INTEGRATED ROBOT AUTOMATION SYSTEM WITH OPENCV AND PLC

BY

CHATCHAWAL KAMOLPORNPHAN

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN ROBOTICS AND AI

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Integrated Robot Automation System with OpenCV and PLC
Student Name	Mr. Chatchawal Kamolpornphan
Student ID.	62011107
Degree	Bachelor of Engineering in Robotics and AI

Project Advisor Signed: Dr. Poom K.

(Dr.Poom Konghuayrob)

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

Project Title	Integrated ABB Robot Automation System with OpenCV and PLC
Student Name	Mr. Chatchawal Kamolpornphan
Student ID.	62011107
Degree	Bachelor of Engineering in Robotics and AI
Project Advisor	Dr. Poom Konghuayrob
Academic Years	2022

ABSTRACT

This project aims to study the integration of different automation components, such as a PC, PLC, HMI, robot, and camera, to create a cohesive system for industrial applications. The primary focus is on the seamless interaction and communication between these components, while the secondary aspect involves using OpenCV for shape detection to control the robot arm. Additionally, this project serves an educational purpose, providing valuable insights into the design, implementation, and challenges of integrating various automation elements. The overall goal is to assess the feasibility, performance, and learning opportunities presented by this integrated system for real-world use and future development.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

ACKNOWLEDGEMENT

The completion of this cooperative study and internship could not have been possible without help and guidance from my intern-COOP supervisor Mr. Thana Cheewasupakorn (Robotic Technical support engineer) for providing me with all the support and troubleshooting help when the project is at a halt during development.

I would like to show my gratitude to my advisor for the capstone project, Dr. Poom Konghuayrob, who has been helping me with the project and providing me with ideas and motivation. He also helps me with the idea of working in the industry which will be a part of the future automating transformation. However, this project could not have been finished without Badr Azougagh, an exchange student from Belgium, who helped me during the development of the project with coding and implementing the camera. Not only that, but he also helps me as a companion while doing the project.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

TABLE OF CONTENT

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

LIST OF FIGURES

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations Terms	Terms
RWS	Robot Web Services
OpenCV	Open-Source Computer Vision Library
SDK	Software Development Kit
PLC	Programmable Logic Controller
HMI	Human-Machine Interface

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 1: INTRODUCTION

This chapter introduces the topic and ideas of the project into context. Mentioning the scope of study of the project, the objective of the whole study, and a summary of the method used to conduct the study

The Integrated Robot Automation System with OpenCV and PLC project is to conduct a study on how one implements an integrated automated robot system using the existing equipment easily available on campus and can be compared to the industry standard in manufacturing or else.

1.1 Background and significance

Industrial robots are automated machines that are used in various manufacturing processes to perform tasks such as welding, painting, assembly, and packaging (palletizing). They are used in a wide range of industries, including automotive, aerospace, and electronics, to increase productivity, reduce labor costs, and improve the quality and consistency of the final product.

The control of industrial robots is a complex task that involves several different technologies, including mechanics, electronics, and software. The robot control system is responsible for receiving commands from an operator or a computer program, interpreting those commands, and then using that information to control the robot's movements.

The significance of industrial robot control lies in its ability to automate repetitive and dangerous tasks, which can improve the efficiency and safety of manufacturing processes. The use of industrial robots can also result in improved quality control and increased flexibility in manufacturing, as robots can be programmed to handle a wide range of tasks, and can easily be retooled or reprogrammed to adapt to new products or production processes. The application and advancements in the field of industrial robot control also open new possibilities and areas of research such as Collaborative Robots

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

which operate in proximity to human workers, and further integration with Artificial Intelligence for decision-making, which might revolutionize the manufacturing industry.

The usage of the commercial industrial robot in education part is still limited due to the existing method of controlling the robot such as teaching pendant programming, and manufacturer's offline programming which are very limited when it comes to understanding the method of system integration. [1]

This study is focused on the independent control of the robot from an outer source, in this case, the Python application with OpenCV library for shape recognition. Along with using PLC to help control the other element of the automated system.

1.2 Scope of study

1. The robot's fundamental control options use its controller panel. utilizing RobotStudio and the program's simulation.
2. Method of controlling a robot using external software.
3. Integration of different Automation components into one cohesive system

1.3 The Objective of the Project

1. To Study the Basic operation of the robot.
2. To understand how to control the robot with the external program using Robot Web Service.
3. To understand the limitation of the EGM robot control.
4. Find a way to control the robot with external data parameters.

1.4 Method of Conducting the Study

1. Study the basic control of the robot
2. Research Robot Web Services in the ABB official manual website.
3. Develop, and select ways to control the parameters.
4. Develop a Python program for shape recognition and API to communicate with the robot controller.
5. Integrate the system with other automation components.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 2: CONCEPTS, THEORIES, AND RELATED TERMS

The chapter in this report titled "Concept Theories and Related Terms" discusses the fundamental principles and terminology underlying the integration of robotics, computer vision, and PLCs in an Integrated Robot Automation System. It provides a thorough understanding of key theoretical concepts such as object recognition, image processing, automation control, and sensor integration, laying the groundwork for practical implementation discussed in later chapters.

2.1 Industrial robot

2.1.1 Industrial Robotic arms, also referred to as articulated robotic arms are quick, accurate, and programmable to carry out a limitless number of tasks in a variety of settings. In factories, they are used to automate the execution of repetitive tasks. [2]

2.1.2 RobotStudio is a simulation and offline programming tool for robotic applications. RobotStudio suite, which uses virtual controller technology, gives you complete assurance that the movements you see on your screen correspond to how the robot will behave in real life. This innovative technology drastically reduces the time it takes to commission a robot installation and increases productivity by allowing you to build, test, and fine-tune it in a virtual environment.

2.1.3 RAPID programming language is a programming language developed by ABB (Asea Brown Boveri) for its family of industrial robots. It is a proprietary, high-level programming language that is used to write control programs for ABB robots. RAPID is designed to be easy to learn and use, and it includes a wide range of commands and functions that can be used to control the robot's movements, sensors, and other features. The language is based on a structured programming approach, which means that the code is organized into blocks, such as procedures, functions, and tasks, that can be reused and shared across different parts of the program.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
1  MODULE MainModule
2    PROC main()
3      MoveJ *,v1000,z50,MyTool;
4      MoveJ *,v1000,z50,MyTool;
5      MoveJ *,v1000,z50,MyTool;
6      MoveJ *,v1000,z50,MyTool;
7      MoveJ *,v1000,z50,MyTool;
8      MoveJ *,v1000,z50,MyTool;
9      MoveJ *,v1000,z50,MyTool;
10     MoveJ *,v1000,z50,MyTool;
11   ENDPROC
12 ENDMODULE
```

Figure 1: Example of RAPID code

2.2 ABB Robot Web Services

Robot Web Services is a platform that enables developers to create custom applications to interact with the robot controller. Robot Web Services exposes RESTful APIs that leverage the HTTPS protocol and the messages are composed of XHTML and JSON. Robot Web Services facilitates platform-independent and language-independent communication with the robot controller. In REST, an URL identifies a resource. The representation of the application data sent from the robot controller can be either in XHTML or JSON format. [3]

Figure 2: How RWS works

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.3 Python application

2.3.1 Request Python library

The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a short, simple API so that can focus on interacting with services and consuming data in the written application.[4]

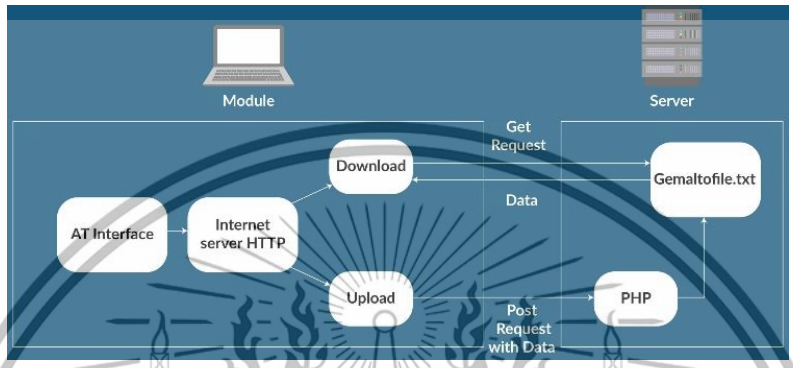

Figure 3: How the Request library works

2.3.2 OpenCV Python library

A Python library called OpenCV makes it possible to carry out image processing and computer vision tasks. It offers a variety of features, such as tracking, face recognition, and object detection. Which in this case we use for item shape detection.[5]

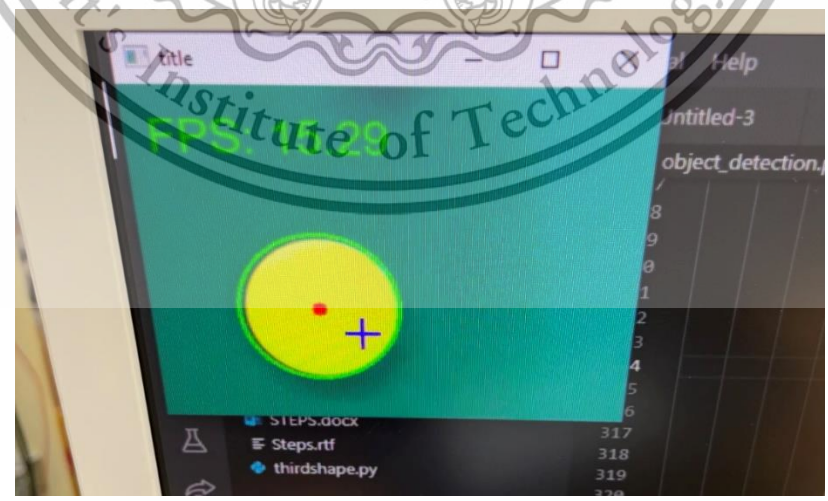

Figure 4: Shape Detection with OpenCV

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.3.3 Pypylon Python library

pypylon is Basler's pylon Camera Software Suite interface for interfacing Basler cameras with the programming language Python. This library creates flexibility for users to modify and build a camera system based on users' requirements. [6]

Figure 5: pypylon Python SDK

2.4 PLC (Programmable Logic Controller)

A PROGRAMMABLE LOGIC CONTROLLER (PLC) is an industrial computer control system that continuously monitors the state of input devices and makes decisions to control the state of output devices based on a custom program.

Using this type of control system, almost any production line, machine function, or process can be greatly improved. The ability to change and replicate the operation or process while collecting and communicating vital information is the most significant advantage of using a PLC. [7]

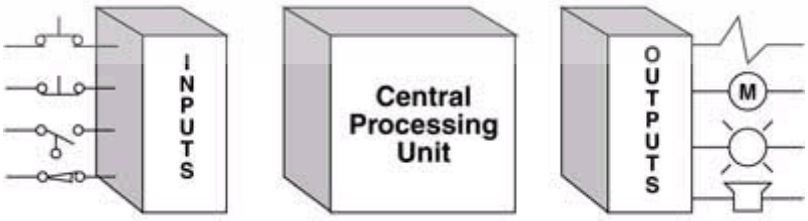

Figure 6: Diagram of PLC

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.5 Basler camera (acA2500-14gc)

Basler acA2500-14gc is a colored industrial camera connect using GigE ethernet. Can be controlled using the Pylon suite application or use pypylon Python SDK to use the stream of video into the image shape recognition. [8]

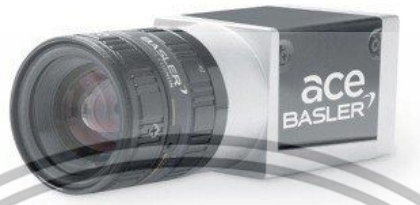

Figure 7: Basler acA2500-14gc Camera

King Mongkut's Institute of Technology Ladkrabang

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 3: METHODOLOGY

This chapter describes the design and development of the Integrated Robot Automation System with OpenCV and PLC. This chapter outlines the step-by-step process followed in assembling the various components and equipment, as well as the methodology employed in developing the application that controls the system. By providing a detailed account of the design methodology, including equipment integration, software development, and testing procedures. The methodology chapter sets the stage for the subsequent chapters, which present the results and conclusion of the implemented system.

3.1 Design Methodology

The design and development of the Integrated Robot Automation System with OpenCV and PLC began with the planning, researching, developing, and writing this paper.

3.2 Planning

In the first month of the project, I begin with the planning phase. I define the scope, objective, and the method itself. After that, I talked with my project advisor about the plan and the scope of the project. With his advice, I was able to organize the next step of the project.

3.3 Research

The research phase begins around the same time as the planning one because the knowledge needed to do the project is clear on which documentation, I need to focus on to be able to start the development phase. In this case, I need to research on these topics.

1. OpenCV Python library is used for writing code that recognizes shapes, detects, and finds the coordinates of the object present from the video feed.
2. Pypylon Python library and Pylon application. I need to do research on this based on the camera equipment chosen for the project, the Basler GigE camera. With the pypylon SDK, the Python code can control all the camera parameters and get the video feed to process.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3. ABB Robot Web Services is a service that allow external output to be input into the robot, execute tasks, or modified value in the robot controller using it.

3.4 Development phase

During this development phase, I will explain and dive deeply into all information and the way of conducting project research. Starting with the chart diagram that explains the flow of communication with each other.
(Shown in Figure 8.)

Figure 8: Flow of the communication between components.

After designing the flow of the communication, now I started to develop code in each part.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

A. ABB RAPID programming part

To create a RAPID program that has variables ready for the new data to come in and modify it. But first, we need to set up the robot to be ready for the program.

The robot will be using the “tVac” tool (Vacumn head) to use the suction to pick up the object on the conveyer belt. We need to define the tool with the teach pendant using the defining tool in the jogging panel (shown in Figure 9.)

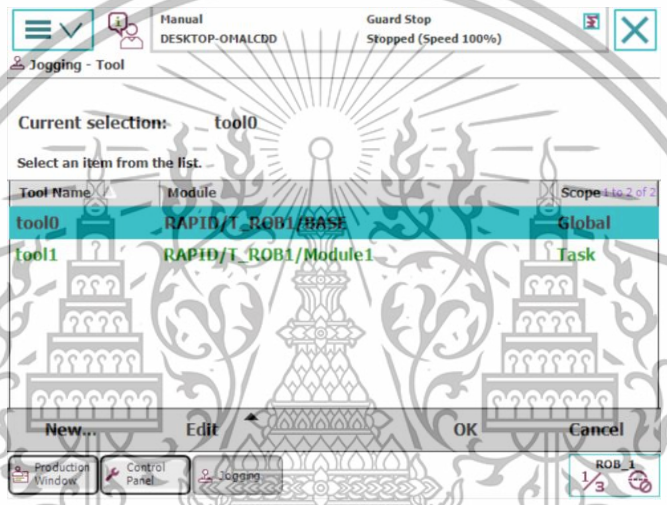

Figure 9: Tooldata page

After finishing the tool-defining step, now we create variables for the robot’s item coordinates. In this case, we define circle and square variables.

```
MODULE MainModule
    PERS num xCircle:=579.682;
    PERS num yCircle:=-317.074;
    PERS num zCircle:=0;
    PERS num xSquare:=556.214;
    PERS num ySquare:=-300.521;
    PERS num zSquare:=0;
    PERS num xTri:=551.425;
    PERS num yTri:=-259.139;
    PERS num zTri:=0;
    PERS num Shape:=1;
    !shape 1 is circle. Shape 2 is square.


    CONST jointtarget homePose:=[[0,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    TASK PERS tooldata tVac:=[TRUE,[[2.1198,-1.83154,115.969],[0.85158,0.271089,-0.427549,0.136104]],[0.3,[9.6,0,107.7],[1,0,0,0],0,0,0.001]];
    TASK PERS loaddata load1:=[0,[0,0,0],[1,0,0,0],0,0,0];
    CONST jointtarget jPosReady:=[[0,0,20,0,-110,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget PlaceCircle:=[[533.64,404.44,1043.93],[0.851247,0.271359,-0.427732,0.137067],[0,-1,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget PlaceSquare:=[[342.40,404.44,1043.94],[0.851251,0.271354,-0.427724,0.13708],[0,-1,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget Conv1:=[[558.13,-299.547,992],[0.8498,0.270542,-0.431076,0.137189],[-2,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figure 10: Variable in RAPID code

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

After defining all the variables, we now focus on the instructions which will dictate the movement of the robot that is fixed. For example, the path that the robot will move to be in the ready position, above the conveyer, and the placing spot of each object.

```
PROC pCircle()

    Conv1.trans.x:=xCircle;
    Conv1.trans.y:=yCircle;

    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    waitTime 0.5;
    MoveL Offs(Conv1, 0, 0, -80), v100, fine, tVac;
    waitTime 1;
    MoveL Conv1,v100,fine,tVac;
    waitTime 0.5;
    Set doVac;
    WaitTime 0.5;
    MoveL Offs(Conv1, 0, 0, -80), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveL PlaceCircle, v100, fine, tVac;
    waitTime 0.5;
    Reset doVac;
    WaitTime 0.5;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveL Offs(PlaceCircle, 0, 0, 20), v100, fine, tVac;!!!
    Set doVac;
    WaitTime 0.5;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(Conv1, -550, 0, -80), v100, fine, tVac;
    MoveL Offs(Conv1, -550, 0, -20), v100, fine, tVac;
    Reset doVac;
    waitTime 0.5;
    MoveL Offs(Conv1, -550, 0, -80), v100, fine, tVac;
ENDPROC
```

Figure 11: RAPID function for the pick and place path

Lastly, for the robot controller, we need to set the mode of the robot to run in continuous mode for the program to run without stopping.

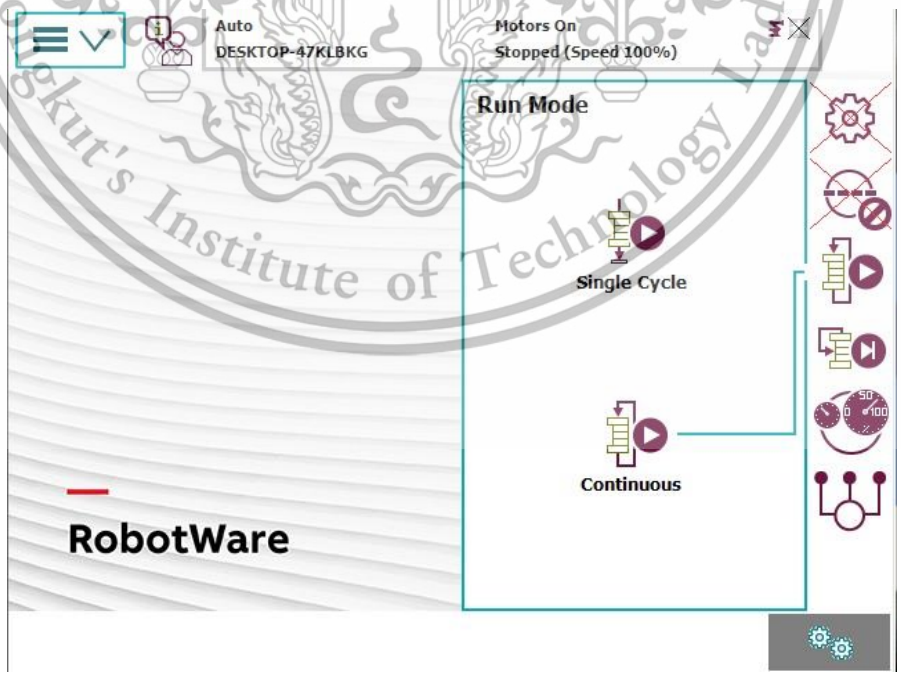

Figure 12: Continuous mode setting

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

B. Python programming part

The code we develop in Python performs item shape detection, converts the pixel coordinates, and sends it to the ABB robot controller. We will explain in detail. Starting with all the library needed to perform its tasks.

The necessary imports are included, such as concurrent.futures for parallel processing, datetime for time-related operations, sleep for delaying execution, uuid for generating unique identifiers, os for operating system-related functionalities, subprocess for running shell commands, cv2 for computer vision operations, numpy for numerical computations, requests for making HTTP requests, PIL for image manipulation, and pylon for Basler GigE camera adjustment.

```
1  from concurrent.futures import ThreadPoolExecutor
2  from datetime import datetime
3  from time import sleep
4  import uuid
5  import os
6  import subprocess
7  import cv2
8  import numpy as np
9  import requests
10 from PIL import Image, ImageDraw, ImageFont
11 from pypylon import pylon
12 from requests.auth import HTTPDigestAuth
13 import time
```

Figure 13: Imports in Python code

The display_fps function takes an image, frame count, and start time as input. It calculates the frames per second (FPS) based on the elapsed time and frame count, adds the FPS text to the image using the PIL library, and returns the modified image as a NumPy array. The reason we need the FPS of the camera is that we want to know how well the camera is performing with the current adjustment.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```python
def display_fps(image, frame_count, start_time):
    elapsed_time = (datetime.now() - start_time).total_seconds()
    fps = frame_count / elapsed_time
    fps_text = f"FPS: {fps:.2f}"
    font = ImageFont.truetype("arial.ttf", 60)
    img_pil = Image.fromarray(image)
    draw = ImageDraw.Draw(img_pil)
    draw.text((30, 30), fps_text, font=font, fill=(0, 255, 0))
    return np.array(img_pil)
```

Figure 14: Display FPS Function

After that, we create a class for the object detection part to be able to set all the variables needed to use and organize it. In this class, there are functions that are used for detecting shapes, yellow circles, and white squares, which use OpenCV to detect the contour of the object based on the color range we defined and applied OpenCV functions such as cv2.minEnclosingCircle and cv2.boundingRect for each shape and get the object's center coordinates in pixel.

After we get the pixel coordinates and the shape of the object, now we create a 'send_center_to_robotarm' function. Starting with defining a 'shape_map' dictionary. This dictionary maps the shape names ('circle', 'square', 'triangle') to their corresponding variables in the robot arm controller. This mapping allows for easy reference when constructing the URLs to update the variables.

```python
def send_center_to_robotarm(x_center, y_center, shape):
    shape_map = {
        'circle': 'xCircle',
        'square': 'xSquare',
        'triangle':'xTri'
    }
```

Figure 15: Shape name and variable that will be sent to the controller

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

After checking if the shape is a valid shape based on the shape_map, the function proceeds to calculate the adjusted x and y coordinates. It seems that there is a scaling and translation operation applied to the original x_center and y_center values. The specific formulas used for the adjustment are:

- **x = ((x_center - 52) * 0.47892377) + 527**
- **y = ((y_center - 52) * 0.48684211) - 334.6**

This formula is based on the fixed location of the camera. To convert pixel coordinates to robot coordinates, we made an equation to convert it and use it in this code.

```
x=((x_center-52)*0.47892377)+527
y=((y_center-52)*0.48684211)-334.6
```

Figure 16: Pixel to robot target equation

The next function is creating the URL for sending the 'x' and 'y' coordinated that already have been converted to the robot controller. The URL is created using string interpolation which uses the 'shape_map' to determine the variable name based on the 'shape' parameters.

After creating the URLs for the coordinates, we also need to send the information about the shape type of the object which the robot needs to know to place the object in the correct location.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
    x_url = f"http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
    MainModule/{shape_map[shape]}?action=set"
    y_url = f"http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
    MainModule/y{shape_map[shape][1:]}?action=set"

    if shape=='circle':
        shape_url = "http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
        MainModule/Shape?action=set"
        shape_value = 1
    else:
        shape_url = "http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
        MainModule/Shape?action=set"
        shape_value = 2
```

Figure 17: Defined URLs

With all the URLs defined, now the code will send the information using the 'session' Python library that we imported earlier.

```
    x_response = session.post(x_url, data={"value": x})
    y_response = session.post(y_url, data={"value": y})
    shape_response = session.post(shape_url, data={"value": shape_value})
```

Figure 18: session execution

C. Ladder logic programming and Human-Machine Interface

In this automated system, we also use PLC to control the conveyer to move the object into the area of interest of the camera and notify the robot controller when the object is there. The equipment connected with the PLC contains the digital infrared sensor attached to the conveyer and the motor that drive the conveyer.

In this case, we also incorporate the HMI (Human-Machine Interface) to create buttons to start and stop the PLC from running and stopping instead of using physical buttons.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

To start, we need to assign each element in the PLC system to be able to create a program. We assign each element as.

- M0 Start button
- M1 Stop button
- X1 Digital Infrared Sensor
- Y1 Conveyer Motor
- Y2 Digital Output to ABB Robot Controller
- T1 2.5-second timer

Figure 19: Ladder Program used in the system

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

For the HMI, we only need to design the page to have 2 buttons, start and stop. We must assign using the AKtool application use for edit Samkoon branded HMI. And the screen can be seen shown below with the configuration window used for the setting for one of the buttons.

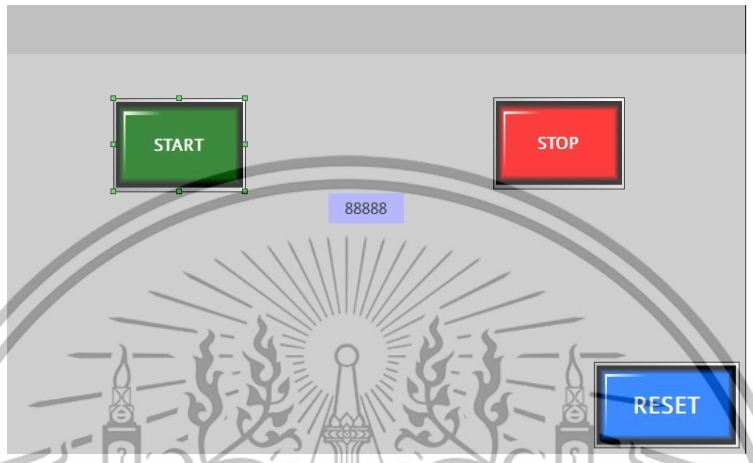

Figure 20: HMI Screen

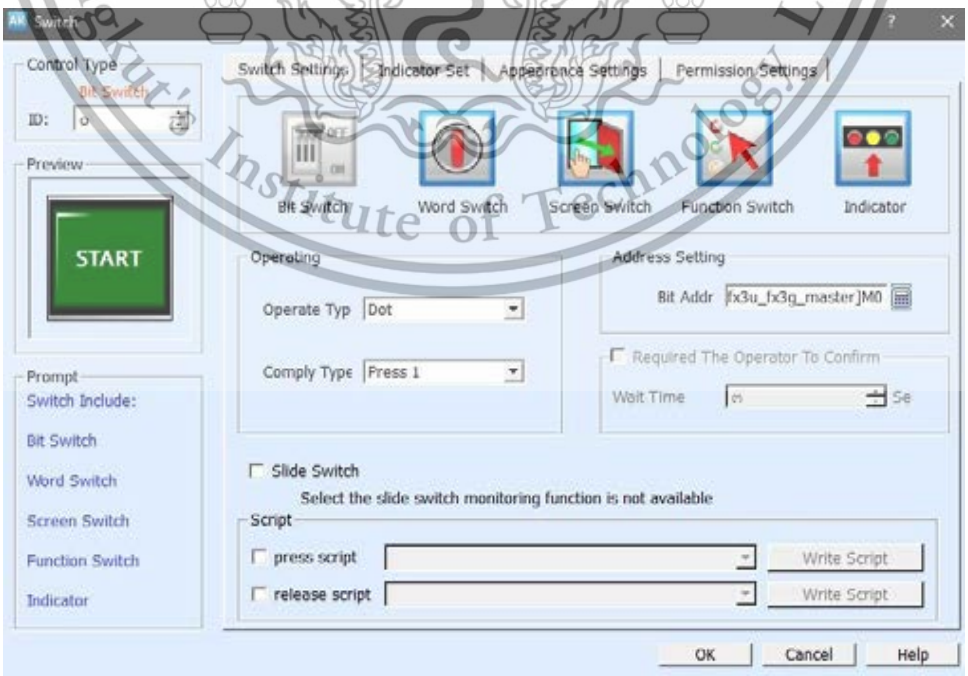

Figure 21: Switch setting

This material is reserved for educational purposes only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

For each switch, the need to have an address for each key to be able to communicate with the PLC through the COM port.

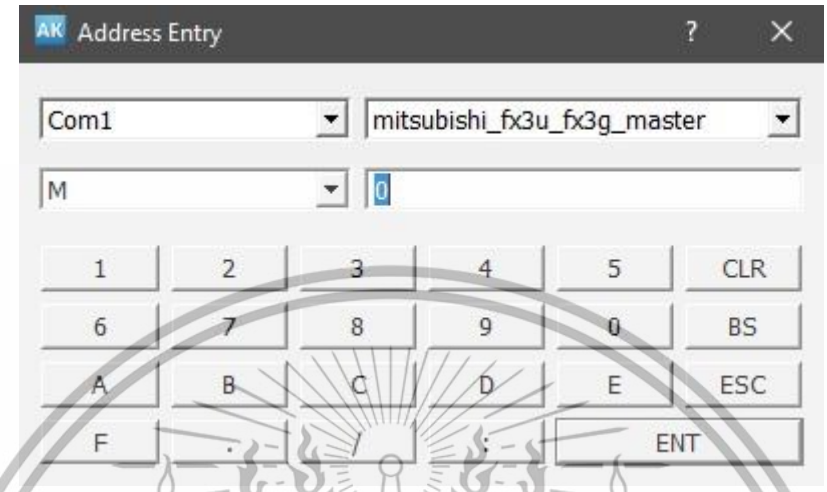

Figure 22: Assign each key to the address

D. Physical components

For this part, we need to assemble everything. Which include all element such as the Robot arm, robot controller, PLC, HMI, and PC.

Figure 23: Basler camera mounting hole

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

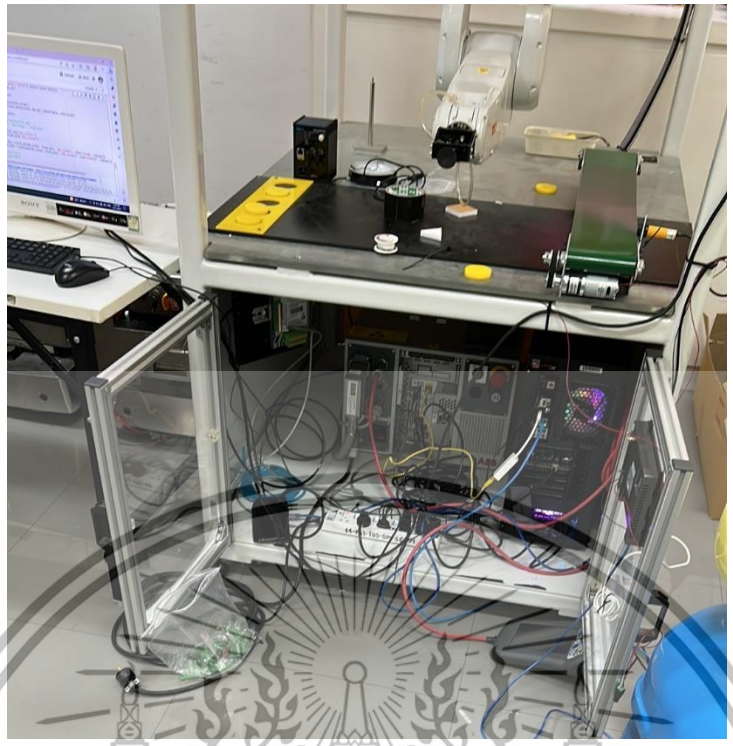

Figure 24: Robot enclosure with all Equipment

E. Complete Flowchart of the system operation

With all the component connected as a system. A flow of execution is formed as a cohesive single flow of operation. Using the ABB Robot Controller as a core for all the process and show the flowchart as the controller's perspective. As shown in Figure 25.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

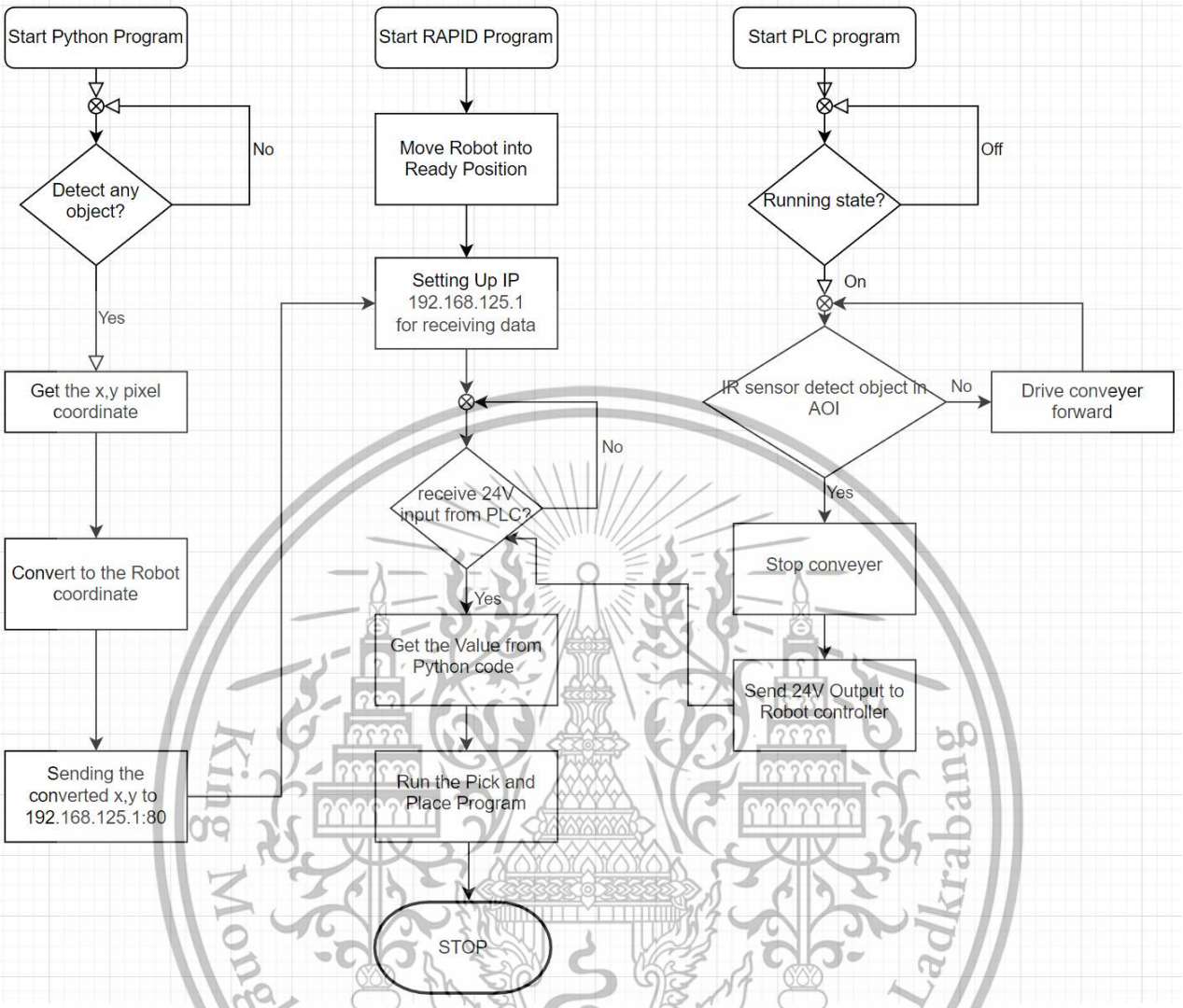

Figure 25: Flowchart of the operation

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 4: PROJECT RESULT

In this chapter, we will be looking at the result of the entire project after the development that took place before.

The system successfully detected shapes, send their location to the robot arm, and picked objects accurately. The communication between the Robot controller, PC, PLC, and HMI allows the system to operate within the program instructed. The Robot moves according to the instruction in the program without any collision, singularity problem, or failed execution.

Figure 26: The screen is shown the PC running the code

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

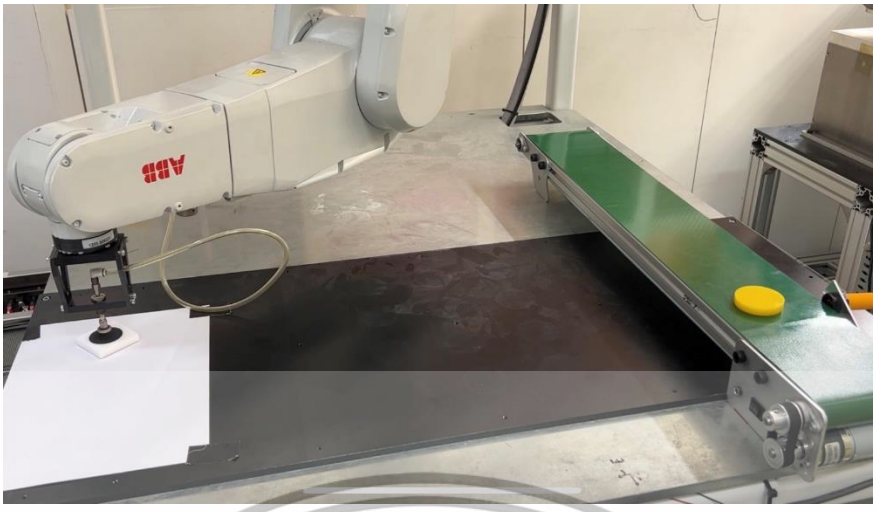

Figure 27: Robot placing the square object

The accuracy of the camera is very good, resulting in the stable coordinates number shown in Figure 27.

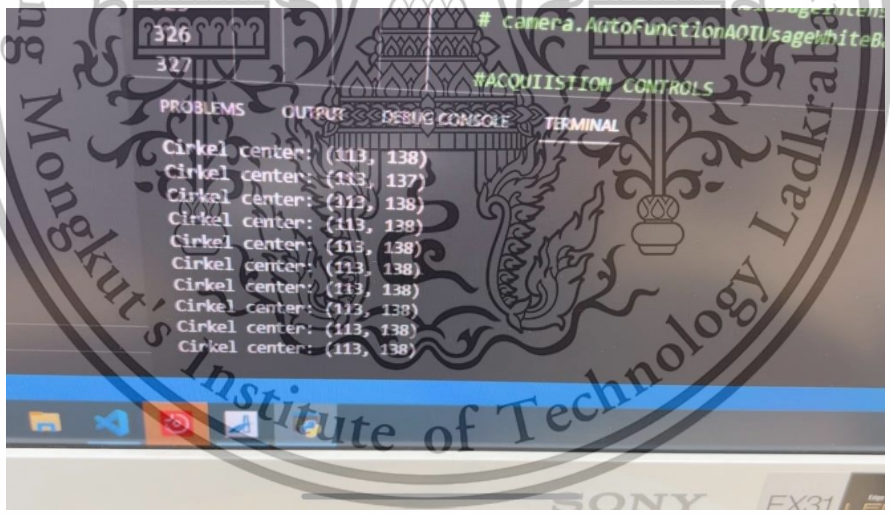

Figure 28: Terminal returns the pixel coordinate of the circle

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

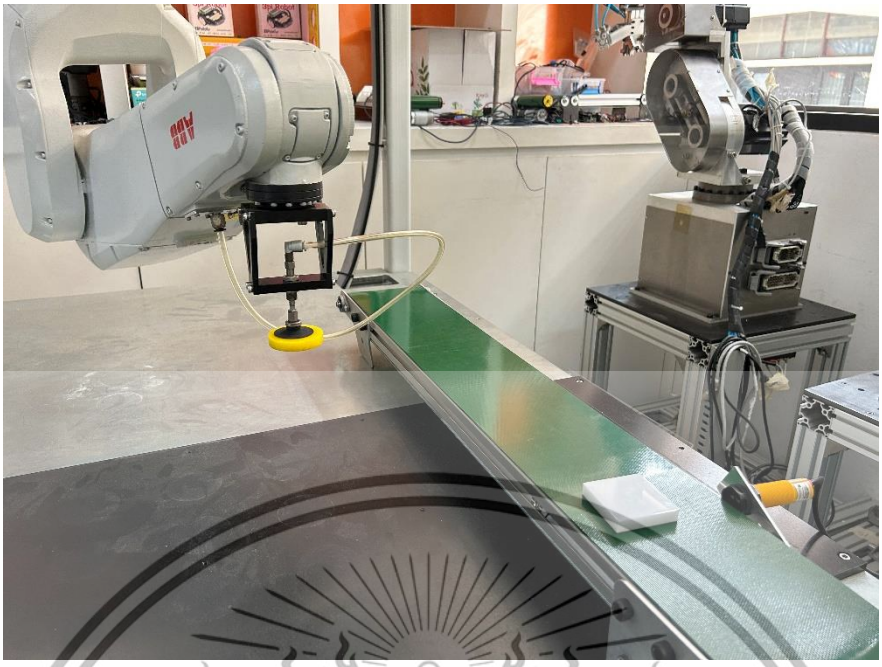

Figure 29: Robot Picking Circle Object

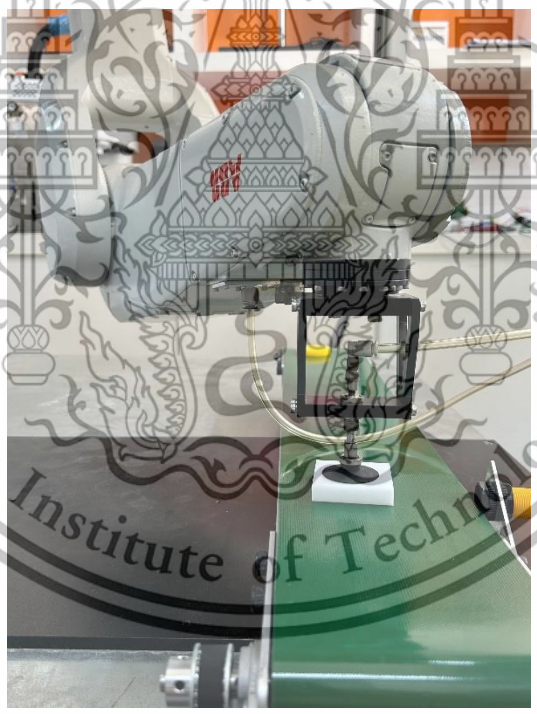

Figure 30: Robot picking object accurately

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

CHAPTER 5: CONCLUSION

The integration of a PC, PLC, HMI, robot, and camera in a unified system presents several challenges but also offers potential benefits for industrial applications. But not only the industrial application, the system also benefits the educational purpose that helps the student understand another way to control a branded industrial robot. But there is some observation that has been made.

2 observation point has been made as

1. The Robot Web Service still has many capabilities that can be studied and used in the future. Some of the execution such as Externally Guided Motion can be configured to move the robot in real time.
2. Controlling the robot using this method has many capabilities. But when the value sent to the robot is always changing. The robot will take the value only just before the time the line of the program run. So, in the application that the object always moving (non-stop conveyer, conveyer tracking), we need to find another method of controlling the robot.

To conclude with those observations, this method of controlling the robot shows many more capabilities to integrate with another system with more complex equipment. Moreover, the documentation of Robot Web Service can be studied more to be able to create a more complex system and make a library out of it to use in the future. However, Controlling the industrial robot this way can only be used for certain use cases in that real-time control is not involved.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

REFERENCE

[1] Owen-Hill, A. (2021) The Pros and cons of 5 robot programming methods, RoboDK blog. Available at: https://robodk.com/blog/robot-programming-methods-pros-and-cons/ (Accessed: 10 May 2023).

[2] Industrial Robotic Arm (no date) Intel. Available at: https://www.intel.com/content/www/us/en/robotics/robotic-arm.html.

[3] (No date) ABB-Robot Web Services. ABB. Available at: https://developercenter.robotstudio.com/api/RWS (Accessed: January 2022).

[4] Real Python (2022) Python's Requests Library (guide), Real Python. Real Python. Available at: https://realpython.com/python-requests/ (Accessed: January 2023).

[5] Team, G.L. (2022) OpenCV tutorial: A guide to learn OpenCV in Python, Great Learning Blog: Free Resources what Matters to Shape your Career! Available at: https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/#:~:text=OpenCV%20is%20a%20Python%20library,%2C%20face%20recognition%2C%20and%20tracking (Accessed: 10 May 2023).

[6] Basler AG (no date) Pylon open source, Basler AG. Available at: https://www.baslerweb.com/en/products/basler-pylon-camera-software-suite/pylon-open-source-projects (Accessed: 10 May 2023).

[7] What is a plc? (No date) AMCI. Available at: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc (Accessed: 10 May 2023).

[8] AG, B. (no date) ACA2500-14GC#, acA2500-14gc | Basler Product Documentation. Available at: https://docs.baslerweb.com/aca2500-14gc (Accessed: 25 April 2023).

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Integrated Robot Automation System with OpenCV and PLC

Chatchawal Kamolpornphan
Robotics and AI Department of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
chatchawal.kam@gmail.com

***Abstract*— This project aims to study the integration of different automation components, such as a PC, PLC, HMI, robot, and camera, to create a cohesive system for industrial applications. The primary focus is on the seamless interaction and communication between these components, while the secondary aspect involves using OpenCV for shape detection to control the robot arm. Additionally, this project serves an educational purpose, providing valuable insights into the design, implementation, and challenges of integrating various automation elements. The overall goal is to assess the feasibility, performance, and learning opportunities presented by this integrated system for real-world use and future development.**

Keywords—Robot Web Services, OpenCV, RobotStudio, RAPID, pypylon, PLC

I. INTRODUCTION

This chapter introduces the topic and ideas of the project into context. Mentioning the scope of study of the project, the objective of the whole study, and a summary of the method used to conduct the study

The Integrated Robot Automation System with OpenCV and PLC project is to conduct a study on how one implements an integrated automated robot system using the existing equipment easily available on campus and can be compared to the industry standard in manufacturing or else.

Use the enter key to start a new paragraph. The appropriate spacing and indent are automatically applied.

A. Background and significance

Industrial robots are automated machines that are used in various manufacturing processes to perform tasks such as welding, painting, assembly, and packaging (palletizing). They are used in a wide range of industries, including automotive, aerospace, and electronics, to increase productivity, reduce labor costs, and improve the quality and consistency of the final product.

The control of industrial robots is a complex task that involves several different technologies, including mechanics, electronics, and software. The robot control system is responsible for receiving commands from an operator or a computer program, interpreting those commands, and then using that information to control the robot's movements.

The significance of industrial robot control lies in its ability to automate repetitive and dangerous tasks, which can improve the efficiency and safety of manufacturing processes. The use of industrial robots can also result in improved quality control and increased flexibility in manufacturing, as robots can be programmed to handle a wide range of tasks, and can easily be retooled or reprogrammed to adapt to new products or production processes. The application and advancements in the field of industrial robot control also open new possibilities and areas of research such as Collaborative Robots which operate in proximity to human workers, and further integration with Artificial Intelligence for decision-making, which might revolutionize the manufacturing industry.

The usage of the commercial industrial robot in education part is still limited due to the existing method of controlling the robot such as teaching pendant programming, and manufacturer's offline programming which are very limited when it comes to understanding the method of system integration. [1]

This study is focused on the independent control of the robot from an outer source, in this case, the Python application with OpenCV library for shape recognition. Along with using PLC to help control the other element of the automated system.

B. Scope of Study

1) The robot's fundamental control options use its controller panel. utilizing RobotStudio and the program's simulation.
2) Method of controlling a robot using external software.
3) Integration of different Automation components into one cohesive system

C. The Objective of the Project

1) To Study the Basic operation of the robot.
2) To understand how to control the robot with the external program using Robot Web Service.
3) To understand the limitation of the EGM robot control.
4) Find a way to control the robot with external data parameters.

D. Method of Conducting the Study

1) Study the basic control of the robot
2) Research Robot Web Services in the ABB official manual website.
3) Develop, and select ways to control the parameters.
4) Develop a Python program for shape recognition and API to communicate with the robot controller.
5) Integrate the system with other automation components.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

II. CONCEPT, THEORIES, AND RELATED TERMS

The chapter in this report titled "Concept Theories and Related Terms" discusses the fundamental principles and terminology underlying the integration of robotics, computer vision, and PLCs in an Integrated Robot Automation System. It provides a thorough understanding of key theoretical concepts such as object recognition, image processing, automation control, and sensor integration, laying the groundwork for practical implementation discussed in later chapters.

A. Industrial Robot

Industrial Robotic arms, also referred to as articulated robotic arms are quick, accurate, and programmable to carry out a limitless number of tasks in a variety of settings. In factories, they are used to automate the execution of repetitive tasks. [2]

RobotStudio is a simulation and offline programming tool for robotic applications. RobotStudio suite, which uses virtual controller technology, gives you complete assurance that the movements you see on your screen correspond to how the robot will behave in real life. This innovative technology drastically reduces the time it takes to commission a robot installation and increases productivity by allowing you to build, test, and fine-tune it in a virtual environment

RAPID programming language is a programming language developed by ABB (Asea Brown Boveri) for its family of industrial robots. It is a proprietary, high-level programming language that is used to write control programs for ABB robots. RAPID is designed to be easy to learn and use, and it includes a wide range of commands and functions that can be used to control the robot's movements, sensors, and other features. The language is based on a structured programming approach, which means that the code is organized into blocks, such as procedures, functions, and tasks, that can be reused and shared across different parts of the program

B. ABB Robot Web Services

Robot Web Services is a platform that enables developers to create custom applications to interact with the robot controller. Robot Web Services exposes RESTful APIs that leverage the HTTPS protocol and the messages are composed of XHTML and JSON. Robot Web Services facilitates platform-independent and language-independent communication with the robot controller. In REST, an URL identifies a resource. The representation of the application data sent from the robot controller can be either in XHTML or JSON format. [3]

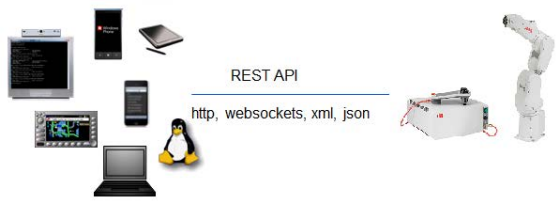

Figure 1: How Robot Web Service works

C. Python Application

Python application A Python application is a software program developed using the Python programming language that performs specific tasks or provides desired functionality. It utilizes Python's syntax, libraries, and tools to create executable code for various purposes. In this case for the purpose of automation system which require these libraries. .

The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a short, simple API so that can focus on interacting with services and consuming data in the written application.[4]

A Python library called OpenCV makes it possible to carry out image processing and computer vision tasks. It offers a variety of features, such as tracking, face recognition, and object detection. Which in this case we use for item shape detection.[5]

Pypylon library is Basler's pylon Camera Software Suite interface for interfacing Basler cameras with the programming language Python. This library creates flexibility for users to modify and build a camera system based on users' requirements. [6]

D. Programmable Logic Controller

A Programmable Logic Controller (PLC) is an industrial computer control system that continuously monitors the state of input devices and makes decisions to control the state of output devices based on a custom program.

Using this type of control system, almost any production line, machine function, or process can be greatly improved. The ability to change and replicate the operation or process while collecting and communicating vital information is the most significant advantage of using a PLC. [7]

E. Basler Camera

Basler acA2500-14gc is a colored industrial camera connect using GigE ethernet. Can be controlled using the Pylon suite application or use pypylon Python SDK to use the stream of video into the image shape recognition. [8]

III. METHODOLOGY

This chapter describes the design and development of the Integrated Robot Automation System with OpenCV and PLC. This chapter outlines the step-by-step process followed in assembling the various components and equipment, as well as the methodology employed in developing the application that controls the system. By providing a detailed account of the design methodology, including equipment integration, software development, and testing procedures. The methodology chapter sets the stage for the subsequent chapters, which present the results and conclusion of the implemented system.

A. Design Methodology

The design and development of the Integrated Robot Automation System with OpenCV and PLC began with the planning, researching, developing, and writing this paper.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

B. Planning

In the first month of the project, I begin with the planning phase. I define the scope, objective, and the method itself. After that, I talked with my project advisor about the plan and the scope of the project. With his advice, I was able to organize the next step of the project.

C. Research

The research phase begins around the same time as the planning one because the knowledge needed to do the project is clear on which documentation, I need to focus on to be able to start the development phase. In this case, I need to research on these topics.

1) OpenCV Python library is used for writing code that recognizes shapes, detects, and finds the coordinates of the object present from the video feed.

2) Pypylon Python library and Pylon application. I need to do research on this based on the camera equipment chosen for the project, the Basler GigE camera. With the pypylon SDK, the Python code can control all the camera parameters and get the video feed to process.

3) ABB Robot Web Services is a service that allow external output to be input into the robot, execute tasks, or modified value in the robot controller using it.

D. Development Phase

During this development phase, I will explain and dive deeply into all information and the way of conducting project research. Starting with the chart diagram that explains the flow of communication with each other.

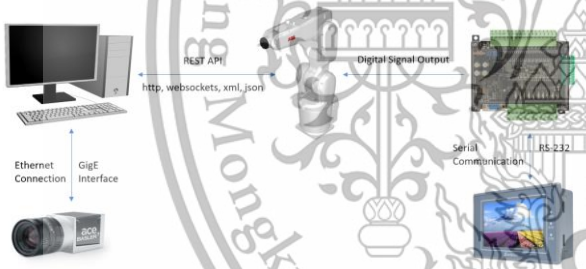

Figure 2: Flow of the communication between components.

After designing the flow of the communication, now I started to develop code in each part.

Starting with the RAPID programming with the robot controller. To create a RAPID program that has variables ready for the new data to come in and modify it. But first, we need to set up the robot to be ready for the program. Set up the tool by defining it in the teaching pendant. Create instruction to receive the data from the Robot Web Services and applied to the variable used in the picking instruction. After that, we now focus on the instructions which will dictate the movement of the robot that is fixed. For example, the path that the robot will move to be in the ready position, above the conveyer, and the placing spot of each object. (Figure 3.)

And last is to set the robot to run in continuous mode for the program to run without stopping.

```
PROC pCircle()

    Conv1.trans.x:=xCircle;
    Conv1.trans.y:=yCircle;

    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    waitTime 0.5;
    MoveL Offs(Conv1, 0, 0, -80), v100, fine, tVac;
    waitTime 1;
    MoveL Conv1,v100,fine,tVac;
    waitTime 0.5;
    Set doVac;
    WaitTime 0.5;
    MoveL Offs(Conv1, 0, 0, -80), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveL PlaceCircle, v100, fine, tVac;
    waittime 0.5;
    Reset doVac;
    WaitTime 0.5;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveL Offs(PlaceCircle, 0, 0, 20), v100, fine, tVac;!!!
    Set doVac;
    waitTime 0.5;
    MoveL Offs(PlaceCircle, 0, 0, -50), v100, fine, tVac;
    MoveAbsJ jPosReady\NoEOffs,v200,fine,tool0;
    MoveL Offs(Conv1, -550, 0, -80), v100, fine, tVac;
    MoveL Offs(Conv1, -550, 0, -20), v100, fine, tVac;
    Reset doVac;
    waitTime 0.5;
    MoveL Offs(Conv1, -550, 0, -80), v100, fine, tVac;
ENDPROC
```

Figure 3: RAPID Function for the picking and placing circle

Python code that we develop performs item shape detection, converts the pixel coordinates, and sends it to the ABB robot controller. We will explain in detail. Starting with all the library needed to perform its tasks

The necessary imports are included, such as 'concurrent.futures' for parallel processing, 'datetime' for time-related operations, sleep for delaying execution, 'uuid' for generating unique identifiers, 'os' for operating system-related functionalities, subprocess for running shell commands, 'cv2' for computer vision operations, 'numpy' for numerical computations, requests for making HTTP requests, 'PIL' for image manipulation, and 'pylon' for Basler GigE camera adjustment

The next is to create instructions for the PC to detect and send the converted coordinates to the robot controller. We create a class for the object detection part to be able to set all the variables needed to use and organize it. In this class, there are functions that are used for detecting shapes, yellow circles, and white squares, which use OpenCV to detect the contour of the object based on the color range we defined and applied OpenCV functions such as cv2.minEnclosingCircle and cv2.boundingRect for each shape and get the object's center coordinates in pixel.

After we get the pixel coordinates and the shape of the object, now we create a 'send_center_to_robotarm' function. Starting with defining a 'shape_map' dictionary. This dictionary maps the shape names ('circle', 'square', 'triangle') to their corresponding variables in the robot arm controller. This mapping allows for easy reference when constructing the URLs to update the variables.

After checking if the shape is a valid shape based on the shape_map, the function proceeds to calculate the adjusted x and y coordinates. It seems that there is a scaling and translation operation applied to the original x_center and y_center values. The specific formulas used for the adjustment are shown in Figure 4. This formula is based on the fixed location of the camera.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

- **x = ((x_center - 52) * 0.47892377) + 527**
- **y = ((y_center - 52) * 0.48684211) - 334.6**

Figure 4: The Pixel to Robot target Equation.

The next function is creating the URL for sending the 'x' and 'y' coordinated that already have been converted to the robot controller. The URL is created using string interpolation which uses the 'shape_map' to determine the variable name based on the 'shape' parameters.

After creating the URLs for the coordinates, we also need to send the information about the shape type of the object which the robot needs to know to place the object in the correct location. With all the URLs defined, now the code will send the information using the 'session' Python library that we imported earlier.

```
x_url = f"http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
MainModule/{shape_map[shape]}?action=set"
y_url = f"http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
MainModule/y{shape_map[shape][1:]}?action=set"

if shape=='circle':
    shape_url = "http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
    MainModule/Shape?action=set"
    shape_value = 1
else:
    shape_url = "http://192.168.125.1:80/rw/rapid/symbol/data/RAPID/T_ROB1/
    MainModule/Shape?action=set"
    shape_value = 2
```

Figure 5: Defined URLs

we also use PLC to control the conveyer to move the object into the area of interest of the camera and notify the robot controller when the object is there. The equipment connected with the PLC contains the digital infrared sensor attached to the conveyer and the motor that drive the conveyer.

In this case, we also incorporate the HMI (Human-Machine Interface) to create buttons to start and stop the PLC from running and stopping instead of using physical buttons.

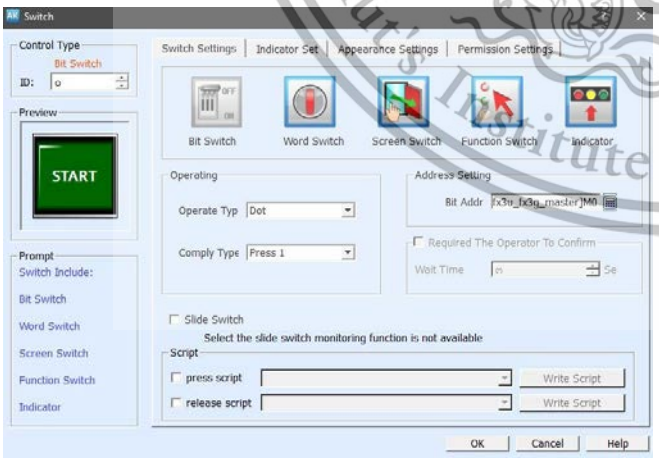

Figure 6: Button setting for the Human-Machine Interface

With all the component connected as a system. A flow of execution is formed as a cohesive single flow of operation. Using the ABB Robot Controller as a core for all the process and show the flowchart as the controller's perspective. As shown in Figure 7.

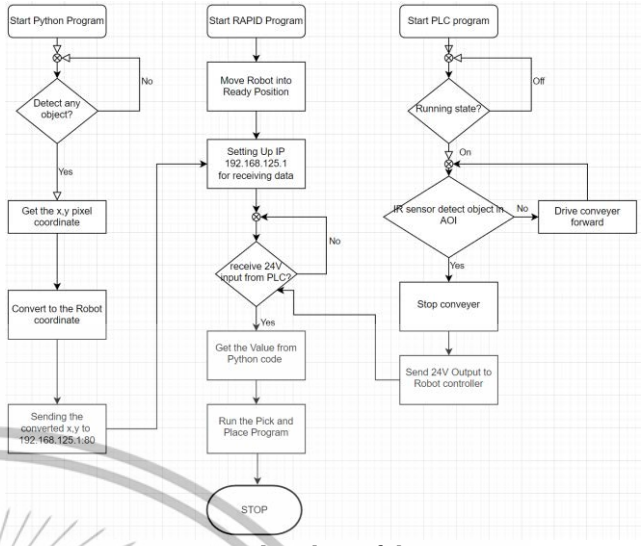

Figure 7: Flowchart of the operation

IV. PROJECT RESULT

In this chapter, we will be looking at the result of the entire project after the development that took place before.

The system successfully detected shapes, send their location to the robot arm, and picked objects accurately. The communication between the Robot controller, PC, PLC, and HMI allows the system to operate within the program instructed. The Robot moves according to the instruction in the program without any collision, singularity problem, or failed execution.

(a) (b) (c)

Figure 8: (a) Robot placing the object back to the conveyer. (b) Picking up object based on the coordinate send by the PC. (c) The object shown in the PC via Camera video feed.

V. CONCLUSION

The integration of a PC, PLC, HMI, robot, and camera in a unified system presents several challenges but also offers potential benefits for industrial applications. But not only the industrial application, the system also benefits the educational purpose that helps the student understand another way to control a branded industrial robot. But there is some observation that has been made.

The Robot Web Service still has many capabilities that can be studied and used in the future. Some of the execution such

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

as Externally Guided Motion can be configured to move the robot in real time.

Controlling the robot using this method has many capabilities. But when the value sent to the robot is always changing. The robot will take the value only just before the time the line of the program run. So, in the application that the object always moving (non-stop conveyer, conveyer tracking), we need to find another method of controlling the robot.

To conclude with those observations, this method of controlling the robot shows many more capabilities to integrate with another system with more complex equipment. Moreover, the documentation of Robot Web Service can be studied more to be able to create a more complex system and make a library out of it to use in the future. However, Controlling the industrial robot this way can only be used for certain use cases in that real-time control is not involved.

ACKNOWLEDGMENTS

The completion of this cooperative study and internship could not have been possible without help and guidance from my intern-COOP supervisor Mr. Thana Cheewasupakorn (Robotic Technical support engineer) for providing me with all the support and troubleshooting help when the project is at a halt during development.

I would like to show my gratitude to my advisor for the capstone project, Dr. Poom Konghuayrob, who has been helping me with the project and providing me with ideas and motivation. He also helps me with the idea of working in the industry which will be a part of the future automating transformation. However, this project could not have been finished without Badr Azougagh, an exchange student from Belgium, who helped me during the development of the project with coding and implementing the camera. Not only that, but he also helps me as a companion while doing the project.

REFERENCES

[1] Owen-Hill, A. (2021) The Pros and cons of 5 robot programming methods, RoboDK blog. Available at: https://robodk.com/blog/robot-programming-methods-pros-and-cons (Accessed: 10 May 2023).

[2] Industrial Robotic Arm (no date) Intel. Available at: https://www.intel.com/content/www/us/en/robotics/robotic-arm.html

[3] (No date) ABB-Robot Web Services. ABB. Available at: https://developercenter.robotstudio.com/api/RWS (Accessed: January 2022).

[4] Real Python (2022) Python's Requests Library (guide), Real Python. Real Python. Available at: https://realpython.com/python-requests/ (Accessed: January 2023).

[5] Team, G.L. (2022) OpenCV tutorial: A guide to learn OpenCV in Python, Great Learning Blog: Free Resources what Matters to Shape your Career! Available at: https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/#:~:text=OpenCV%20is%20a%20Python%20library,%2C%20face%20recognition%2C%20and%20tracking (Accessed: 10 May 2023).

[6] Basler AG (no date) Pylon open source, Basler AG. Available at: https://www.baslerweb.com/en/products/basler-pylon-camera-software-suite/pylon-open-source-projects (Accessed: 10 May 2023).

[7] What is a plc? (No date) AMCI. Available at: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc (Accessed: 10 May 2023).

[8] AG, B. (no date) ACA2500-14GC#, acA2500-14gc | Basler Product Documentation. Available at: https://docs.baslerweb.com/aca2500-14g (Accessed: 25 April 2023).

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.