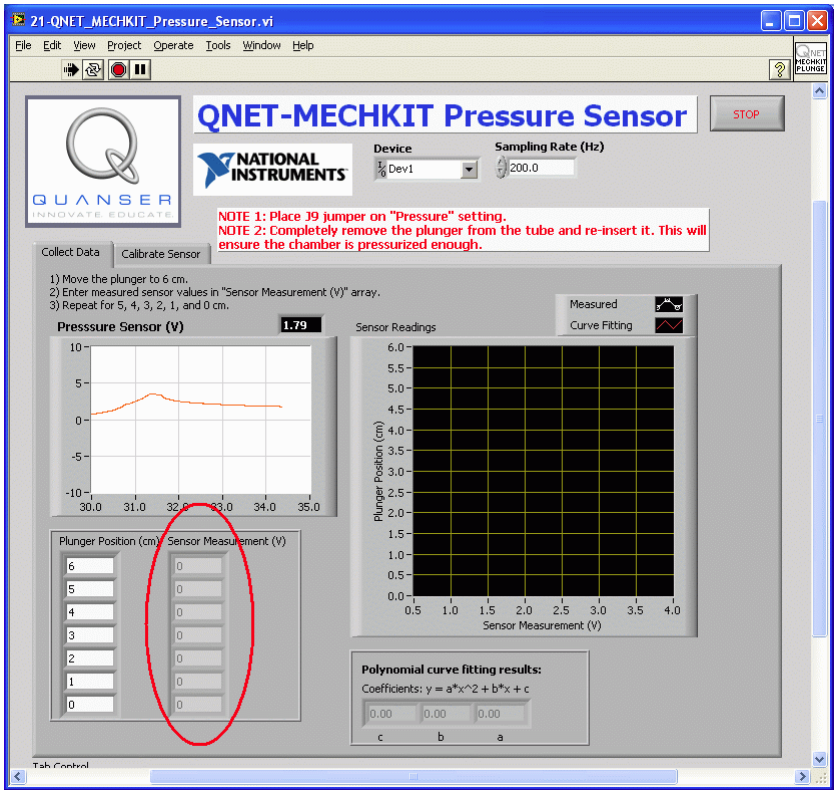


Figure 4.2: Collecting pressure data

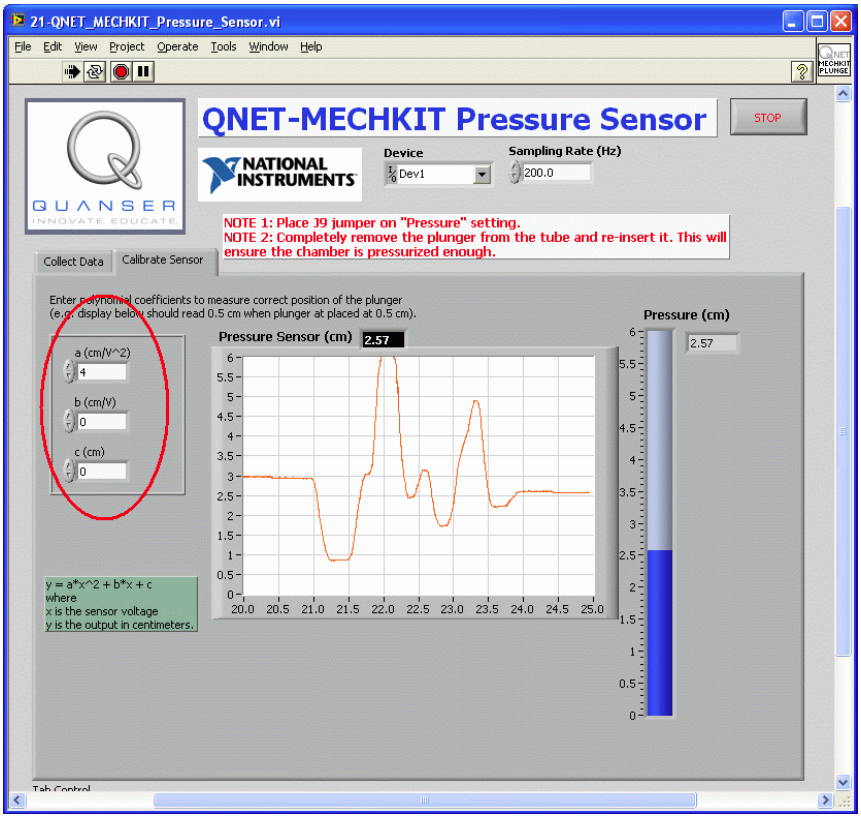


Figure 4.3: Calibrating the pressure sensor

4.3 Lab 1: Collect Data

1. Ensure *J9* is set to *Pressure*.
2. Open and configure the QNET MECHKIT Pressure VI as described in Section 15.3. Make sure the correct *Device* is chosen. Important: Completely remove the plunger from the tube and re-insert it. This will ensure the chamber is pressurized enough.
3. Run *QNET_MECHKIT_Pressure_Sensor.vi*
4. Push the plunger up to the 6 cm marked on the MECHKIT board and measure the resulting voltage using the *Pressure (V)* scope (or the digital display).
5. Enter the result in the *Sensor Measurement (V)* array, as indicated in Figure 4.2.
6. Repeat for when the plunger is at 5.0 cm, 4.0 cm, 3.0 cm, 2.0 cm, 1.0 cm, and 0 cm. The pressure sensor is quadratic. The coefficients for the second-order polynomial are generated and the fitted curve is automatically plotted.
7. Enter collected results and capture the Sensor Readings scope.
8. Click on Stop button to stop the VI.

Table 4.1 Pressure sensor results

Parameter	Value	Units	Notes
Sensor Measurement: at 6.0 cm		V	
Sensor Measurement: at 5.0 cm		V	
Sensor Measurement: at 4.0 cm		V	
Sensor Measurement: at 3.0 cm		V	
Sensor Measurement: at 2.0 cm		V	
Sensor Measurement: at 1.0 cm		V	
Sensor Measurement: at 0.0 cm		V	

4.4 Lab 2: Calibrate Sensor

1. Run the QNET MECHKIT Pressure Sensor.vi.
2. In the Calibrate Sensor tab, enter the polynomial coefficients, as illustrated in Figure 4.3, to measure correct position of the plunger. Verify that the sensor is reading properly, e.g. display should read 0.5 cm when plunger is placed at 0.5 cm.
3. Enter the a, b, and c, parameters used.
4. Click on Stop button to stop the VI.

4.6 Results

Table 4.2 Pressure sensor results summary

Parameter	Value	Units	Notes
a		cm/V^2	
b		cm/V	
c		Cm	

5. Piezo Sensor

5.1 Background

Piezo sensors measure vibration. The piezo sensor on the QNET-MECHKIT trainer, shown in Figure 5.1, is connected to a plastic band that has a brass disc weight at the end.

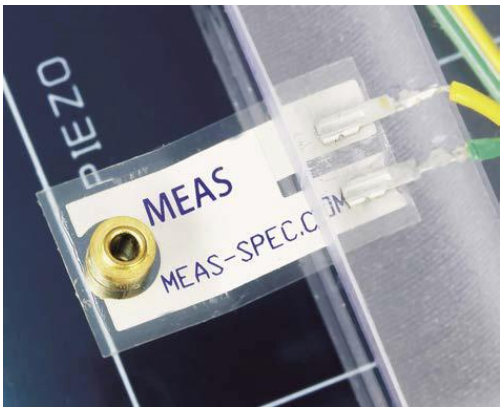


Figure 5.1: Piezo sensor on the QNET mechatronic sensors trainer

5.2 Lab 1: Data Analysis

1. Ensure J8 is set to Piezo.
2. Open and configure the QNET MECHKIT Piezo VI. Make sure the correct *Device* is chosen.
3. Run *QNET_MECHKIT_Piezo.vi*
4. Manually perturb the plastic band that is attached to the piezo sensor by flicking it and examine the response in the *Piezo (V)* scope.
5. Grab the end of the plastic band and move it slowly up and down. Examine the response.
6. Based on these two tests, what does the Piezo sensor measure? How is this different then a strain gage measurement? Capture a sample Piezo (V) scope response after it has been perturbed (by flicking it).
7. Click on Stop button to stop the VI.

5.3 Lab 2: Natural Frequency

1. Run the *QNET_MECHKIT_Piezo.vi*
2. Manually perturb the piezo sensor.
3. Capture the resulting power spectrum response and give the measured natural frequency. Hint: You can use the cursor to take measurements off the graph.
4. Click on Stop button to stop the VI.

Experiment # S3

Potentiometer and Infrared

6. Potentiometer

6.1 Background

Rotary potentiometers are absolute analog sensors used to measure angular position, such as a load shaft of a motor. They are great to obtain a unique position measurement. However, caution must be used as their signal is discontinuous. That is, after a few revolutions potentiometers will reset their signal back to zero. The potentiometer on the QNET MECHKIT board is shown in Figure 6.1.

Figure 6.1: Potentiometer knob on QNET mechatronic sensors trainer

6.2 Potentiometer Virtual Instrument

The virtual instrument used to collect data using the potentiometer is shown in Figure 6.2. The virtual instrument used to calibrate potentiometer data is shown in Figure 6.3.

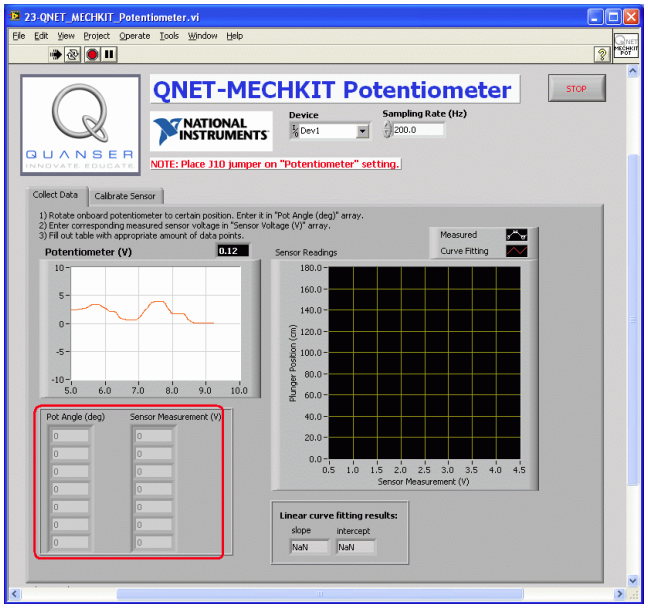


Figure 6.2: Collecting potentiometer data

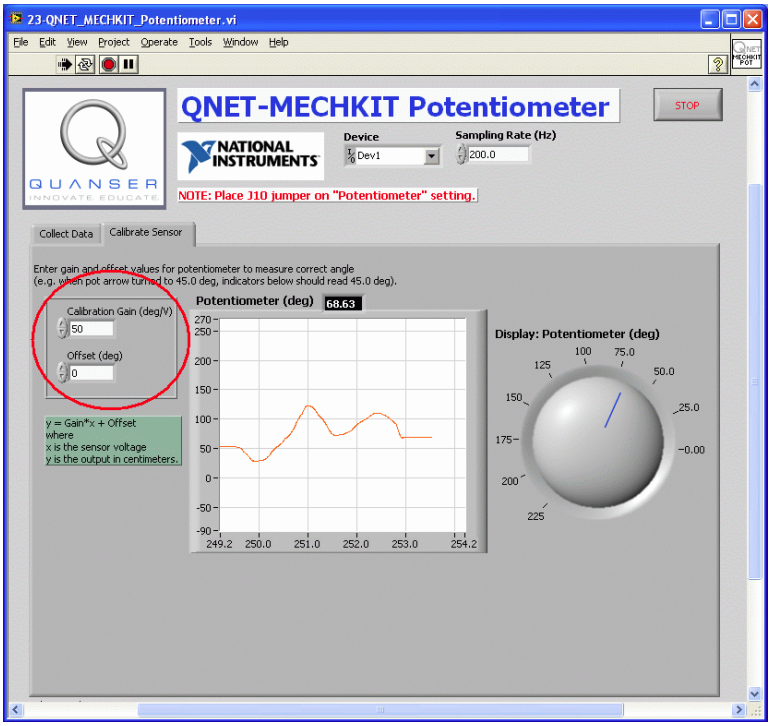


Figure 6.3: Calibrating the potentiometer

6.3 Lab 1: Collect Data

1. Ensure J10 is set to POT.
2. Open and configure the QNET MECHKIT Potentiometer VI as described in Section 15.5. Make sure the correct *Device* is chosen.
3. Run *QNET_MECHKIT_Potentiometer.vi*
4. Rotate the arrowhead of the potentiometer to a certain position, e.g. 45 degrees.
5. Enter the position in the Pot Angle (deg) array, as indicated in Figure 6.2.
6. Enter corresponding measured sensor voltage in Sensor Measurement (V) array (shown in Figure 6.2).
7. Fill out table with an appropriate amount of data points. Notice that as the measured potentiometer readings are entered, a curve is automatically generated to fit the data. The slope and intercept of this line is generated as well.
8. Enter the collected data and capture the Sensor Reading chart.
9. Click on Stop button to stop the VI.

Table 6.1 Strain gage results

Parameter	Value	Units	Notes
Sensor Measurement: at 0 deg		V	
Sensor Measurement: at 45 deg		V	
Sensor Measurement: at 90 deg		V	
Sensor Measurement: at 135 deg		V	
Sensor Measurement: at 180 deg		V	
Gain		deg/V	
Offset		deg	

6.4 Lab 2: Calibrate Sensor

1. Run *QNET_MECHKIT_Potentiometer.vi*
2. In the Calibrate Sensor tab, set the Gain and Offset controls, as indicated in Figure 6.3, to values such that the potentiometer measures the correct angle. Verify that the sensor is reading properly, e.g. when pot arrow is turned to 45.0 deg, the Display: Potentiometer (deg) knob indicator should read 45.0 degrees.
3. Enter Gain and Offset values used.
4. Click on Stop button to stop the VI.

6.6 Results

Table 6.2 Potentiometer results summary

Parameter	Value	Units	Notes
Gain		deg/V	
Offset		deg	

7. Infrared

7.1 Background

Infrared (IR) sensors are widely used in robots, automotive systems, and various other applications that require an accurate, medium-range non-contact position measurement. An IR sensor is typically composed of an infrared emitting diode (IRED), a position sensing detector (PSD), and a signal processing circuit. It outputs a voltage the correlates to the distance of the remote target. The infrared distance measuring sensor on the QNET MECHKIT board is shown in Figure 7.1. Infrared-based distance sensors typically have a smaller maximum range than sonar but the resolution is better.

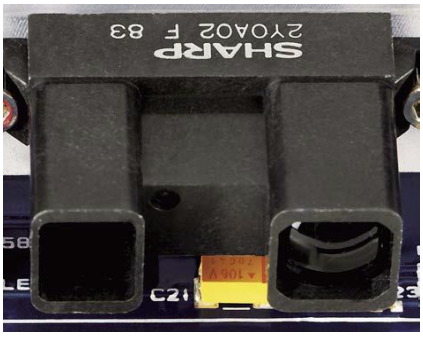


Figure 7.1: IR sensor on QNET mechatronic sensors trainer

7.2 Infrared Virtual Instrument

The virtual instrument used to collect data using the IR sensor is shown in Figure 7.2. The virtual instrument used to calibrate IR range data is shown in Figure 7.3.

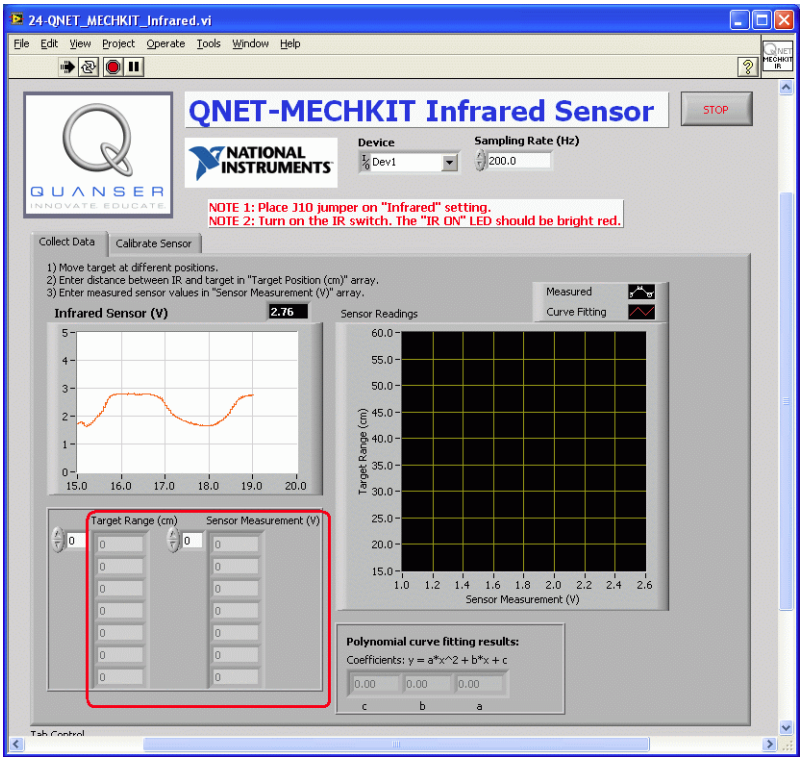


Figure 7.2: Collecting IR data

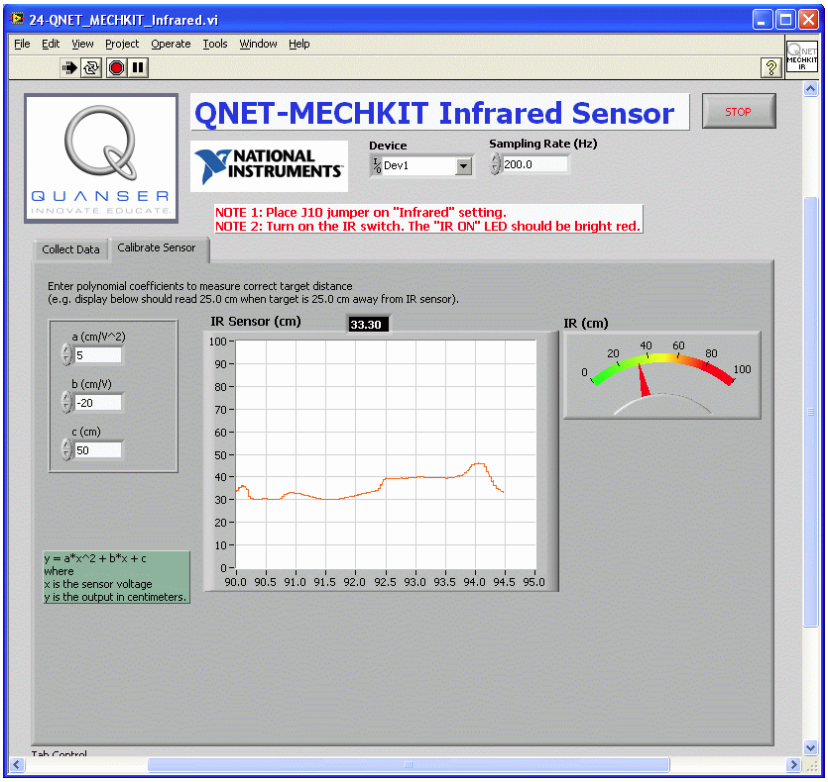


Figure 7.3: Calibrating the IR sensor

7.3 Lab 1: Collect Data

1. Ensure J10 is set to Infrared.

2. Open and configure the QNET MECHKIT Infrared VI as described in Section 15.6. Make sure the correct *Device* is chosen.
3. Run *QNET_MECHKIT_Infrared.vi*
4. Turn ON the IR switch to enable the Infrared sensor. The IR ON LED should be lit bright red. **Important**: Make sure you turn OFF the IR switch when the experiment is over. When active, the infrared sensor tends to generate noise in other sensor measurements.
5. Get a target, such as a sturdy piece of cardboard, that is at least 10 by 10 cm2 with a reflective colour like white or yellow.
6. Begin with the target close to the IR sensor and slowly move it away.
7. Once its range of operation is found, enter the distance between the target and the IR sensor in the Target Range (cm) array, as shown in Figure 7.2.
8. Enter the corresponding measured voltage from the IR sensor in the Sensor Measurement (V) array, as shown in Figure 7.2.
9. Repeat for different target positions. The IR sensor is quadratic. As the measurements are entered, the coefficients for the second-order polynomial are generated and the fitted curve is automatically plotted.
10. Record your distance and voltage observations and capture the corresponding Sensor Readings scope.

Table 7.1 Infrared sensor results

Parameter	Value	Units	Notes
Sensor Measurement: at 17 cm		V	
Sensor Measurement: at 22 cm		V	
Sensor Measurement: at 27 cm		V	
Sensor Measurement: at 32 cm		V	
Sensor Measurement: at 37 cm		V	
Sensor Measurement: at 42 cm		V	
Sensor Measurement: at 47 cm		V	
Sensor Measurement: at 52 cm		V	
Sensor Measurement: at 57 cm		V	

7.4 Lab 2: Calibrate Sensor

1. Run *QNET_MECHKIT_Infrared.vi*
2. In the Calibrate Sensor tab, enter the polynomial coefficients to correctly measure the distance of the target. Check that it is measuring correctly, e.g. when target is 25.0 cm away, the display should read 25.0 cm.
3. Enter the a, b and c values used in Table 7.2.
4. Click on Stop button to stop the VI.

7.5 Results

Table 7.2 Infrared sensor results summary

Parameter	Value	Units	Notes
a		cm/V^2	
b		cm/V	
c		Cm	

Labwork Report

Experiment :

Name :

ID :

Objective and summary of the procedure:

Results:

Analysis and Conclusion: