**INTERNSHIP REPORT**

**Intech Systems Chennai Pvt. Ltd.**

**Presented by**

Vennilah P

**BONAFIDE**

Certified that this internship report is the bonafide work of

Ms. VENNILAH .P who carried out the internship under my supervision.

**Mrs. S. Divya**

HR of Intech Systems Chennai

**ABSTRACT**

This report details the internship undertaken at Intech Systems Chennai, focusing on PLC and AI VISION. The primary objective of the internship was to have clear understanding about AI VISION and PLC along with Ladder Logic Programming. Over the course of June 11th 2024 to July 11th 2024, I was involved in gaining knowledge about the domains available in the company.

Key activities included PLC programming based on real life situations and real life applications , Python programming related to AI vision. The internship provided practical experience in handling the PLC of various brands and programming in python to access the web cam and complete control over coding as per our daily life needs, and most of all integrating both PLC and AI VISION.

The results of my work included a project we are working on based on the knowledge gained at the intern location and the inclusion of PLC and AI vision. The experience also enhanced my understanding of PLC and Python programming for AI, which I will carry forward into my future career.

In conclusion, this internship was a valuable opportunity to apply theoretical knowledge in a practical setting, develop professional skills, and contribute to meaningful projects within Intech Systems Chennai.

**TABLE OF CONTENTS**

**S.No TITLE Pg.No.**

1. About Intech 6
2. Concepts learnt 9
3. PLC and its types 10
4. Ladder logic Program 16
5. Relays vs Switches 17
6. Example programs 20
7. AI VISION 26
8. Programs using python (AI VISION) 31
9. Why PLC when there is AI? 39
10. Certification 41

**ABOUT INTECH**

**SOME OF THE PRODUCTS:**

AUTOMATION SYSTEMS: Simple Machine Automation to Complex Process and Total Factory Automation Solutions; All under One Roof. Presenting a wide array of Simple to Advanced Automation products to cater to all your Automation Requirements.

****

Safety, vision and IDENT: A strong product portfolio and application strength in Safety, Vision, RFID, Barcode & Laser Marking enables us to be the most popular Safety, Laser Marking and Advanced Vision Solution provider in the market today.



CONTROL COMPONENTS: Wide range of Indicators, Meters and Controllers ranging from basic function type to fully featured PID & Ramp Soak type. Intech is your one stop solution for all your Indication & Control needs.



SENSING SOLUTION: From basic proximity sensors to high precision spot and profile sensors with 2µm accuracy we´re here to cover every sensing requirement of the industry. Leave your sensing worries to Intech for the most appropriate solution.



**BRANDS:**

* OMRON
* MITSUBISHI
* DELTA
* AUTONICS
* ORIENTALMOTOR
* MICROSCAN
* IPA
* TECHNOMECH
* HONEYWELL

**SOLUTIONS:**

* IOT and traceability
* Project and panel Engineering
* Application case studies
* Solutions by Industry
* Solution by Process
* Solution by Products

**CONCEPTS LEARNT**

The concepts covered in the course of this one month of internship is

* PLC
* Ladder Logic Programming
* AI vision
* Integrating PLC and AI vision
* Basics of Robotics

These topics were effectively taught and also the engagement of an individual in a workspace , working and interacting with peers, lively interaction with new faces were encountered in the course of one month.

**PLC AND ITS TYPES**

**WHAT IS PLC?**

A Programmable Logic Controller (PLC) is a specialized digital computer extensively used in the automation of electromechanical processes. These processes include control of machinery on factory assembly lines, amusement rides, or lighting fixtures, among others. PLCs are designed to manage complex industrial processes and are pivotal in the industrial automation landscape due to their robustness, flexibility, and real-time operation capabilities.

**Key Features of PLC:**

1. Robustness:

PLCs are engineered to operate reliably under harsh industrial conditions such as extreme temperatures, high humidity, electrical noise, and significant vibration. This robustness ensures that they can continue to function optimally in environments where traditional computers would fail.

2. Programmable:

The programmable nature of PLCs allows engineers to configure them to perform a wide range of tasks. The programming is typically done using specialized languages such as Ladder Logic, Function Block Diagram (FBD), and Structured Text (ST). Ladder Logic, in particular, is user-friendly for those familiar with electrical control diagrams, facilitating ease of programming and troubleshooting.

3. Input/Output (I/O) Modules:

PLCs are equipped with numerous I/O modules that allow them to interface with various input devices (e.g., sensors, switches) and output devices (e.g., motors, valves). This modularity enables PLCs to monitor and control a multitude of devices, providing the necessary flexibility for diverse applications.

4. Real-time Operation:

PLCs are designed for real-time applications, meaning they can process inputs and execute outputs almost instantaneously. This capability is crucial for maintaining the precise control needed in automated systems, ensuring timely responses to changing conditions.

**Applications of PLC:**

1. Manufacturing:

In manufacturing, PLCs are used to control assembly lines, robotic devices, and machinery functions. They ensure the synchronization and precise operation of various processes, enhancing productivity and reducing human error.

2. Automotive Industry:

PLCs play a significant role in the automotive industry, managing processes such as welding, painting, and assembling car parts. Their precise control ensures high-quality production and efficient use of resources.

3. Energy Sector:

PLCs are essential in the management of power generation, transmission, and distribution systems. They help in monitoring and controlling the complex operations of power plants, substations, and grid networks, ensuring reliable and efficient energy supply.

4. Building Automation:

In building automation, PLCs control systems like HVAC (Heating, Ventilation, and Air Conditioning), lighting, and security systems. They provide integrated control solutions that enhance energy efficiency, safety, and comfort in commercial and residential buildings.

5. Utilities:

PLCs are widely used in water and wastewater treatment plants. They automate various processes such as filtration, chemical dosing, and pumping, ensuring efficient operation and compliance with environmental regulations.

**TYPES OF PLC:**

1. Compact PLC**:** A Compact PLC, also known as a fixed PLC, has a fixed number of I/O modules built into the unit. These are typically used for smaller applications where the number of input and output points is limited and fixed. They are cost-effective and simple to set up, making them ideal for small-scale automation tasks.

****

1. Modular PLC**:** Modular PLCs allow users to add or remove I/O modules as needed. These systems consist of a central processor and multiple I/O modules that can be customized based on the application requirements. They offer greater flexibility and scalability compared to compact PLCs, making them suitable for medium to large-scale applications.

****

1. Rack-Mounted PLC: Rack-mounted PLCs are similar to modular PLCs but are designed to fit into standard industrial racks. They consist of multiple modules that can be easily added or replaced without disrupting the entire system. These PLCs are used in large and complex industrial environments where space efficiency and modularity are crucial.

****

1. PLC with Integrated HMI (Human-Machine Interface)**:** These PLCs combine the functionalities of a PLC with an integrated HMI, providing both control and user interface capabilities in a single unit. This integration simplifies the system architecture and is useful in applications where space is limited and a user interface is necessary for monitoring and control.

****

**LADDER LOGIC PROGRAM**

Ladder Logic is a graphical programming language used to develop software for Programmable Logic Controllers (PLCs). Resembling electrical relay logic diagrams, it uses symbols representing devices like relays, timers, and counters. Ladder diagrams consist of "rungs" that represent operations, which execute sequentially from top to bottom. Each rung comprises input conditions and output actions, mimicking electrical circuits. Easy to understand and implement, Ladder Logic is widely used in industrial automation for controlling machinery, processes, and systems. Its simplicity makes it accessible to engineers and technicians familiar with electrical control systems, facilitating efficient programming and troubleshooting.

**RELAYS Vs. SWITCHES**

Relays and switches are both fundamental components used in electrical and electronic circuits to control the flow of current, but they serve different purposes and operate differently. Here’s a comparison to differentiate them:

**Relays:**

Function: A relay is an electrically operated switch. It uses an electromagnetic coil to open or close contacts in a separate circuit.

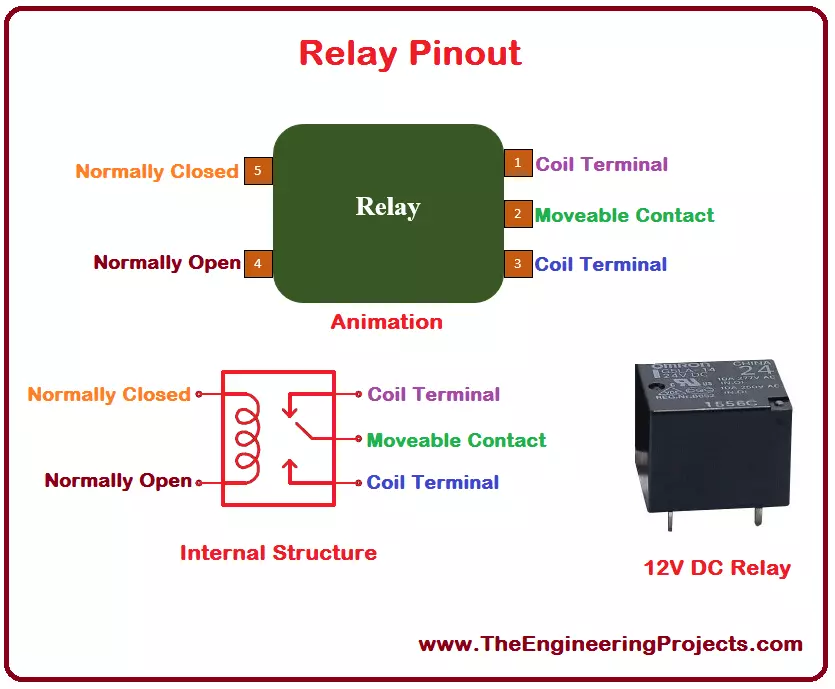
Operation: Relays operate by applying an electrical current to the coil, creating a magnetic field that moves a set of contacts to open or close a circuit.

Control: They can control high-power devices with a low-power signal, providing electrical isolation between the control and the controlled circuits.

Applications: Used in automation systems, motor control, protection systems, and where remote or automatic switching is needed.

Types: Electromechanical relays, solid-state relays, reed relays, and time delay relays.

Complexity: More complex than switches, often used in scenarios requiring multiple circuits or higher power levels.



**Switches:**

Function: A switch is a manual device that opens or closes a circuit, allowing or interrupting the flow of electricity.

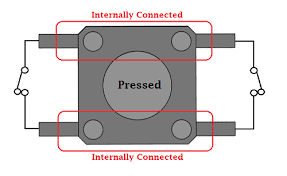
Operation: Operated manually by a user or mechanically, a switch simply connects or disconnects the electrical path.

Control: Directly controlled by physical action, such as pressing, flipping, or rotating.

Applications: Used in everyday devices like light fixtures, household appliances, and electronic gadgets for on/off control.

Types: Toggle switches, push-button switches, rocker switches, slide switches, and rotary switches.

Complexity: Simpler than relays, suitable for straightforward applications where manual control is sufficient.

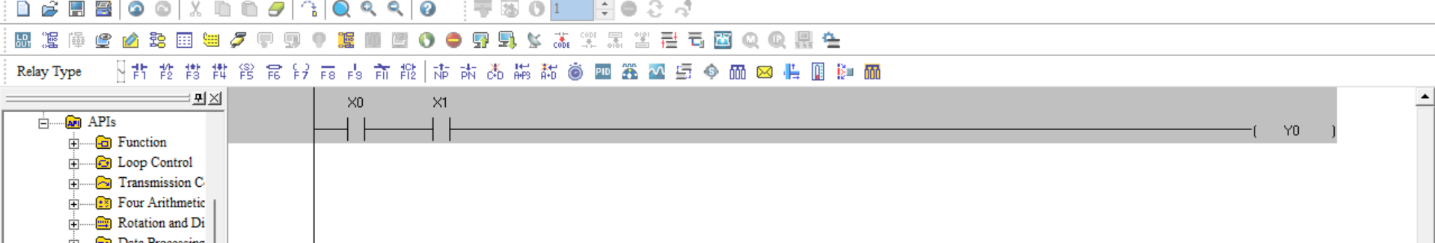


**EXAMPLE PLC PROGRAMS**

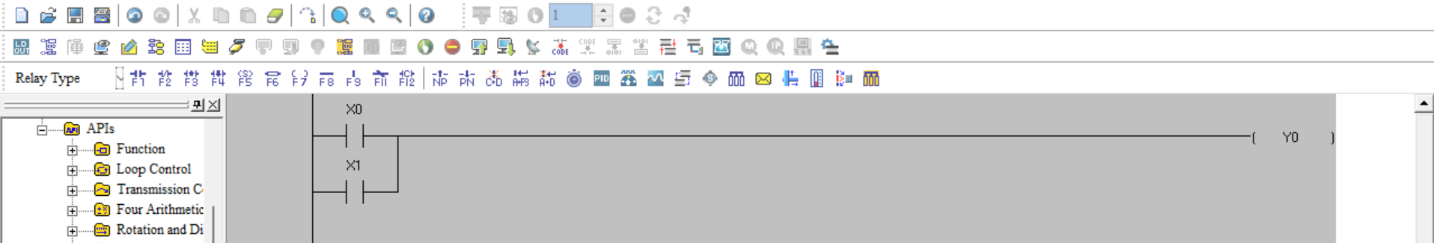
The suggested open software was Delta WPL software which offers almost all the accessibilities as that of the normal paid software. More than 20 ladder logic programs were given to solve and a few of them are listed below.

**Q1.** Give the basic Ladder Logic program for the logic gates such as AND , OR, EXOR, EXNOR, NAND , NOR

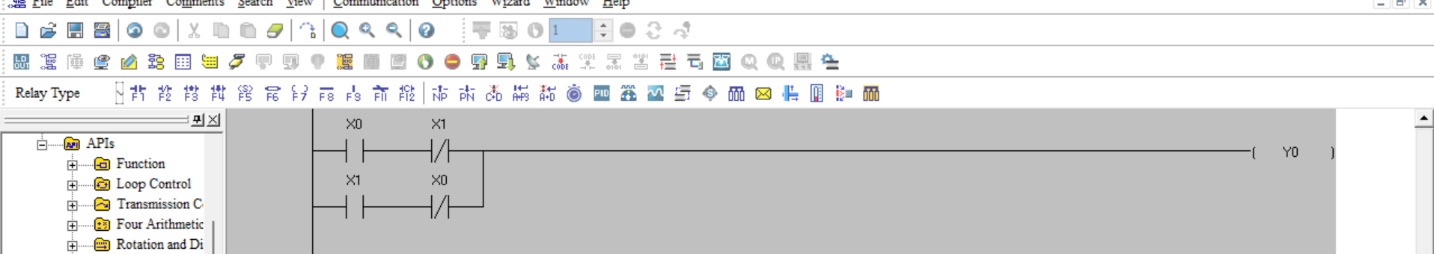
**AND**



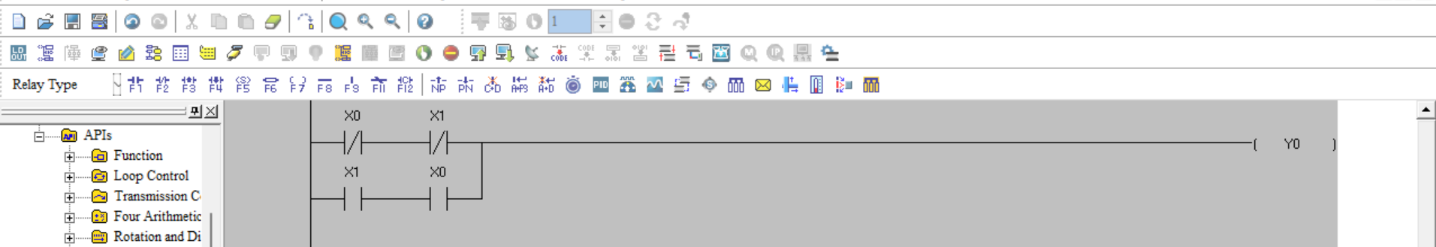
**OR**



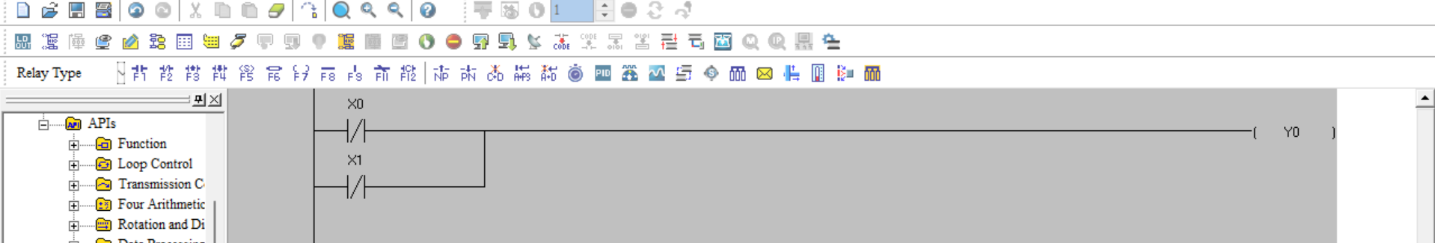
**EXOR**



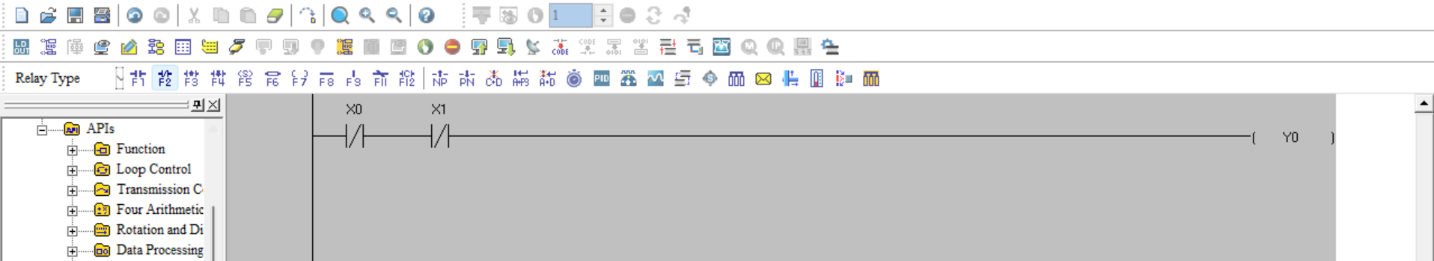
**EXNOR**

****

**NAND**

****

**NOR**

****

**Timer:**

A timer in a Programmable Logic Controller (PLC) is a control mechanism that counts time intervals and triggers actions after a set period. Timers are crucial for operations that require delays, timing sequences, or precise control over the duration of activities. They are widely used in automation processes to ensure proper sequencing and timing of operations.

**Types of Timers in PLC:**

**ON-Delay Timer (TON):**

* Starts counting when the input signal is received.
* After the preset time elapses, the output is activated.

**OFF-Delay Timer (TOF):**

* Starts counting when the input signal is removed.
* After the preset time elapses, the output is deactivated.

**Note:** We know that 1000ms = 1s, hence in a timer we give 100ms for 10 s. A typical timer is given by the notation, ----(T0 K100)---- . This particular timer runs for 10 seconds.

**COUNTER:**

A counter in a Programmable Logic Controller (PLC) is a device that counts occurrences of input signals, such as pulses or events. Counters are used in various industrial applications to keep track of repetitive operations, such as counting the number of items produced on a conveyor belt or the number of machine cycles.

**Types of Counters in PLC:**

**Up Counter (CTU):**

* Counts upwards each time an input signal is received.
* The counter increases by one for each pulse until it reaches a preset value.

**Down Counter (CTD):**

* Counts downwards each time an input signal is received.
* The counter decreases by one for each pulse until it reaches zero.

**Up/Down Counter (CTUD):**

* Can count both upwards and downwards.
* Uses two inputs: one for counting up and one for counting down.

**Q2.** Give the ladder logic program for the following concept mentioned below,

Once a button is pressed , after a five second delay, the forward switch is ON and this is sensed by the forward reed switch . Once it is detected by the forward reed switch, the forward switch is OFF and it stays in the same position for a 10 second delay and then the reverse switch is turned ON and turned OFF , this must lead to a count of 1. As the cycle gets completed, the count must be increased by one .

Let us consider

X1=start push button

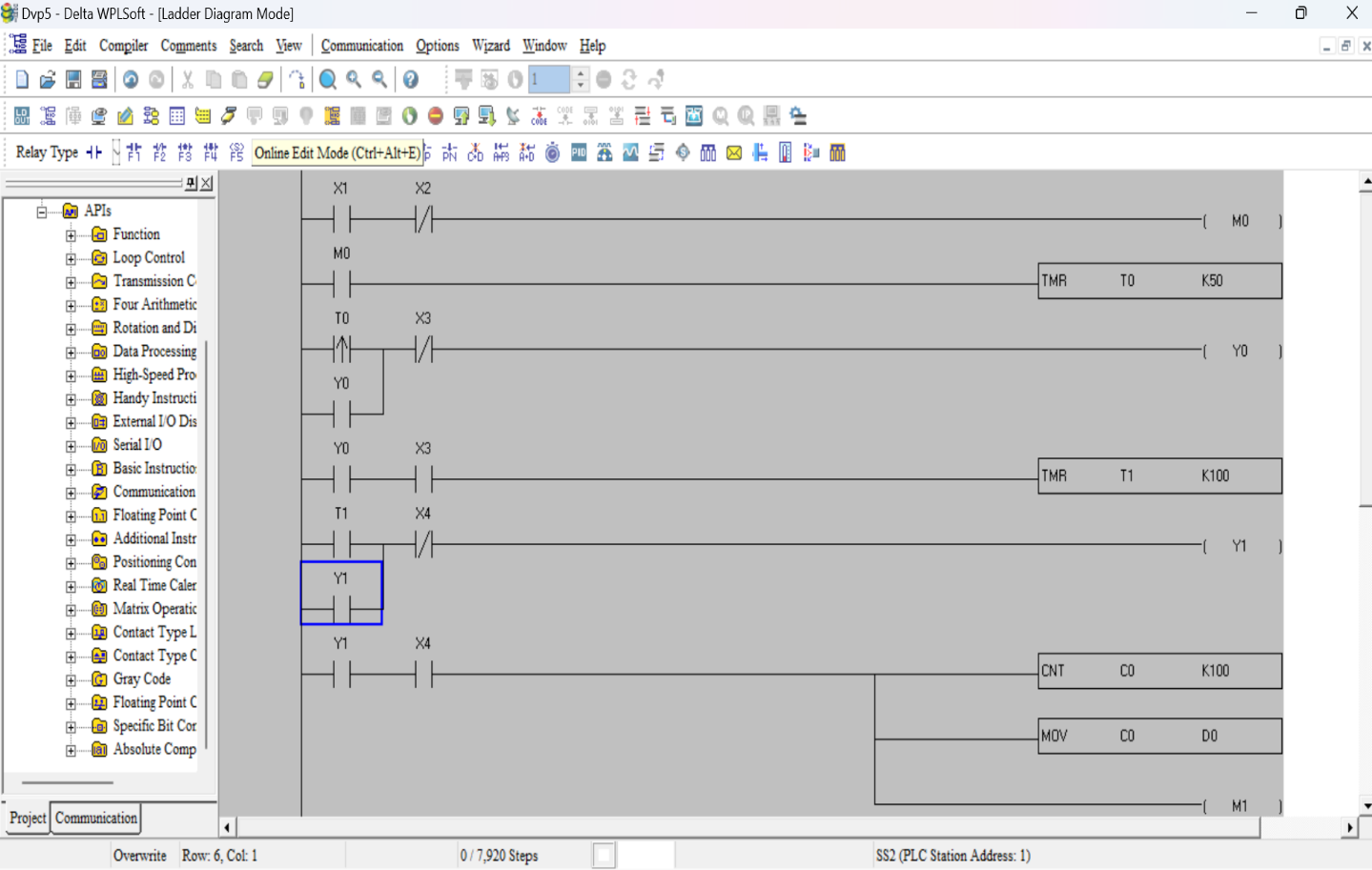
X2=stop push button

X3=forward reed switch

X4=reverse switch

Y0=cylinder forward

Y1=cylinder between



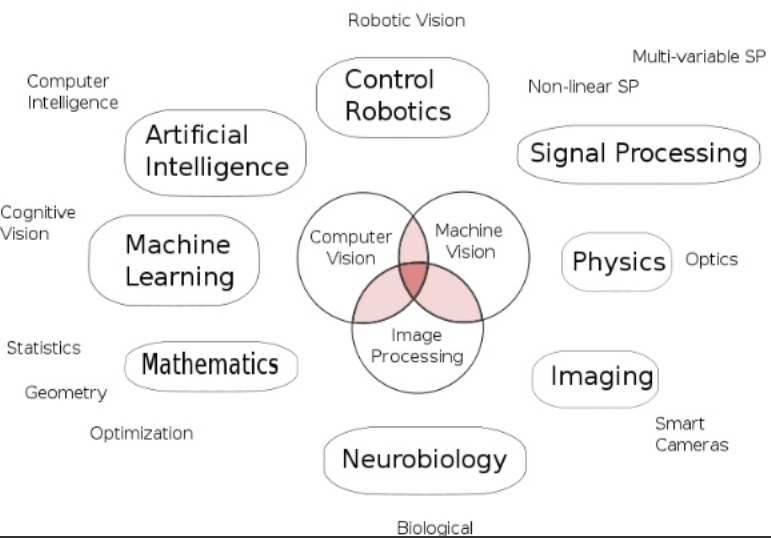
**AI VISION**

**INTRODUCTION:**

Machine vision (Computer Vision) is the technology to replace or complement manual visual inspections and measurements with digital cameras and image processing. The technology is used in a variety of different industries to automate the production, increase production speed and yield, and to improve product quality.

Machine vision in operation can be described by a four-step flow:

* Imaging: Take an image.
* Processing and analysis: Analyse the image to obtain a result.
* Communication: Send result to the control system of the process.
* Action: Take action depending on the vision system's result.



**INDUSTRIES SERVING:**

Machine vision applications can also be categorized according to industry type, for example:

* Automotive
* Electronics
* Food
* Logistics
* Manufacturing
* Robotics
* Packaging
* Pharmaceutical
* Steel and mining
* Wood

**APPLICATION TYPES:**

**Inspect**: In inspection applications the purpose of the vision system is to validate certain features, for example presence or absence of a correct label on a bottle, screws in an assembly, chocolates in a box, or defects.

**Identify**: In an identification application the vision system reads various codes (1D,2D Barcodes) and alphanumeric characters (text and numbers).

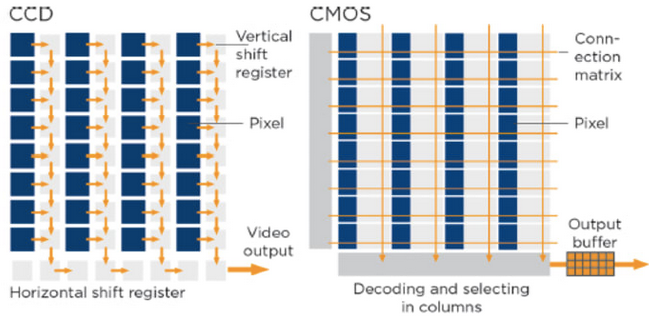
**Locate**: In locating applications, the purpose of the vision system is to find the object and report its position and orientation.

**Measure**: In measurement applications the purpose of the vision system is to measure physical dimensions of the object. Examples of physical dimensions are distance, diameter, curvature, area, height, and volume.

**IMAGER:**

Sensor Types

1. CCD (Charge-Coupled Device)
2. CMOS (Complementary Metal Oxide Semiconductor)



**OPTICS:**

The lens focuses the light that enters the camera in a way that creates a sharp image. Some of the important terms are,

* Field of View
* Focus
* Focal length
* Angle of view
* Aperture and F-stop
* Depth of field

**PROCESSING AND ANALYSIS:**

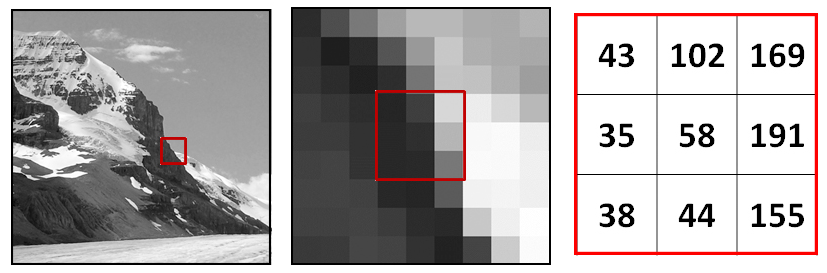
First of all the image is captured using different lenses as per the client's request. After the image has been grabbed, the next step is image analysis. This is where the desired features are extracted automatically by algorithms (Pixel Counting, Pixel Matching, Edge Detection, Calibration etc) and conclusions are drawn. A feature is the general term for information in an image. Algorithms are also referred to as tools or functions. Sometimes the image needs pre-processing (digital filters) before the feature extraction.

**Camera Programming:**

* Ready-to-use software - FZ\_FH Software, AutoVision, Insight Explorer
* Generic environment - Microsoft Visual Studio, OpenCV, Python, C++, VB

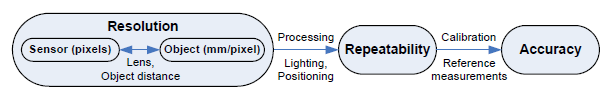
**Region of Interest:**

A ROI (Region of Interest) is a selected area of concern within an image to restrict the area of analysis and to allow for different analyses in different areas of the image. An image can contain any number of ROIs.



**Cycle time:**

Vision systems that operate in automated production lines often need to be fast. The speed aspect of a vision system’s performance is defined by its cycle time. The concept can be divided into subcategories, as illustrated by the flow diagram.



**PROGRAMS USING AI VISION (PYTHON)**

Python programming to access both the web cam of the system as well as multiple cameras are taught. Packages such as pandas, numpy , cv2 were downloaded prior to this. Couple of programs are stated below as example,

To capture the image of using webcam , the program is given as,

import cv2

import time

# Initialize the webcam

cap = cv2.VideoCapture(0)

while True:

    # Capture frame-by-frame

    ret, frame = cap.read()

# Display the captured frame

    cv2.imshow('Webcam', frame)

# Wait for keypress

    key = cv2.waitKey(1)

# Press 's' to save the image

    if key == ord('s'):

        # Generate a unique filename based on timestamp

        timestamp = time.strftime("%Y%m%d-%H%M%S")

        filename = f"captured\_image\_{timestamp}.jpg"

# Save the image

        cv2.imwrite(filename, frame)

        print(f"Image saved as {filename}")

    # Press 'q' to quit

    elif key == ord('q'):

        break

# Release the webcam and close all windows

cap.release()

cv2.destroyAllWindows()

To send a mail automatically to someone when the code is executed,

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

# Email configuration

sender\_email = "pvennilah@gmail.com"

receiver\_email = "nandineraja@gmail.com"

password = "anila0506"

# Create a multipart message

message = MIMEMultipart()

message["From"] = sender\_email

message["To"] = receiver\_email

message["Subject"] = "Test Email"

# Add body to email

body = "hoi!!! Is this nanuUuUuU????"

message.attach(MIMEText(body, "plain"))

# Connect to SMTP server

with smtplib.SMTP\_SSL("smtp.gmail.com", 465) as server:

    server.login(sender\_email, password)

    server.sendmail(sender\_email, receiver\_email, message.as\_string())

print("Email sent successfully!")

To turn on plc,

from pymodbus.client import ModbusTcpClient  
from pymodbus.exceptions import ModbusException  
import time  
  
# Replace with your PLC's IP address and port  
PLC\_IP = '192.168.1.5'  
PLC\_PORT = 502  
DATA\_REGISTER = 1280 # Address of the first data register (if needed for reading)  
  
def write\_coil(client, address, value):  
 *""" Write a value (True/False) to a specific coil address """* try:  
 result = client.write\_coil(address, value, unit=1)  
 if result.isError():  
 print(f"Error writing to coil {address}: {result}")  
 else:  
 print(f"Successfully wrote {value} to coil {address}")  
 except ModbusException as e:  
 print(f"Modbus exception: {e}")  
  
def read\_coil(client, address):  
 *""" Read a value from a specific coil address """* try:  
 result = client.read\_coils(address, count=1, unit=1)  
 if result.isError():  
 print(f"Error reading coil {address}: {result}")  
 else:  
 print(f"Value at coil {address}: {result.bits[0]}")  
 return result.bits[0]  
 except ModbusException as e:  
 print(f"Modbus exception: {e}")  
  
def main():  
 # Create a Modbus TCP client  
 client = ModbusTcpClient(PLC\_IP, port=PLC\_PORT)  
  
 # Connect to the PLC  
 if not client.connect():  
 print(f"Failed to connect to PLC at {PLC\_IP}:{PLC\_PORT}")  
 return  
  
 try:  
 # Turn on the first output (assuming writing True turns it on)  
 write\_coil(client, DATA\_REGISTER, True)  
 time.sleep(1) # Wait for a moment to ensure the coil is written  
  
 # Read the coil to confirm it's turned on  
 on\_value = read\_coil(client, DATA\_REGISTER)  
 if on\_value:  
 print("The output is ON")  
 else:  
 print("Failed to turn ON the output")  
  
 time.sleep(2) # Keep the output on for 2 seconds  
  
 # Turn off the first output (assuming writing False turns it off)  
 write\_coil(client, DATA\_REGISTER, False)  
 time.sleep(1) # Wait for a moment to ensure the coil is written  
  
 # Read the coil to confirm it's turned off  
 off\_value = read\_coil(client, DATA\_REGISTER)  
 if not off\_value:  
 print("The output is OFF")  
 else:  
 print("Failed to turn OFF the output")  
 finally:  
 # Close the connection  
 client.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

To integrate PLC with vision to count the occurrence of the particular colour. The PLC output is triggered once the colour is encountered and is counted,

import cv2  
import numpy as np  
from pymodbus.client import ModbusTcpClient  
from pymodbus.exceptions import ModbusException  
import time  
import threading  
  
# Replace with your PLC's IP address and port  
PLC\_IP = '192.168.1.5'  
PLC\_PORT = 502  
DATA\_COIL =1280 # Address of the coil to control (adjust as per your PLC configuration)  
  
def write\_coil(client, address, value):  
 *""" Write a value (True/False) to a specific coil address """* try:  
 result = client.write\_coil(address, value, unit=1)  
 if result.isError():  
 print(f"Error writing to coil {address}: {result}")  
 else:  
 print(f"Successfully wrote {value} to coil {address}")  
 except ModbusException as e:  
 print(f"Modbus exception: {e}")  
  
def detect\_baby\_pink\_and\_trigger():  
 # Create a Modbus TCP client  
 client = ModbusTcpClient(PLC\_IP, port=PLC\_PORT)  
  
 # Connect to the PLC  
 if not client.connect():  
 print(f"Failed to connect to PLC at {PLC\_IP}:{PLC\_PORT}")  
 return  
  
 # Initialize webcam  
 cap = cv2.VideoCapture(0) # Adjust 0 if using multiple webcams  
  
 # Flag to control output duration  
 output\_active = False  
  
 def turn\_off\_output():  
 nonlocal output\_active  
 time.sleep(2) # Keep the output on for 2 seconds  
 write\_coil(client, DATA\_COIL, False)  
 output\_active = False  
  
 try:  
 while True:  
 ret, frame = cap.read()  
 if not ret:  
 print("Failed to capture frame from webcam")  
 break  
  
 # Convert frame to HSV color space  
 hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  
  
 # Define range of baby pink color in HSV  
 lower\_pink = np.array([150, 50, 150])  
 upper\_pink = np.array([170, 255, 255])  
  
 # Threshold the HSV image to get only baby pink colors  
 mask = cv2.inRange(hsv, lower\_pink, upper\_pink)  
  
 # Find contours of detected objects  
 contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
  
 # Draw bounding box around detected objects and label it  
 for contour in contours:  
 x, y, w, h = cv2.boundingRect(contour)  
 cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2) # Green rectangle  
 cv2.putText(frame, "HERE IS BABY PINK!!!", (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)  
  
 # Count non-zero pixels in the mask  
 if cv2.countNonZero(mask) > 1000: # Adjust threshold as needed  
 if not output\_active:  
 write\_coil(client, DATA\_COIL, True)  
 output\_active = True  
 # Start a timer thread to turn off the output  
 threading.Thread(target=turn\_off\_output).start()  
 else:  
 if output\_active:  
 write\_coil(client, DATA\_COIL, False)  
 output\_active = False  
  
 cv2.imshow('Frame', frame)  
 if cv2.waitKey(1) & 0xFF == ord('q'): # Press 'q' to exit  
 break  
  
 finally:  
 # Close the webcam and Modbus connection  
 cap.release()  
 cv2.destroyAllWindows()  
 client.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 detect\_baby\_pink\_and\_trigger()

Count update in the excel file once the colour is encountered without the help of PLC,

import cv2

import numpy as np

import pandas as pd

from datetime import datetime

import time

# Define the HSV color range for dark blue

lower\_color = np.array([110, 50, 50])

upper\_color = np.array([130, 255, 255])

# Initialize the webcam

cap = cv2.VideoCapture(0)

# Initialize a DataFrame to store the counts and timestamps

log\_df = pd.DataFrame(columns=["Timestamp", "Count"])

# Initialize flag to track object detection

object\_detected = False

# Function to update the log DataFrame and save to Excel

def log\_count(count):

    timestamp = datetime.now()

    log\_df.loc[len(log\_df)] = [timestamp, count]

    log\_df.to\_excel("darkblue\_object\_counts.xlsx", index=False)

while True:

    # Capture frame-by-frame

    ret, frame = cap.read()

    if not ret:

        break

    # Convert the frame to HSV color space

    hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

    # Create a mask for the specified color range

    mask = cv2.inRange(hsv, lower\_color, upper\_color)

    # Find contours in the mask

    contours, \_ = cv2.findContours(mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    # Draw bounding box around the detected color and count the objects

    count = 0

    for contour in contours:

        if cv2.contourArea(contour) > 500:  # Adjust this threshold as needed

            x, y, w, h = cv2.boundingRect(contour)

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # Put the name of the color on top of the bounding box

            cv2.putText(frame, 'dark\_blue', (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

            count += 1

            # Log the count to the Excel sheet if object is newly detected

            if not object\_detected:

                log\_count(count)

                object\_detected = True

    # Reset object detected flag if no objects are currently detected

    if count == 0:

        object\_detected = False

    # Display the original frame with bounding boxes

    cv2.imshow('Frame', frame)

    # Wait for keypress

    key = cv2.waitKey(1)

    # Press 'q' to quit

    if key == ord('q'):

        break

    # Pause execution for 1 second

    time.sleep(1)

# Release the webcam and close all windows

cap.release()

cv2.destroyAllWindows()

**WHY CHOOSE PLC WHEN THERE IS AI?**

***Why Choose PLC Over AI for Industrial Automation?***

Programmable Logic Controllers (PLCs) are preferred over Artificial Intelligence (AI) in many industrial automation contexts due to their reliability, simplicity, and suitability for real-time control.

Reliability: PLCs are built to withstand harsh industrial environments, including extreme temperatures, dust, and electrical noise. Their robustness ensures consistent performance in critical applications, minimizing downtime and maintenance needs.

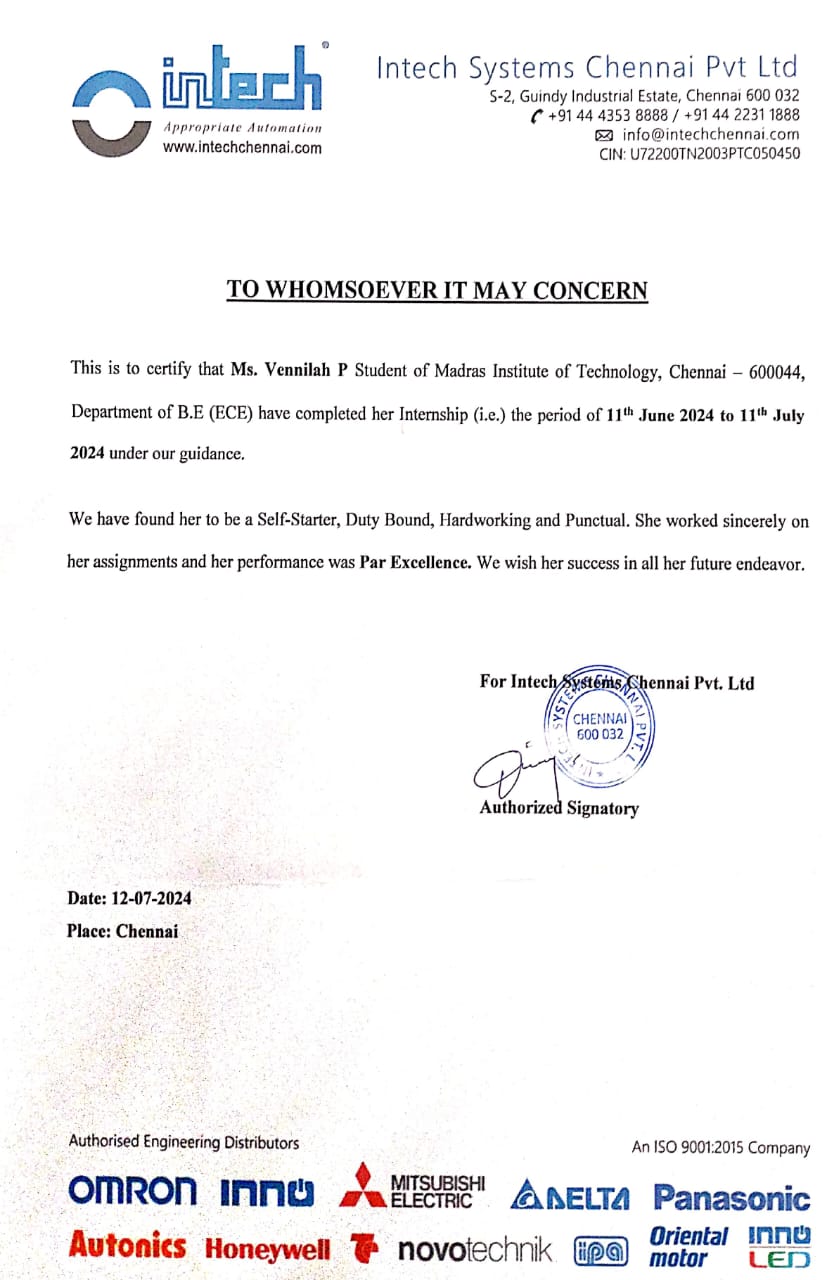
Deterministic Performance: PLCs provide precise and predictable control, executing logic in a deterministic manner. This is crucial for processes requiring exact timing and coordination, such as assembly lines and conveyor systems, where any delay or unpredictability could cause significant disruptions.

Ease of Use: PLCs are designed for engineers and technicians familiar with ladder logic, making them easier to program, troubleshoot, and maintain without needing extensive training in advanced computing or data science. This accessibility speeds up deployment and reduces operational costs.

Safety and Standards: PLCs comply with stringent industrial safety standards and certifications, ensuring they meet regulatory requirements. Their proven track record in managing safety-critical operations makes them a trusted choice in industries like manufacturing, automotive, and energy.

While AI offers advanced capabilities like predictive maintenance and adaptive control, PLCs remain the preferred choice for straightforward, reliable, and safe control tasks in industrial environments. They provide the dependability and ease of use that many industries require, making them indispensable in traditional automation.

**CERTIFICATION**

****